KERJA PRAKTIK – EF234603
Desain dan Implementasi Prototipe Antarmuka Aplikasi *Mobile Watch* Berbasis *Flutter* Dengan Dukungan *Tomcat Website Server*

PT Telkom Indonesia Regional IV Jateng DIY
Jl. Pahlawan no 10 Semarang.

Periode : 15 Juli 2024 – 15 Oktober 2024

Oleh:

Beauty Valen Fajri **5025211227**
Meyroja Jovancha Firoos **5025211204**

Pembimbing Departemen
Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing Lapangan
Adhi Widyanto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

KERJA PRAKTIK – EF234603
Desain dan Implementasi Prototipe Antarmuka Aplikasi *Mobile Watch* Berbasis *Flutter* dengan Dukungan *Tomcat Website Server*

PT Telkom Indonesia Regional IV Jateng DIY
Jl. Pahlawan no 10 Semarang.

Periode : 15 Juli 2024 – 15 Oktober 2024

Oleh:

Beauty Valen Fajri	**5025211227**
Meyroja Jovancha Firoos	**5025211204**

Pembimbing Departemen
Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing Lapangan
Adhi Widyanto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika
Cerdas Institut Teknologi Sepuluh Nopember
Surabaya 2024

LEMBAR PENGESAHAN
KERJA PRAKTIK

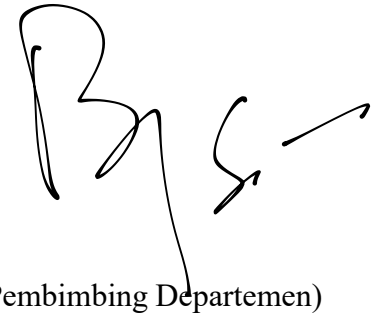
Desain dan Implementasi Prototipe Antarmuka Aplikasi *Mobile Watch* Berbasis *Flutter* dengan Dukungan *Tomcat Website Server*

Oleh:

Beauty Valen Fajri	5025211227
Meyroja Jovancha Firoos	5025211204

Disetujui oleh Pembimbing Kerja Praktik:

1. Bagus Jati Santoso, S.Kom., Ph.D.
   NIP. 198611252018031001



(Pembimbing Departemen)

2. Adhi Widyanto, S.Kom., M.Kom.

(Pembimbing Lapangan)

Surabaya, 12 November 2024

Desain dan Implementasi Prototipe Antarmuka Aplikasi *Mobile Watch* Berbasis *Flutter* dengan Dukungan *Tomcat Website Server*

Nama Mahasiswa 1	: Beauty Valen Fajri (5025211227)
Nama Mahasiswa 2	: Meyroja Jovancha Firoos (5025211204)
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing Lapangan	: Adhi Widyanto, S.Kom., M.Kom.

ABSTRAK

Penggunaan teknologi *mobile* saat ini semakin berkembang dan memberikan banyak kemudahan dalam berbagai aspek kehidupan, termasuk dalam pengelolaan data. PT Telkom Indonesia Regional IV Jawa Tengah ingin membuat sistem *Tomcat* yang sudah dibuat dalam bentuk *website* ke dalam aplikasi *mobile* agar memudahkan pengguna dalam memonitor data secara *real-time* melalui perangkat *mobile*.

Proses perancangan melibatkan analisis kebutuhan pengguna, desain antarmuka pengguna (UI/UX), serta pengembangan aplikasi berbasis *Android*. Dalam implementasinya, aplikasi ini menggunakan *PHPMyAdmin* sebagai *database* lokal untuk menyimpan data, dan *website* Tomcat digunakan sebagai referensi untuk pengelolaan serta pengujian data. Hasil dari implementasi ini menunjukkan bahwa aplikasi *Mobile Watch* mampu berfungsi dengan baik dalam melakukan monitoring dan pengelolaan data secara lokal. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman *mobile Dart* dengan *framework Flutter* yang terhubung dengan *database* lokal (*localhost*). Aplikasi ini diharapkan dapat mempermudah teknisi lapangan dalam pengerjaannya.

Kata Kunci: **Aplikasi *Mobile*, *Database* Lokal, *Tomcat*, Desain dan Implementasi, *Flutter, Dart, UI/UX, PHPMyAdmin*.**

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa karena atas berkat limpahan rahmat dan lindungan-Nya kami dapat melaksanakan salah satu kewajiban sebagai mahasiswa Departemen Informatika, yakni Kerja Praktik (KP).

Kami menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan KP maupun penyusunan buku laporan ini. Namun, kami berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Kami mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan KP ini.

Melalui laporan ini, kami juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pelaksanaan KP hingga penyusunan laporan. Orang-orang tersebut antara lain adalah :

1. Orang tua penulis.
2. Bapak Bagus Jati Santoso, S.Kom., Ph.D. selaku dosen pembimbing departemen.
3. Bapak Adhi Widyanto, S.Kom., M.Kom. dan Tim MSO selaku pembimbing lapangan kami di PT Telkom Indonesia Regional IV Jawa Tengah.
4. Teman-teman penulis yang senantiasa memberikan semangat dan dukungan kepada penulis selama melaksanakan KP.

Surabaya, 12 November 2024
Beauty Valen Fajri
Meyroja Jovancha Firoos

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam era digital yang semakin berkembang, kebutuhan akan aplikasi *mobile* yang efisien dan andal semakin meningkat. Aplikasi *mobile* tidak hanya berfungsi sebagai alat komunikasi, tetapi juga sebagai sarana untuk mengakses, memantau, dan mengelola berbagai data penting secara *real-time*. Dalam konteks ini, pengembangan aplikasi *mobile* yang terintegrasi dengan sistem *database* lokal menjadi salah satu solusi yang tepat untuk menyediakan layanan yang cepat dan responsif.

Penggunaan *database Localhost* dalam pengembangan aplikasi *mobile* menawarkan beberapa keuntungan, seperti kecepatan akses data yang lebih tinggi, penghematan *bandwidth*, serta kemampuan untuk bekerja dalam mode *offline*. Namun, untuk memastikan bahwa aplikasi dapat berjalan dengan baik dan terintegrasi secara efektif dengan layanan lain, dibutuhkan server lokal serta acuan yang andal. Di sinilah Tomcat berperan sebagai acuan utama dalam pengembangan aplikasi *mobile Watch.*

Tomcat adalah *Website* yang digunakan oleh Telkom untuk mengakses, memantau dan mengelola berbagai data penting. Dalam proyek ini, Tomcat digunakan sebagai acuan utama untuk mengembangkan aplikasi *mobile Watch* dan menggunakan *Database Localhost*. Dengan acuan Tomcat, aplikasi *Watch* diharapkan dapat mengimplementasikan *Website* Tomcat ke dalam aplikasi *mobile*, untuk mengakses dan mengelola data dengan cara yang lebih praktis, terstruktur dan aman.

Proyek "*Mobile Watch*" bertujuan untuk mengembangkan sebuah aplikasi yang dapat digunakan untuk memantau data secara *real-time* dengan memanfaatkan *Database Localhost*, serta mengacu pada *Website* Tomcat. Aplikasi ini dirancang untuk memberikan pengguna kemampuan untuk mengakses informasi penting dengan cepat dan praktis, serta memastikan data selalu tersedia meskipun dalam kondisi konektivitas yang terbatas.

Dengan latar belakang tersebut, laporan ini akan membahas proses desain dan implementasi aplikasi *mobile Watch*, serta bagaimana integrasi antara *Database Localhost* dengan aplikasi *mobile Watch*, yang mengacu pada *Website* Tomcat untuk mencapai kinerja aplikasi yang optimal.

1.2. Tujuan

Tujuan Kerja Praktik ini adalah untuk menyelesaikan kewajiban kuliah Kerja Praktik di Institut Teknologi Sepuluh Nopember dengan beban 4 SKS. Selain itu juga untuk membantu PT Telkom Indonesia Regional IV Jawa Tengah dalam memonitor data secara *real-time* melalui perangkat *mobile*. Tujuan dari pengimplementasian aplikasi tersebut antara lain :

1. Aplikasi ini dirancang untuk memudahkan teknisi PT Telkom Indonesia dalam memonitor data secara *real-time*, sehingga proses pengawasan dapat dilakukan dengan lebih cepat dan efisien.
2. Desain antarmuka yang responsif dan *user-friendly* membantu teknisi dalam mengoperasikan aplikasi dengan lebih mudah, sehingga mengurangi potensi kesalahan dalam pengelolaan data.
3. Dengan akses *real-time* ke data yang dibutuhkan, manajemen dan teknisi dapat mengambil keputusan lebih cepat dan tepat, yang dapat meningkatkan produktivitas dan efektivitas operasional.

1.3. Manfaat

Aplikasi *Mobile* Watch yang dikembangkan dengan integrasi *database Localhost* dan mengacu pada *website* Tomcat menawarkan berbagai manfaat signifikan, antara lain :

1. Aplikasi ini memungkinkan akses data secara *real-time*, pengguna dapat memantau dan mengelola informasi serta data penting dengan efisien. Akses ini didukung oleh penggunaan *database* lokal, yang juga memungkinkan aplikasi untuk tetap berfungsi meskipun dalam kondisi *offline*, mengurangi ketergantungan pada konektivitas internet dan meningkatkan keandalan aplikasi.
2. Aplikasi *Mobile Watch* mengimplementasikan fitur-fitur pengelolaan data yang lebih terstruktur dan praktis. Hal ini memastikan bahwa pengguna dapat mengolah serta mengakses data dengan lebih efektif dan aman. Aplikasi ini, dengan demikian, berperan penting dalam mendukung operasional kerja yang membutuhkan pemantauan data yang handal dan responsif, khususnya dalam lingkungan dengan kebutuhan akses informasi yang tinggi dan mendesak.

1.4. Rumusan Masalah

Berikut ini merupakan rumusan masalah pada Kerja Praktik pembuatan Aplikasi *Mobile Watch* Menggunakan *Database Localhost* Dengan Acuan *Website* Tomcat. Berikut ini rumusan masalah pada KP pembuatan Aplikasi *Mobile Watch* Menggunakan *Database Localhost* Dengan Acuan *Website* Tomcat :

1. Bagaimana merancang dan mengimplementasikan aplikasi *Mobile Watch* yang mampu mengintegrasikan *database Localhost* untuk mendukung akses dan pengelolaan data secara *real-time* dengan kecepatan dan keandalan yang optimal?
2. Bagaimana memastikan aplikasi *Mobile Watch* dapat mengadopsi fitur-fitur utama dari *website* Tomcat untuk memungkinkan pemantauan dan manajemen data yang aman, terstruktur, dan efisien?
3. Bagaimana mengatasi tantangan teknis dalam menjaga fungsi aplikasi *Mobile Watch* tetap berjalan meskipun dalam kondisi konektivitas yang terbatas, sehingga aplikasi dapat diandalkan dalam berbagai situasi operasional?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja Praktik ini dilaksanakan pada waktu dan tempat sebagai berikut :

Lokasi : PT Telkom Indonesia Regional IV Jateng DIY

Alamat: Jl Pahlawan no 10 Semarang

Waktu : 15 Juli 2024 - 15 Oktober 2024

1.6. Metodologi Kerja Praktik

Tahapan pengerjaan kerja praktik dapat dijabarkan sebagai berikut :

1. **Perumusan Masalah**

   Untuk memahami permasalahan yang perlu diselesaikan, diberikan penjelasan alasan di balik kebutuhan akan aplikasi ini. Rincian tentang bagaimana alur sistem akan berjalan juga diuraikan secara mendetail. Penjelasan ini diberikan oleh pembimbing lapangan selama Kerja Praktik. Dari penjelasan tersebut, diperoleh catatan-catatan penting yang memberikan gambaran tentang sistem berbasis *website* ini. Berdasarkan informasi tersebut, diputuskan untuk membangun sistem berbasis aplikasi dengan memanfaatkan bahasa pemrograman *mobile Dart* dengan *framework Flutter*, *Python* untuk *cleaning* data, serta menggunakan *database PHPMyAdmin*.

2. **Studi Literatur**

   Setelah menetapkan *database*, bahasa pemrograman, dan *tools* tambahan yang akan digunakan, dilakukan studi literatur untuk memahami cara penerapannya dalam pengembangan sistem yang diperlukan. Pada tahap ini, proses pencarian, pembelajaran, pengumpulan, dan pemahaman informasi serta literatur yang relevan dilakukan untuk mendukung implementasi aplikasi ini. Informasi tersebut dapat diperoleh dari internet, terutama untuk istilah-istilah umum yang terkait dengan implementasi sistem informasi.

3. **Analisis dan Perancangan Sistem**

   Langkah ini mencakup penjelasan awal mengenai sistem, termasuk cara kerjanya dalam berbagai skenario. Dari penjelasan awal tersebut, diperoleh gambaran umum mengenai kebutuhan fungsional dan non-fungsional secara garis besar. Selanjutnya, dilakukan klarifikasi dan spesifikasi lebih lanjut terhadap kebutuhan-kebutuhan tersebut. Sebagai hasilnya, dibuatlah diagram kasus penggunaan yang menggambarkan berbagai skenario penggunaan untuk sistem prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server*.

4. **Implementasi Sistem**

   Implementasi sistem didasarkan pada perancangan dan analisis yang telah dilakukan sebelumnya, serta pada kasus penggunaan dan pemilihan *tools* yang telah ditentukan. Pada tahap ini, terdapat dua pekerjaan utama

yang dilakukan yakni desain antarmuka *mobile* (*Front-end*) dan sistem *database* (*Localhost*), serta penggunaan bahasa pemrograman *Python* untuk *cleaning* data. Hal ini dilakukan secara bertahap dengan target progres yang ditetapkan setiap hari untuk meningkatkan hasil dari hari sebelumnya. Kemajuan dalam penyelesaian aplikasi terus dipantau oleh pembimbing lapangan Telkom PT Telkom Indonesia Regional IV Jateng DIY.

5. **Pengujian dan Evaluasi**

   Pengujian dilakukan dengan mengevaluasi fitur-fitur yang telah dikembangkan. Kesesuaian sistem dengan kebutuhan yang ditetapkan akan menentukan keberhasilan proses pengujian. Evaluasi ini akan menghasilkan penilaian apakah sistem sudah memenuhi tujuan dan kebutuhan yang diinginkan atau belum.

6. **Kesimpulan dan Saran**

   Kesimpulan diambil berdasarkan hasil pengujian dan evaluasi sistem yang telah dilakukan. Kesimpulan ini mencakup pencapaian tujuan dan kesesuaian sistem dengan kebutuhan pengguna. Selain itu, saran diberikan untuk perbaikan dan pengembangan lebih lanjut, termasuk rekomendasi peningkatan fitur atau aspek lain yang memerlukan perhatian di masa mendatang.

1.7. Sistematika Laporan

Laporan KP ini terdiri dari tujuh bab dengan rincian sebagai berikut :

1. **Bab I Pendahuluan**

   Pada bab ini berisi tentang latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi kerja praktik, dan sistematika laporan.

2. **Bab II Profil Perusahaan**

   Pada bab ini berisi tentang profil perusahaan tempat melaksanakan kerja praktik, yakni PT Telkom Indonesia Regional IV Jateng DIY.

3. **Bab III Tinjauan Pustaka**

   Pada bab ini berisi tentang dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

4. **Bab IV Analisis dan Perancangan Sistem**

   Pada bab ini berisi hasil pembelajaran atau analisis terhadap apa saja yang diperlukan dan harus diperhatikan dalam pengembangan aplikasi yang dikerjakan selama KP. Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

5. **Bab V Implementasi Sistem**

   Pada bab ini dijelaskan tentang uraian mengenai tahapan yang dilakukan untuk proses implementasi aplikasi.

6. **Bab VI Pengujian dan Evaluasi**

   Pada bab ini dijelaskan tentang hasil pengujian dan evaluasi dari

sistem aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

7. **Bab VII Kesimpulan dan Saran**
   Pada bab ini dijelaskan tentang kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil PT Telkom Indonesia Regional IV Jateng DIY

PT Telkom Indonesia Regional IV Jateng DIY merupakan salah satu unit regional dari PT Telekomunikasi Indonesia Tbk (Telkom), yang berfokus pada layanan telekomunikasi dan jaringan di wilayah Jawa Tengah dan Daerah Istimewa Yogyakarta (DIY). Sebagai penyedia layanan komunikasi terbesar di Indonesia, Telkom Regional IV bertanggung jawab untuk menyediakan berbagai layanan, termasuk telepon, internet, dan jaringan data bagi pelanggan individu, bisnis, dan institusi pemerintah di wilayah tersebut. Melalui infrastruktur yang modern dan inovatif, Telkom Regional IV Jateng DIY berperan penting dalam mendukung transformasi digital di Jawa Tengah dan DIY.

Telkom Regional IV terus berupaya meningkatkan kualitas layanan dengan memperluas jaringan *fiber optic* dan memperkenalkan teknologi terbaru guna memenuhi kebutuhan pelanggan yang semakin kompleks. Selain itu, Telkom juga berkomitmen untuk berkontribusi terhadap perkembangan ekonomi lokal melalui berbagai program tanggung jawab sosial perusahaan (CSR) dan kemitraan strategis dengan pemerintah daerah dan sektor swasta. Dengan dedikasi yang kuat untuk menjaga kepuasan pelanggan, Telkom Regional IV Jateng DIY terus memperkuat posisinya sebagai pemimpin di industri telekomunikasi di wilayah tersebut [1].

2.2. Profil CNOP & OLO *Fulfillment Unit Managed Service Operation* (MSO)

CNOP & OLO *Fulfillment Unit Managed Service Operation* (MSO) yakni salah satu unit operasional yang berada di bawah PT Telkom Indonesia, yang berfokus pada pengelolaan dan pemenuhan layanan *Connectivity Network Operation Process* (CNOP) serta *Other Licensed Operator* (OLO). Unit ini bertugas memastikan semua aspek layanan jaringan, baik internal maupun eksternal, berjalan dengan optimal. CNOP & OLO *Fulfillment* Unit MSO berperan dalam proses instalasi, pemeliharaan, dan manajemen jaringan yang melibatkan berbagai operator telekomunikasi, termasuk layanan yang disediakan oleh pihak ketiga. Melalui pendekatan layanan yang terintegrasi, unit ini membantu memastikan bahwa pelanggan mendapatkan layanan konektivitas yang handal dan berkualitas tinggi.

Selain itu, CNOP & OLO *Fulfillment* Unit MSO juga berperan strategis dalam mendukung operasional harian yang terkait dengan infrastruktur telekomunikasi. Unit ini bekerja sama dengan berbagai pihak untuk menjaga stabilitas dan kinerja jaringan, serta memastikan bahwa semua kebutuhan pelanggan, baik korporat maupun retail, terpenuhi sesuai dengan standar yang ditetapkan. Dengan kompetensi yang kuat dalam manajemen layanan terkelola, unit ini menjadi pilar penting dalam menjaga efisiensi dan kualitas layanan telekomunikasi yang dioperasikan oleh PT Telkom Indonesia.

2.3. Lokasi PT Telkom Indonesia Regional IV Jateng DIY

Jl. Pahlawan no 10 Semarang

2.4. Visi dan Misi Perusahaan

Visi:
Menjadi digital *telco* pilihan utama untuk memajukan masyarakat [1].

Misi :

a. Mempercepat pembangunan infrastruktur dan platform digital cerdas yang berkelanjutan, ekonomis, dan dapat diakses oleh seluruh masyarakat.
b. Mengembangkan talenta digital unggulan yang membantu mendorong kemampuandigital dan tingkat adopsi digital bangsa.
c. Mengorkestrasi ekosistem digital untuk memberikan pengalaman digital pelangganterbaik [1].

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan mengenai dasar teori yang digunakan selama proses pengerjaan dan pengembangan aplikasi.

3.1. *Dart*

Dart adalah bahasa pemrograman yang bersifat *open-source*. Ini adalah bahasa pemrograman yang sepenuhnya berorientasi objek dengan sintaksis gaya C. *Dart* awalnya dikembangkan oleh Google pada tahun 2011 dan kemudian disetujui sebagai standar oleh ECMA. *Dart* adalah bahasa yang dioptimalkan untuk mengembangkan aplikasi *cross-platform*. *Dart* digunakan untuk pengembangan aplikasi *mobile and website* sekaligus [2].

3.2. *Flutter*

Flutter adalah SDK *open-source* untuk mengembangkan aplikasi *mobile* yang memiliki kinerja tinggi dan lebih dapat diandalkan untuk sistem operasi seperti iOS dan *Android*. Fitur signifikan dari *Flutter* adalah kompilasi *just-in-time* yang mengeksekusi kode komputer yang mencakup kompilasi selama eksekusi program pada saat *runtime* daripada sebelum eksekusi [3].

3.3. Android Studio

Android studio adalah *Integrated Development Environment* (IDE) resmi untuk pengembangan aplikasi Android, yang didasarkan dari *IntelliJ IDEA* sebuah ide popular yang dikembangkan oleh *Jetbrains* untuk pemrograman *Java* dan *Kotlin*. Android studio menawarkan fitur yang mampu meningkatkan produktivitas dalam pengembangan aplikasi [4].

3.4. PHP

PHP (Hypertext Preprocessor) adalah bahasa pemrograman *open-source* yang banyak digunakan untuk pengembangan web dan bisa disisipkan ke dalam HTML. Berbeda dengan *JavaScript* yang berjalan disisi *client*, *PHP* dieksekusi di server, menghasilkan *HTML* yang dikirim ke klien tanpa mengungkapkan kode aslinya [5].

3.5. PHP*MyAdmin*

PHPMyAdmin adalah perangkat lunak gratis yang ditulis dalam *PHP* yang dimaksudkan untuk menangani administrasi server basis data *MySQL* atau *MariaDB*. *PHPMyAdmin* digunakan untuk melakukan sebagian besar tugas administrasi, termasuk membuat basis data, menjalankan *query*, dan menambahkan akun pengguna [5].

3.6. *OpenStreetMap*

OpenStreetMap adalah layanan peta gratis yang dapat diedit dari seluruh dunia yang dibuat oleh pengguna. Aplikasi ini didirikan di Inggris pada tahun 2004 sebagai hasil dari kelangkaan data peta berkualitas tinggi yang tersedia untuk umum. Jalan, bangunan, alamat, toko dan perusahaan, area yang diminati, rel kereta api, jalan setapak, angkutan umum, penggunaan lahan dan fitur alam, dan lebih banyak lagi informasi lainnya yang disertakan dalam *OpenStreetMap* [6].

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN
PERANCANGAN SISTEM

4.1. Analisis Sistem

Pada bab ini akan dijelaskan mengenai tahapan dalam membangun prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server,* berupa analisis dari sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan fungsional.

4.1.1. Definisi Umum Aplikasi

Aplikasi *Watch* adalah aplikasi internal perusahaan yang digunakan untuk memantau status tower (*down* atau *up*) dan fitur tambahan seperti akses data Tsel dan Olo. Aplikasi ini menggunakan sistem akun terdistribusi, sehingga hanya karyawan MSO yang memiliki akun yang dapat mengaksesnya. Aplikasi ini memiliki satu tampilan utama yang digunakan oleh karyawan MSO untuk input data secara umum.

4.1.2. Analisis Kebutuhan

Dalam aplikasi ini, terdapat fungsi-fungsi yang harus dipenuhi oleh sistem. Kebutuhan ini terbagi ke dalam dua jenis, yakni kebutuhan fungsional dan kebutuhan non- fungsional.

a. Kebutuhan Fungsional

Kebutuhan fungsional pada aplikasi ini menjelaskan bagaimana sistem itu bekerja. Kebutuhan fungsional dari prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server* dijelaskan pada Tabel 4.1.

Tabel 4.1 Kebutuhan Fungsional Aplikasi *Mobile Watch*

Kode Kebutuhan	Deskripsi Kebutuhan
F-001	Melakukan *Login*
F-002	Memonitor Alarm Gangguan
F-003	Melakukan Pengecekan Status Gangguan
F-004	Melakukan Akses ke Data OLO (*Other Licensed Operator*)
F-005	Memvalidasi atau Melakukan Edit Pada Data OLO
F-006	Melakukan Akses ke Data Telkomsel (TSel)
F-007	Memvalidasi atau Melakukan Edit Pada Data TSel
F-008	Melakukan Akses ke Data Order Aktif

Kode Kebutuhan	Deskripsi Kebutuhan
F-001	Melakukan Login
F-002	Memonitor Alarm Gangguan
F-003	Melakukan Pengecekan Status Gangguan
F-009	Memvalidasi Data Pada Order Aktif

Pada Tabel 4.1 terdapat kebutuhan fungsional prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server*, yang dapat diakses oleh aktor dan memiliki fitur yang berbeda bergantung pada *roles* dari aktor.

b. Kebutuhan Non-fungsional

Kebutuhan non-fungsional adalah kebutuhan pengguna untuk mendefinisikan bagaimana batasan dan karakteristik dari sebuah sistem yang dibangun. Kebutuhan non- fungsional dari Aplikasi *mobile watch* terdapat pada Tabel 4.2.

Tabel 4.2 Kebutuhan Non-Fungsional Aplikasi *Mobile Watch*

Kode Kebutuhan	Deskripsi Kebutuhan
NF-001	Sistem dapat diakses oleh pengguna
NF-002	Sistem dapat diakses pada iOS maupun Android
NF-003	Sistem memiliki tampilan antarmuka yang mudah dipahami
NF-004	Sistem memiliki audit log untuk merekam setiap aktivitas yang sudah dilakukan oleh admin dan user
NF-005	Sistem dapat menangani *requests* per detik untuk memastikan respon yang cepat pada saat akses data OLO, data TSel dan data order aktif oleh admin maupun user
NF-006	Sistem dapat memastikan bahwa data yang digunakan dalam sistem harus terlindung dari akses yang tidak berwenang

Pada Tabel 4.2 terdapat kebutuhan non-fungsional prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server*, yang meliputi akses pengguna, OS yang kompatibel, antarmuka yang mudah dipahami, dan keamanan data pengguna.

4.2. Diagram Kasus Penggunaan

Pembahasan dengan pembimbing lapangan tentang fitur-fitur yang perlu

ada dalam prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server,* menghasilkan beberapa fitur yang dijadikan diagram kasus penggunaan (*Use Case* Diagram) sehingga memudahkan untuk dipahami. *Use Case*Diagram yang telah dibuat dapat dilihat pada Gambar 4.2.1.

Gambar 4.2.1 Diagram *Use Case* Aplikasi *Mobile Watch*

Pada Gambar 4.2.1 terdapat 2 *roles,* yaitu User dan Admin. Admin memiliki akses untuk memonitor alarm gangguan, memeriksa status gangguan, melihat rekomendasi, serta memvalidasi atau mengedit data dari OLO (*Other Licensed Operator*) dan TSEL (Telkomsel). Selain itu, Admin juga dapat mengakses dan memvalidasi atau mengedit data order yang aktif. Di sisi lain, *user* memiliki akses yang lebih terbatas, namun tetap dapat memeriksa status gangguan dan melihat rekomendasi

yang diberikan oleh sistem. User hanya tidak dapat melakukan Validasi atau Edit Data Tsel, OLO, dan Order Aktif.

4.3. Spesifikasi Kasus Penggunaan

4.3.1. Melakukan Login

Tabel 4.3 berikut merupakan tabel *use case* dari aplikasi *mobile watch* melakukan *login*.

Tabel 4.3 *Use Case* Aplikasi *Mobile Watch* Melakukan *Login*

Nama	Melakukan *login*
Kode	UC001
Deskripsi	Admin ataupun *User* dapat masuk ke akun sesuai *role*
Tipe	Fungsional
Pemicu	Admin atau *user* menekan tombol 'log in' setelah mengisi *username* dan *password* pada halaman login aplikasi *mobile watch*
Aktor	Admin ataupun *User*
Kondisi Awal	*Form login* ditampilkan
Kondisi Akhir	Admin atau *user* dapat melakukan kegiatan pada sistem sesuai kewenangannya
Alur Kejadian Secara Normal	*1.* Admin atau *user* mengisi *form login* *2.* Admin atau *user* menekan tombol '*Log in'* *3.* Sistem mencocokkan data *login* dengan *database* *4.* Sistem menampilkan halaman utama
Alur Kejadian Alternatif	1. Aktor tidak mengisi *form login* dengan lengkap a. Sistem memberi peringatan bahwa kolom harus diisi. b. Kembali ke alur normal nomor 2. 2. Data yang diinputkan tidak cocok dengan basis data *a.* Sistem memberi peringatan bahwa email atau *password* salah. b. Kembali ke Alur Normal nomor 4. 3. Koneksi Internet tidak stabil atau putus a. Sistem memberi peringatan bahwa koneksi internet tidak tersedia dan meminta pengguna untuk mencoba lagi. b. Kembali ke nomor 2 ketika koneksi sudah pulih.
Pengecualian	-

4.3.2. Monitor Alarm Gangguan

Tabel 4.4 berikut merupakan tabel *use case* dari Aplikasi *Mobile Watch* monitor alarm gangguan.

Tabel 4.4 *Use Case* Aplikasi *Mobile Watch* Monitor Alarm Gangguan

Nama	Monitor Alarm Gangguan
Kode	UC002
Deskripsi	Admin ataupun *user* dapat melakukan monitor pada alarm yang mengalami gangguan
Tipe	Fungsional
Pemicu	Admin ataupun *user* menekan *button* alarm di halaman utama
Aktor	Admin ataupun *User*
Kondisi Awal	*Button* alarm ditampilkan
Kondisi Akhir	Admin ataupun *User* dapat melakukan pengecekan daerah yang mengalami gangguan pada *button* alarm
Alur Kejadian Secara Normal	1. Admin ataupun *User* menekan *button* alarm 2. Sistem akan menampilkan daerah yang mengalami kendala
Alur Kejadian Alternatif	1. Data tidak terbaca : sistem menampilkan pesan bahwa data tidak terbaca atau terjadi kesalahan teknis, dan meminta pengguna untuk mencoba lagi. 2. Kembali ke nomor 1 ketika masalah teratasi.
Pengecualian	-

4.3.3. Melakukan Pengecekan Status Gangguan

Tabel 4.5 berikut merupakan tabel *use case* dari aplikasi *mobile watch* melakukan pengecekan status gangguan.

Tabel 4.5 Tabel *Use Case Mobile Watch* Melakukan Pengecekan Status Gangguan

Nama	Melakukan Pengecekan Status Gangguan
Kode	UC003
Deskripsi	Admin ataupun *User* dapat melihat status gangguan secara detail
Tipe	Fungsional
Pemicu	Admin ataupun *User* menekan *button* 'Lihat Status' di bagian titik merah daerah gangguan pada *button* Alarm
Aktor	Admin ataupun *User*
Kondisi Awal	Admin ataupun *User* melihat tab pada bagian setiap daerah yang terkena gangguan
Kondisi Akhir	Admin ataupun *User* dapat menekan *button* 'Lihat Status' di setiap tab yang ditampilkan
Alur Kejadian Secara Normal	1. Admin ataupun *User* menekan tombol 'Lihat Status' pada *page* alarm 2. Sistem akan menampilkan status pada daerah yang terkena gangguan
Alur Kejadian Alternatif	1. Data status gangguan tidak terbaca a. Sistem menampilkan pesan bahwa data tidak terbaca atau terjadi kesalahan teknis, dan meminta pengguna untuk mencoba lagi. b. Kembali ke nomor 1 jika masalah teratasi.
Pengecualian	-

4.3.4. Melakukan Akses ke Data TSEL

Tabel 4.6 berikut merupakan tabel *use case* dari dari aplikasi *mobile watch* melakukan akses ke data TSEL.

Tabel 4.6 Tabel *Use Case Mobile Watch* Melakukan Akses ke Data TSEL

Nama	Melakukan Akses ke Data TSEL
Kode	UC004
Deskripsi	Admin ataupun *User* dapat mencari kebutuhan data TSEL berdasarkan nama Witel dan STO
Tipe	Fungsional
Pemicu	Admin ataupun *User* menekan *button Watch* TSEL di halaman utama
Aktor	Admin ataupun *User*
Kondisi Awal	*Button* Watch TSEL ditampilkan berdasarkan nama Witel dan STO
Kondisi Akhir	Sistem menampilkan data TSEL berdasarkan nama Witel dan STO
Alur Kejadian Secara Normal	1. Admin ataupun *User* menekan *button watch* TSEL 2. Sistem akan menampilkan data yang dibutuhkan yakni nama Witel dan STO 3. Sistem akan menampilkan data TSEL yang berdasarkan nama Witel dan STO
Alur Kejadian Alternatif	1. Data Tsel tidak terbaca a. sistem menampilkan pesan data tidak terbaca atau terjadi kesalahan teknis, dan meminta pengguna untuk mencoba lagi. b. Kembali ke nomor 1 setelah masalah teratasi.
Pengecualian	-

4.3.5. Memvalidasi atau Melakukan Edit Pada Data TSEL

Tabel 4.7 berikut merupakan tabel *use case* dari aplikasi *mobile watch* memvalidasi atau melakukan edit pada data TSEL.

Tabel 4.7 *Use Case Mobile Watch* Memvalidasi atau Melakukan Edit Data TSEL

Nama	Memvalidasi atau Melakukan Edit Pada Data TSEL
Kode	UC005
Deskripsi	Aktor dapat mengedit data TSEL sesuai dengan kebutuhannya
Tipe	Fungsional
Pemicu	Aktor menekan *button* validasi pada laman data TSEL
Aktor	Admin ataupun *User*
Kondisi Awal	Aktor ingin mengedit data TSEL
Kondisi Akhir	Aktor dapat melihat laman data TSEL yang diedit
Alur Kejadian Secara Normal	1. Aktor menekan tombol validasi pada halaman data TSEL. 2. Sistem akan menampilkan halaman validasi. 3. Aktor melakukan pengeditan sesuai kebutuhan. 4. Aktor menekan tombol simpan. 5. Sistem menampilkan hasil halaman data TSEL yang telah diedit.
Alur Kejadian Alternatif	1. Penyimpanan gagal karena kesalahan sistem. a. Sistem menampilkan pesan bahwa terjadi kesalahan saat menyimpan perubahan, dan meminta pengguna mencoba lagi. b. Kembali ke Alur Normal nomor 4. 2. Input karakter tidak valid. a. Sistem menampilkan bahwa input mengandung karakter yang tidak diperbolehkan dan meminta perbaikan. b. Kembali ke Alur Normal nomor 3.
Pengecualian	-

4.3.6. Melakukan Akses ke Data OLO

Tabel 4.8 berikut merupakan tabel *use case* dari aplikasi *mobile watch* melakukan akses ke data OLO.

Tabel 4.8 *Use Case Mobile Watch* Melakukan Akses ke Data OLO

Nama	Melakukan Akses ke Data OLO
Kode	UC006
Deskripsi	Aktor dapat melakukan akses untuk melihat ke data OLO
Tipe	Fungsional
Pemicu	Aktor menekan *button* data OLO
Aktor	Admin ataupun *User*
Kondisi Awal	Aktor ingin melakukan akses ke data OLO
Kondisi Akhir	Aktor dapat melihat data pada laman data OLO
Alur Kejadian Secara Normal	1. *User* menekan tombol untuk mengakses data OLO. 2. *User* mengisi nama pelanggan dan WITEL. 3. Sistem menampilkan informasi berdasarkan parameter yang telah dimasukkan.
Alur Kejadian Alternatif	1. Data OLO tidak terbaca a. sistem menampilkan pesan data tidak terbaca atau terjadi kesalahan teknis, dan meminta pengguna untuk mencoba lagi. b. Kembali ke nomor 1 setelah masalah teratasi.
Pengecualian	-

4.3.7. Memvalidasi atau Melakukan Edit Pada Data OLO

Tabel 4.9 berikut merupakan tabel *use case* dari aplikasi *mobile watch* memvalidasi atau melakukan edit pada data OLO.

Tabel 4.9 *Use Case Mobile Watch* Memvalidasi atau Melakukan Edit Data OLO

Nama	Memvalidasi atau Melakukan Edit Pada Data OLO
Kode	UC007
Deskripsi	Aktor dapat memvalidasi atau melakukan edit pada Data OLO
Tipe	Fungsional
Pemicu	Aktor menekan button validasi pada laman data OLO.
Aktor	Admin atau *User*
Kondisi Awal	Aktor ingin memvalidasi atau mengedit data OLO
Kondisi Akhir	Aktor dapat memvalidasi atau mengedit data OLO
Alur Kejadian Secara Normal	1. Aktor menekan *button* validasi pada laman data OLO 2. Sistem akan menampilkan halaman validasi 3. Aktor mengedit sesuai kebutuhan 4. Aktor menekan *button* simpan 5. Sistem menampilkan hasil halaman data OLO yang diedit
Alur Kejadian Alternatif	1. Penyimpanan gagal karena kesalahan sistem. a. Sistem menampilkan pesan bahwa terjadi kesalahan saat menyimpan perubahan, dan meminta pengguna mencoba lagi. b. Kembali ke Alur Normal nomor 4. 2. Input karakter tidak valid. c. Sistem menampilkan bahwa input mengandung karakter yang tidak diperbolehkan dan meminta perbaikan. d. Kembali ke Alur Normal nomor 3.
Pengecualian	-

4.3.8. Melakukan Akses ke Data Order Aktif

Tabel 4.10 berikut merupakan tabel *use case* dari aplikasi *mobile watch* melakukan akses ke data order aktif.

Tabel 4.10 *Use Case Mobile Watch* Melakukan Akses Data Order Aktif

Nama	Melakukan Akses ke Data Order Aktif
Kode	UC008
Deskripsi	Aktor dapat melakukan akses ke data Order Aktif
Tipe	Fungsional
Pemicu	Aktor menekan *button* Order Aktif
Aktor	Admin ataupun *User*
Kondisi Awal	Aktor ingin mengakses data Order Aktif
Kondisi Akhir	Aktor dapat melihat data Order Aktif.
Alur Kejadian Secara Normal	1. Aktor menekan *button* Order Aktif. 2. Sistem menampilkan data Order Aktif Assurance dan Fulfillment.
Alur Kejadian Alternatif	1. Data Order Aktif tidak terbaca a. sistem menampilkan pesan data tidak terbaca atau terjadi kesalahan teknis, dan meminta pengguna untuk mencoba lagi. b. Kembali ke nomor 1 setelah masalah teratasi.
Pengecualian	-

4.3.9. Memvalidasi Data pada Order Aktif

Tabel 4.11 berikut merupakan tabel *use case* dari aplikasi *mobile watch* memvalidasi data pada order aktif.

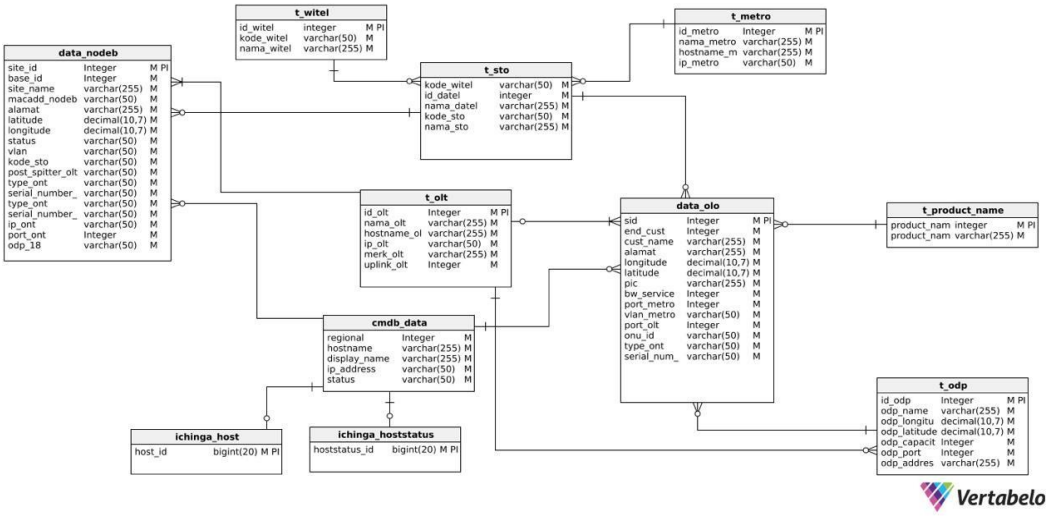
Tabel 4.11 *Use Case Mobile Watch* Memvalidasi Data pada Order Aktif

Nama	Memvalidasi Data pada Order Aktif
Kode	UC009

Deskripsi	Aktor dapat memvalidasi data pada Order Aktif
Tipe	Fungsional
Pemicu	Aktor menekan tombol *upload* foto lokasi atau uji petik.
Aktor	Admin ataupun *User*
Kondisi Awal	Aktor ingin melakukan validasi dengan mengunggah foto lokasi atau uji petik .
Kondisi Akhir	File foto lokasi atau uji petik berhasil terunggah.
Alur Kejadian Secara Normal	1. Aktor menekan *button upload* foto lokasi atau uji petik. 2. Aktor memilih file foto yang ingin diunggah 3. Sistem menerima file unggahan aktor.
Alur Kejadian Alternatif	1. File foto gagal diunggah a. Sistem menampilkan pesan bahwa terjadi kesalahan saat menyimpan perubahan, dan meminta pengguna mencoba lagi. b. Kembali ke Alur Normal nomor 2. 2. Ukuran file terlalu besar. a. Sistem menampilkan pesan bahwa file foto terlalu besar dan memberi batasan file yang dapat diunggah. b. Kembali ke Alur Normal nomor 2.
Pengecualian	-

4.4. *Conceptual* Data Model

Error! Reference source not found. berikut adalah *conceptual* data model dari aplikasi *mobile watch*.

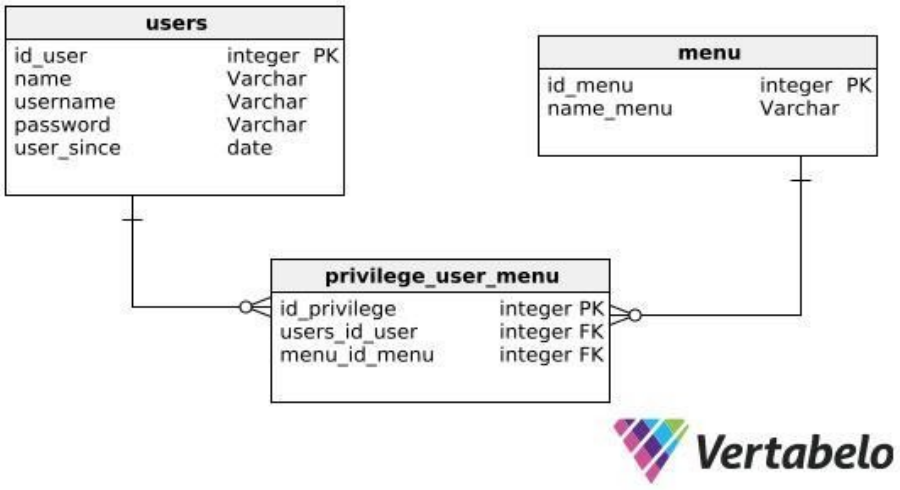


Gambar 4.4.1 Conceptual Data Model Aplikasi Mobile Watch

Error! Reference source not found. menunjukkan 11 tabel yang mewakili data yang akan digunakan dalam aplikasi *watch mobile*. Tabel utama, seperti **data_nodeb** dan **data_olo**, menyimpan informasi tentang situs, pelanggan, dan parameter teknis terkait. Sementara itu, tabel referensi, seperti **t_witel**, **t_sto**, **t_olt**, dan **t_metro**, menyediakan detail mengenai wilayah, sentral telekomunikasi, perangkat OLT, dan jaringan metro. Tabel **cmdb_data** mengelola data konfigurasi perangkat yang terhubung dengan tabel status perangkat, yaitu **icinga_host** dan **icinga_hoststatus**. Struktur hubungan antar tabel ini mencakup informasi perangkat, pelanggan, dan konfigurasi jaringan, memungkinkan integrasi data dari perangkat hingga layanan pelanggan.

4.5. *Physical* Data Model

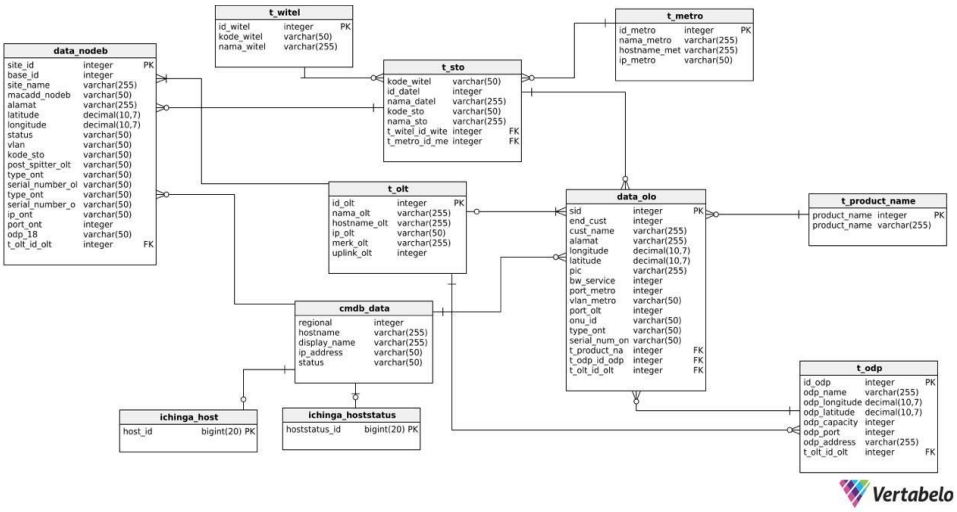
Gambar 4.5.1 berikut *Physical Data* Model dari Aplikasi *Mobile Watch*.



Gambar 4.5.1 *Physical* Data Model *User Login* Aplikasi *Watch Mobile*

Model pada Gambar 4.5.1 menetapkan detail teknis, termasuk tipe data danrelasi, untuk mengatur hak akses pengguna terhadap menu tertentu dalam aplikasi.

1. **Tabel *user***: Menyimpan informasi pengguna, seperti nama, *username*, *password*, dan tanggal registrasi (*user_since*).
2. **Tabel menu:** Berisi data mengenai menu aplikasi yang dapat diakses pengguna.
3. **Tabel *privilege_user_menu***: Berfungsi sebagai tabel relasi antara tabel ***user*** dan **menu**, mengatur hak akses setiap pengguna ke menu yang tersedia, dengan ***users_id_user*** dan **menu_id_menu** sebagai foreign key.



Gambar 4.5.2 *Physical* Data Model Aplikasi *Mobile Watch*

Gambar 4.5.2 menampilkan *Physical* Data Model (PDM) aplikasi yang mengelola data terkait node, STO, witel, metro, produk, serta data operasional yang disimpan dalam beberapa tabel inti, seperti **data_nodeb**, **t_witel**, **t_sto**, **t_metro**, **t_olt**, **data_olo**, **t_product_name**, dan **t_odp**. **Tabel data_nodeb**: Menyimpan informasi rinci tentang node B, seperti lokasi, alamat, tipe ONT, dan detail koneksi termasuk kode STO dan nomor seri.

1. **Tabel t_witel, t_sto, dan t_metro**: Menyediakan data geografis dan administratif seperti nama wilayah, kode area, dan detail STO.

2. **Tabel t_olt**: Menyimpan informasi perangkat OLT, seperti tipe, hostname, dan alamat IP.

3. **Tabel data_olo**: Berisi data operasional untuk pelanggan akhir, mencakup alamat, informasi teknis, dan ID produk.

4. **Tabel t_product_name dan t_odp**: Menyimpan informasi mengenai produk dan Optical Distribution Point (ODP).

Hubungan antar tabel dijaga melalui *foreign key* yang memastikan integritas data, misalnya antara **data_nodeb** dengan **t_olt**, serta **data_olo** dengan **t_product_name** dan **t_odp**.

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

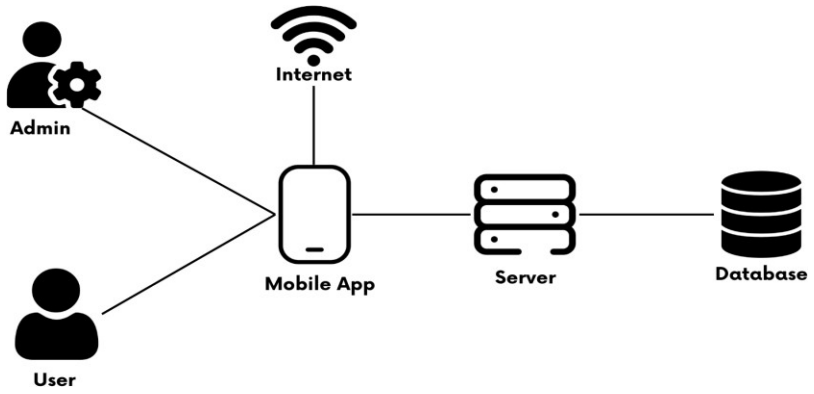
Bab ini membahas tentang implementasi dari perancangan sistem dan pengaplikasian sistem dalam bentuk prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server*.

5.1. Implementasi Sistem

Sistem yang dibuat merupakan sistem digital untuk manajemen data pada aplikasi *mobile watch*. Fitur-fitur utama dari aplikasi ini meliputi akses oleh admin dan pengguna melalui aplikasi seluler yang memerlukan koneksi internet. Adapun fitur-fitur dari *watch* adalah memonitor alarm gangguan, memeriksa status gangguan, melihat rekomendasi, serta memvalidasi atau mengedit data dari OLO (*Other Licensed Operator*) dan TSEL (Telkomsel). Aplikasi ini merupakan penerapan dalam bentuk *mobile apps* dari pengembangan web yang sudah ada dengan penambahan beberapa fitur. Aplikasi ini dibuat menggunakan *Flutter* untuk bagian *front-end*.

5.2. Implementasi Arsitektur Sistem

Pada bagian ini akan digambarkan arsitektur sistem. Adapun diagram arsitektur sistem yang diterapkan untuk aplikasi *mobile watch* terdapat pada Gambar 5.2.1.

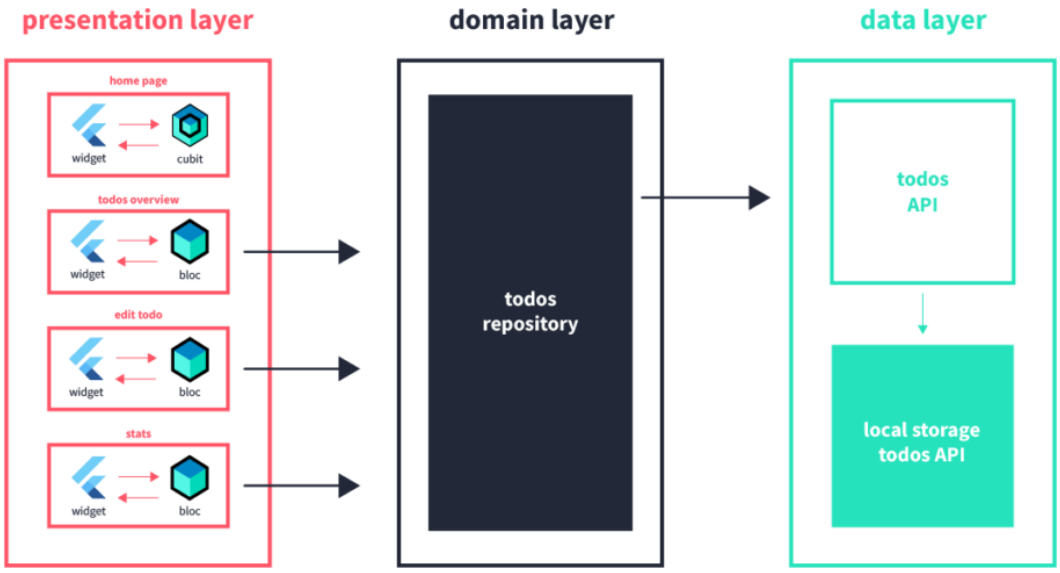


Gambar 5.2.1 Diagram Arsitektur Aplikasi *Mobile Watch*

Pada Gambar 5.2.1, ditampilkan diagram arsitektur sistem untuk aplikasi *mobile watch.* Diagram ini menunjukkan bahwa admin dan pengguna dapat mengakses aplikasi seluler yang memerlukan koneksi internet. Data yang dimasukkan melalui aplikasi ini akan diintegrasikan oleh server dan disimpan dalam *database* yang terkait. Diagram ini juga memperlihatkan pembagian antara *front-end* aplikasi, yang berinteraksi langsung dengan pengguna melalui antarmuka *mobile app*, dan *back- end* aplikasi, yang mencakup integrasi dengan *server* dan *database*.

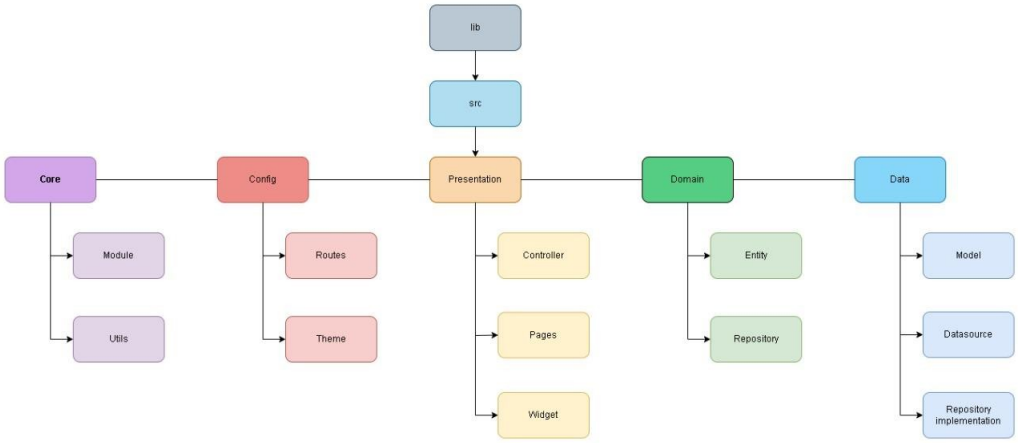
5.2.1. Implementasi *Front-End*

Pada bagian ini akan dibahas mengenai implementasi pada *front-end* aplikasi. *Front-end* sistem aplikasi ini dibangun menggunakan sebuah arsitektur yaitu *clean architecture*. *Clean Architecture* adalah pendekatan arsitektur perangkat lunak yang berfokus pada pemisahan kode berdasarkan tanggung jawab nya.



Gambar 5.2.1.1 *Clean Architecture* pada *Flutter*

Pada Gambar 5.2.1.1 terdapat tiga lapisan utama yang terdapat dalam arsitektur ini, yaitu *presentation layer*, berfungsi sebagai lapisan paling luar dan berinteraksi langsung dengan user melalui tampilan antarmuka aplikasi. Kemudian ada domain layer, bertanggung jawab untuk memproses input tersebut dan menghasilkan output. Output tersebut kemudian dikirim kembali ke *presentation layer* untuk ditampilkan kepada pengguna. Terakhir yaitu data layer, bertanggung jawab untuk berinteraksi dengan dunia nyata, misalnya dengan database, jaringan, atau sistem operasi. Data layer menyediakan data yang dibutuhkan oleh domain layer untuk memproses input dari pengguna.



Gambar 5.2.1.2 *Overview* dan Struktur *Watch App Mobile*

Pada Gambar 5.2.1.2 berisi tentang *overview* dan struktur aplikasi *mobile watch* untuk bagian *front- end* yang mengimplementasikan layer *clean architecture* yang ada pada Gambar 5.2.1.1. Berdasarkan ilustrasi dari Gambar 5.2.1.2, akan dijelaskan implementasi dari masing-masing komponen pada struktur *project*:

1. ***Entity***

Entity memegang peranan penting sebagai model inti yang merepresentasikan data dan perilaku dalam domain bisnis.

```
6   class User {
7     final int? id;
8     final String name;
9     final String username;
10    String? userSince;
11    User({
12      this.id,
13      required this.name,
14      required this.username,
15      this.userSince,
16    });
17  }
```

Gambar 5.2.1.3 Contoh Implementasi *Entity*

Kode di atas merupakan salah satu contoh implementasi dalam pembuatan sebuah *class entity, class User* tersebut memiliki properti (*id, name, username, userSince*) yang merepresentasikan atribut dari sebuah entitas pengguna. Class ini hanya berfokus pada pemodelan data (*id, name, username, userSince*) tanpa menyertakan logika atau perilaku lain, sehingga dapat dikategorikan sebagai *entity*. *Entity* ini digunakan untuk menyimpan dan merepresentasikan data pengguna, yang kemudian dipakai di bagian lain dari kode, seperti di *UserController* untuk *login* dan penyimpanan informasi pengguna ke dalam *SharedPreferences*.

2. ***Repository***

Repository adalah sebuah lapisan abstrak yang bertanggung jawab untuk mengelola dan menyediakan data yang diperlukan oleh aplikasi dari berbagai sumber (misalnya, *database, API,* atau *local storage*).

```dart
Future<MySqlConnection> connectToDatabase() async {
  final settings = ConnectionSettings(
    host: '10.0.2.2',
    port: 3306,
    user: 'admin',
    password: 'Anca-3128',
    db: 'watch_data',
  );

  final connection = await MySqlConnection.connect(settings);
  return connection;
}
```

Gambar 5.2.2.4 Contoh Implementasi Repository Pertama

```dart
Future<List<Map<String, dynamic>>> fetchAlarmMarkerData(
    {
      String? status,
      String? site_id,
      String? site_name,
      String? address,
    }) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();

    var query =
    '''
    SELECT
    site_id,
    site_name,
    latitude,
    longitude,
    alamat AS address,
    status
    FROM data_nodeb
    ''';
```

Gambar 5.2.2.5 Contoh Implementasi Repository Pertama

Fungsi ini mengambil data dari tabel data_nodeb berdasarkan parameter filter seperti *site_id* atau *site_name*. Hasil *query* diubah menjadi list of maps (*List<Map<String, dynamic>>)* yang berisi informasi seperti *site_id, site_name, latitude, longitude, address,* dan status. Ini mirip dengan metode *getAllEvent* pada *EventRepository* karena bertugas untuk mengambil kumpulan data dari database dan memasukkannya ke dalam format yang bisa digunakan oleh bagian lain dari aplikasi.

3. **Model**

Model digunakan di dalam data layer untuk merepresentasikan data dari sumber eksternal (seperti *database, API,* atau file). Model biasanya mengimplementasikan atau memperluas *entity* yang ada di domain *layer*.

```
6   class User {
7     final int? id;
8     final String name;
9     final String username;
10    String? userSince;
11    User({
12      this.id,
13      required this.name,
14      required this.username,
15      this.userSince,
16    });
17  }
18
19  class UserController extends GetxController {
20    final Rx<User?> user = Rx<User?> (null);
```

Gambar 5.2.3.6 Contoh Implementasi Model Pertama

```
SharedPreferences prefs = await SharedPreferences.getInstance();
prefs.setString(
  'id',
  userid.toString(),
);
prefs.setString(
  'Name',
  userData['Name'],
);
prefs.setString(
  'Username',
  userData['Username'],
);
String userSinceString = userData['user_since']?.toString()??'';
prefs.setString(
  'user_since',
  userSinceString,
);

user.value = User(
  id: userid,
  name: userData['Name'],
  username: userData['Username'],
  userSince: userSinceString,
);
```

Gambar 5.2.3.7 Contoh Implementasi Model Kedua

Class User di atas memiliki properti (*id, name, username, userSince*) yang digunakan untuk merepresentasikan data pengguna yang diambil dari basis data. Data yang diambil dari *query SQL (results.first)* diubah ke dalam bentuk *instance User*. Class ini dapat digunakan untuk melakukan pemetaan data dari basis data atau sumber eksternal lainnya, yang merupakan salah satu fungsi utama dari sebuah model. User berfungsi sebagai model yang menyimpan informasi pengguna yang diambil dari *database*. Meskipun *class* ini belum memiliki metode *fromJson* atau *toJson*, class ini sudah bertindak sebagai model yang mewakili data pengguna. *Class User* dalam kode diatas berfungsi sebagai model karena digunakan untuk memetakan data yang diambil dari *database* ke dalam struktur data aplikasi.

4. ***DataSource***

DataSource adalah komponen yang digunakan untuk mengambil data dari sumber eksternal seperti database atau API. Dalam konteks *clean architecture*, *DataSource* digunakan di lapisan data untuk mengakses sumber data secara langsung.

```dart
Future<bool> login(String username, String password) async {
  MySqlConnection? conn;
  try{
    conn = await connectToDatabase();
    var results = await conn.query(
      '''
      SELECT * FROM users
      WHERE username = ?
      AND password = ?
      ''',
      [username,password],
    );
    var userData = results.first;
    var userid = userData['id'] as int;
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.setString(
      'id',
      userid.toString(),
    );
    prefs.setString(
      'Name',
      userData['Name'],
    );
    prefs.setString(
      'Username',
      userData['Username'],
    );
    String userSinceString = userData['user_since']?.toString()??'';
    prefs.setString(
      'user_since',
      userSinceString,
    );
```

Gambar 5.2.4.8 Contoh Implementasi *DataSource* Pertama

Gambar 5.2.4.9 Contoh Implementasi *DataSource* Kedua

Fungsi *login* ini berperan sebagai *DataSource* karena mengambil data dari *database*, memprosesnya, dan mengembalikannya ke lapisan lain, yang merupakan fungsi utama dari sebuah *DataSource*. *Login* berperan untuk mengambil data pengguna dari database MySQL berdasarkan username dan password yang diberikan.

5. ***Repository Implementation***

Repository Implementation atau *RepositoryImpl* merupakan salah satu komponen dalam arsitektur perangkat lunak yang bertanggung jawab untuk mengimplementasikan kontrak yang didefinisikan oleh *interface repository*. Pada dasarnya, *repository implementation* mengelola akses data dari berbagai sumber seperti basis data, API, file, atau *cache* lokal, dan menyediakan data tersebut ke *domain layer* atau *business logic layer* dalam format yang konsisten dan mudah digunakan.

```
77    Future<Map<String, dynamic>> fetchAlarmDetails(String siteId) async {
78      MySqlConnection? conn;
79      try {
80        conn = await connectToDatabase();
81
82        var query =
83        '''
84        SELECT
85        type_olt,
86        hostname_olt,
87        ip_address_olt,
88        uplink_olt,
89        port_splitter_olt,
90        type_ont,
91        serial_number_ont,
92        ip_address_ont,
93        port_ont,
94        odp_name,
95        odp_longitude,
96        odp_latitude,
97        odp_address
98        FROM data_nodeb
99        WHERE site_id = ?
100       ''';
101
102       final results = await conn.query(query, [siteId]);
103       if(results.isNotEmpty) {
104         final row = results.first;
```

Gambar 5.2.5.10 Contoh Implementasi *Repository Implementation*

Fungsi ini mengambil detail alarm dari tabel data_nodeb berdasarkan siteId. Fungsi ini menambahkan kondisi *WHERE* berdasarkan parameter yang diterima (site_id dan site_name) dan mengembalikan hasil query dalam bentuk *List<Map<String, dynamic>>,* yang berisi informasi seperti site_id, site_name, latitude, longitude, address, dan status. Ini mirip dengan implementasi repository, di mana fungsi ini bertanggung jawab untuk mengelola data dari sumber eksternal dan mengembalikan hasil yang sudah diproses. Fungsi ini memproses hasil *query* menjadi *Map<String, dynamic>* yang berisi detail lengkap dari *site_id* yang dicari. *fetchAlarmDetails* mengelola data dari *database*, memprosesnya, dan mengembalikan hasilnya ke lapisan lain.

6. ***Controller***

Controller pada *GetX* adalah komponen inti yang digunakan untuk mengelola logika bisnis serta *state* dalam aplikasi *Flutter*. *Controller* di *GetX* lebih sederhana dan langsung menghubungkan logika bisnis dengan UI. Fungsi utama *Controller* dalam *GetX* adalah, mengelola *state*, memisahkan logika bisnis dari UI, serta *Dependency Injection*.

```dart
class UserController extends GetxController {
  final Rx<User?> user = Rx<User?> (null);

  Future <MySqlConnection> connectToDatabase() async {
    final setting = ConnectionSettings(
      host: '10.0.2.2',
      port: 3306,
      user: 'admin',
      password: 'Anca-3128',
      db: 'watch_data',
    );

    await Future.delayed(const Duration(milliseconds: 100));
    final connection = await MySqlConnection.connect(setting);
    return connection;
  }
```

Gambar 5.2.6.11 Contoh Implementasi *Controller* Pertama

```dart
  Future<bool> login(String username, String password) async {
    MySqlConnection? conn;
    try{
      conn = await connectToDatabase();
      var results = await conn.query(
        '''
        SELECT * FROM users
        WHERE username = ?
        AND password = ?
        ''',
        [username,password],
      );
      var userData = results.first;
      var userid = userData['id'] as int;
      SharedPreferences prefs = await SharedPreferences.getInstance();
```

Gambar 5.2.6.12 Contoh Implementasi *Controller* Kedua

Gambar 5.2.6.13 Contoh Implementasi *Controller* Ketiga

Bagian kode yang menggunakan *GetX Controller* adalah *UserController*, di mana *class* ini bertanggung jawab atas logika bisnis terkait *user* seperti login dan penyimpanan data. *Controller* ini memanfaatkan Rx untuk mengelola state secara reaktif, dan perubahan data dalam *controller* secara otomatis mengupdate UI yang terkait.

7. ***Pages***

Pages merujuk pada tampilan antarmuka pengguna yang di *render* pada layar sebagai halaman. Dalam *Flutter*, halaman ini dibuat dengan menggunakan *StatelessWidget* atau *StatefulWidget*. *Pages* bertanggung jawab untuk menampilkan informasi, mengelola interaksi pengguna, dan menavigasi ke halaman lain di aplikasi.

```dart
import 'package:flutter/material.dart';

class MapScreen extends StatefulWidget {
  const MapScreen({super.key});

  @override
  State<MapScreen> createState() => _MapScreenState();
}

class _MapScreenState extends State<MapScreen> {
  List<Marker> markers = [];
  late MapController mapController;

  @override
  void initState() {
    super.initState();
    mapController = MapController();
    _loadMarkers();
  }
```

Gambar 5.2.7.14 Contoh Implementasi *Pages* Pertama

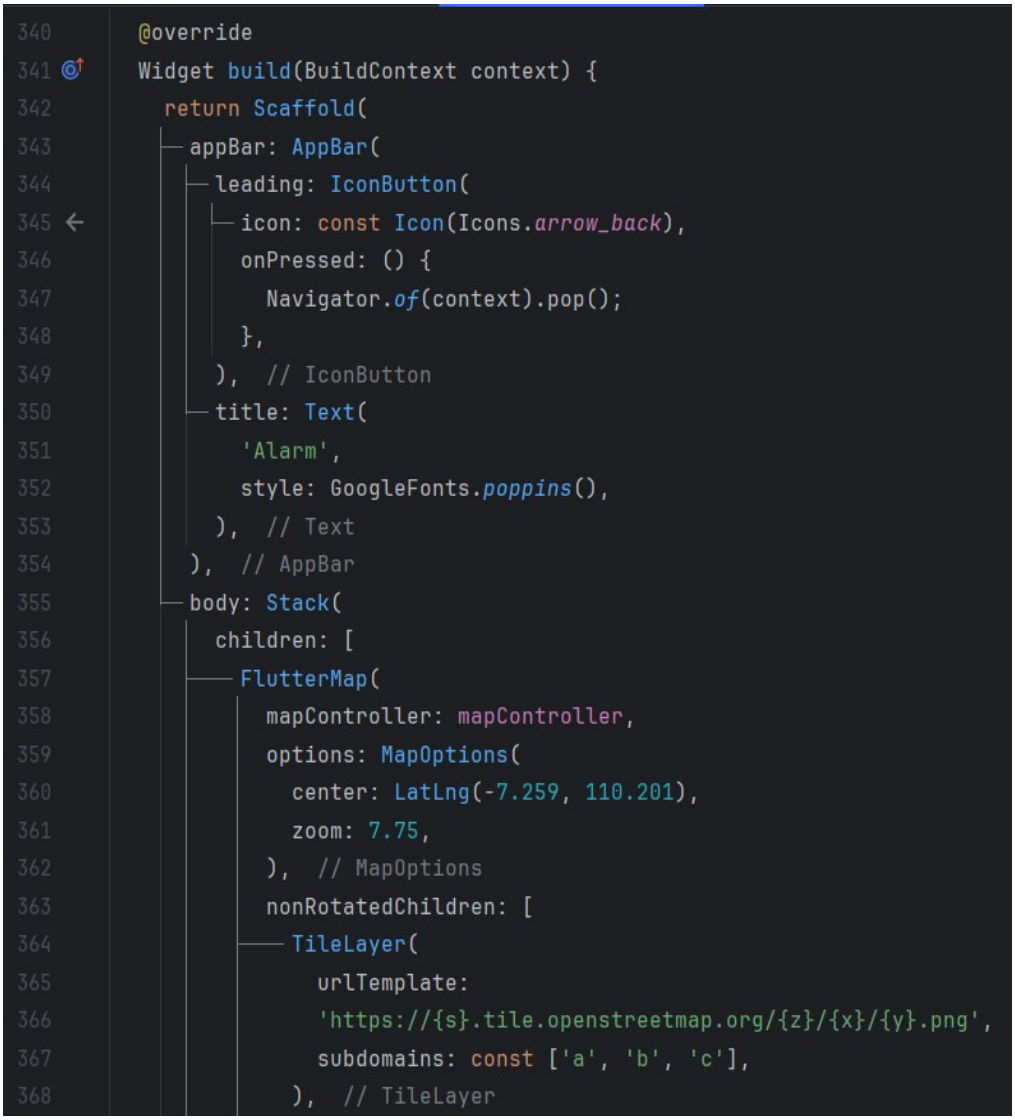
Gambar 5.2.7.15 Contoh Implementasi *Pages* Kedua

Di dalam kode ini, *MapScreen* didefinisikan sebagai *StatefulWidget*, yang berarti halaman tersebut dapat merespons perubahan state tanpa perlu memuat ulang seluruh halaman. Karena peta dan marker dalam aplikasi ini dinamis, *StatefulWidget* cocok digunakan di sini. *MapScreen* adalah sebuah *Page* yang ditulis menggunakan *StatefulWidget*. Ini adalah halaman antarmuka pengguna yang dibuat dengan mengkombinasikan berbagai *widget Flutter*, seperti *Scaffold*, *AppBar*, dan *FlutterMap*. State dikelola di dalam *class* *_MapScreenState*, dan halaman ini dapat berubah secara dinamis berdasarkan data yang diambil dari API. Dengan menggunakan struktur ini, *Flutter* memungkinkan untuk membuat antarmuka pengguna yang fleksibel, responsif, dan interaktif.

8. ***Widget***

Widget adalah elemen dasar untuk membangun antarmuka pengguna (UI). Sebuah *widget* merepresentasikan tampilan atau bagian dari aplikasi, seperti teks, gambar, tombol, atau komponen yang lebih kompleks. *Flutter* menggunakan pendekatan berbasis *widget*, dimana segala elemen UI, bahkan *layout* dan perilaku, dibangun

menggunakan *widget*.



```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      leading: IconButton(
        icon: const Icon(Icons.arrow_back),
        onPressed: () {
          Navigator.of(context).pop();
        },
      ), // IconButton
      title: Text(
        'Alarm',
        style: GoogleFonts.poppins(),
      ), // Text
    ), // AppBar
    body: Stack(
      children: [
      FlutterMap(
        mapController: mapController,
        options: MapOptions(
          center: LatLng(-7.259, 110.201),
          zoom: 7.75,
        ), // MapOptions
        nonRotatedChildren: [
        TileLayer(
            urlTemplate:
            'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
            subdomains: const ['a', 'b', 'c'],
          ), // TileLayer
```

Gambar 5.2.8.16 Contoh Implementasi *Widget*

9. ***Routes***

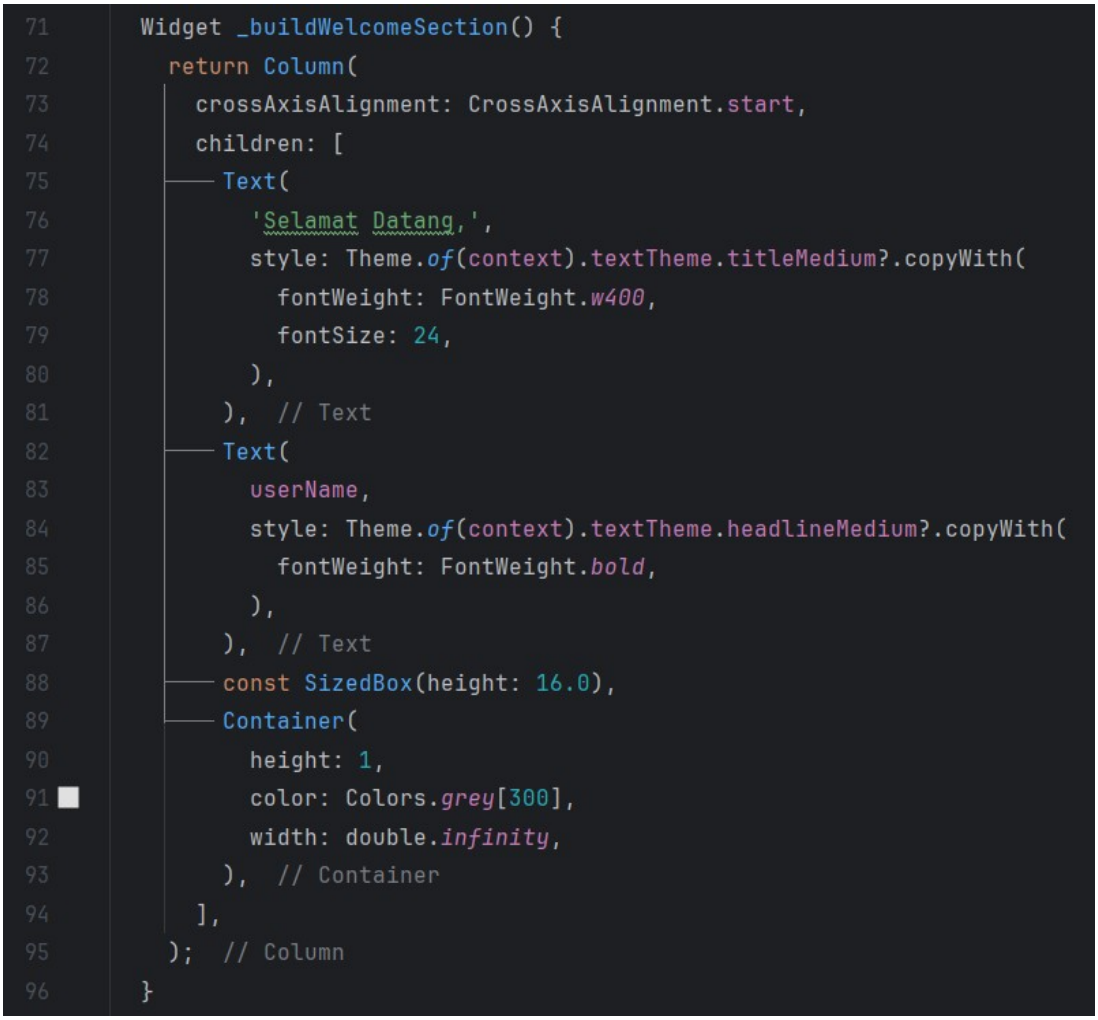
Routes dalam *Flutter* adalah sistem navigasi yang memungkinkan aplikasi untuk berpindah antar layar atau halaman. *Flutter* menyediakan sistem *routing* untuk mendefinisikan, mengelola, dan mengontrol navigasi dalam aplikasi. *Routing* memainkan peran penting dalam pengembangan aplikasi dengan arsitektur *Clean Architecture*, karena dapat memisahkan tampilan antarmuka dengan logika bisnis.

```
Container(
  width: double.infinity,
  margin: const EdgeInsets.only(top: 30),
  alignment: Alignment.center,
  child: ElevatedButton(
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => StatusPage(
            site_id: siteId,
            site_name: siteName,
          ),  // StatusPage
        ),  // MaterialPageRoute
      );
    },
```

Gambar 5.2.9.17 Contoh Implementasi *Routes*

10. ***Theme***

Theme dalam *Flutter* adalah sistem yang memungkinkan pengembang mengatur gaya dan tampilan konsisten di seluruh aplikasi. *ThemeData* memungkinkan penggunaan kembali *style* yang sama untuk berbagai elemen, sehingga mengurangi pengaturan yang berulang-ulang. Ini adalah bagian penting dalam Arsitektur Clean karena memungkinkan pemisahan antara logika dan tampilan, serta membantu menciptakan konsistensi antarmuka pengguna di seluruh aplikasi.



```
Widget _buildWelcomeSection() {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        'Selamat Datang,',
        style: Theme.of(context).textTheme.titleMedium?.copyWith(
          fontWeight: FontWeight.w400,
          fontSize: 24,
        ),
      ),  // Text
      Text(
        userName,
        style: Theme.of(context).textTheme.headlineMedium?.copyWith(
          fontWeight: FontWeight.bold,
        ),
      ),  // Text
      const SizedBox(height: 16.0),
      Container(
        height: 1,
        color: Colors.grey[300],
        width: double.infinity,
      ),  // Container
    ],
  );  // Column
}
```

Gambar 5.2.10.18 Contoh Implementasi *Theme*

11. ***Module***

Module adalah konsep untuk mengorganisir kode berdasarkan fungsionalitas atau tanggung jawab, yang mendukung *Clean Architecture* dan modularisasi. Modularisasi membantu memecah aplikasi menjadi bagian-bagian yang lebih kecil, terpisah, dan independen. Masing-masing *module* bertanggung jawab atas satu area fungsional tertentu, seperti manajemen data, UI, layanan jaringan, atau *dependency injection*.

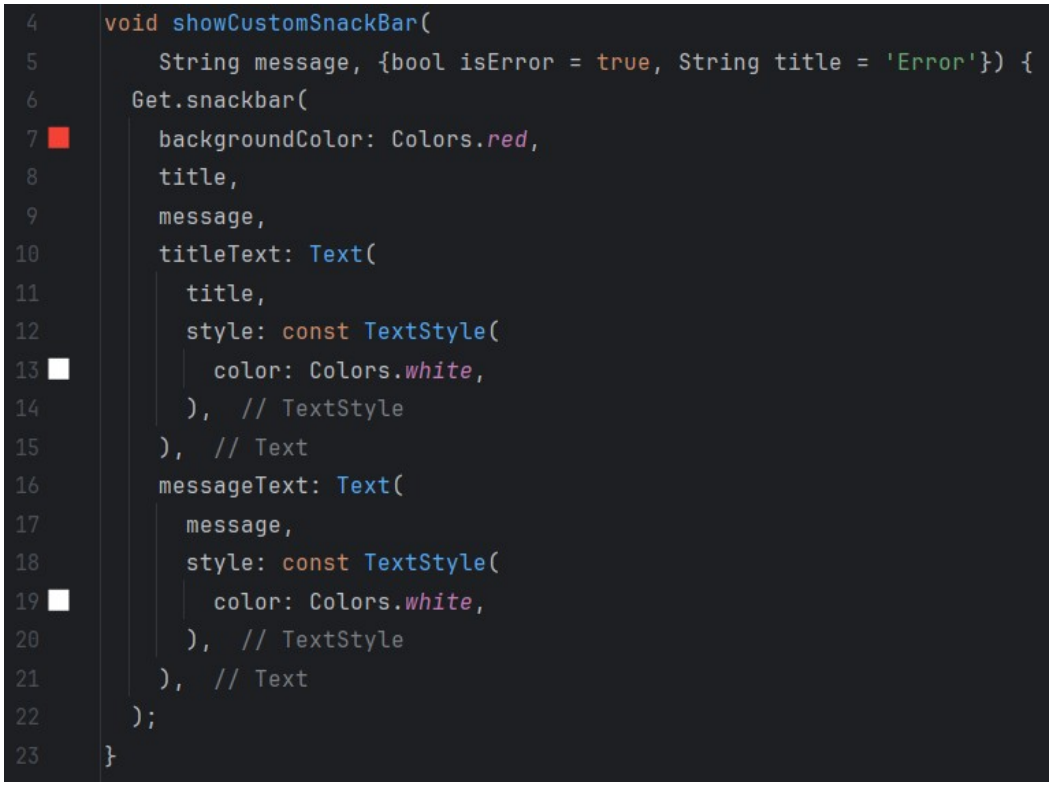
```dart
@override
void initState() {
  super.initState();
  _loadUserName();
}

_loadUserName() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  setState(() {
    userName = prefs.getString('Name') ?? '';
  });
}
_logout() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  await prefs.clear();
  Navigator.of(context).pushReplacement(
    MaterialPageRoute(
      builder: (context) => const LogIn(),
    ),  // MaterialPageRoute
  );
}
```

Gambar 5.2.11.19 Contoh Implementasi *Module*

12. ***Utils***

Utils (*Utilities*) adalah sekumpulan fungsi atau kelas yang bersifat umum dan tidak tergantung pada logika bisnis tertentu. *Utils* dirancang untuk mengurangi pengulangan kode dengan menyediakan fungsionalitas tambahan yang bisa digunakan kembali di berbagai modul atau komponen aplikasi.



```dart
void showCustomSnackBar(
    String message, {bool isError = true, String title = 'Error'}) {
  Get.snackbar(
    backgroundColor: Colors.red,
    title,
    message,
    titleText: Text(
      title,
      style: const TextStyle(
        color: Colors.white,
      ),  // TextStyle
    ),  // Text
    messageText: Text(
      message,
      style: const TextStyle(
        color: Colors.white,
      ),  // TextStyle
    ),  // Text
  );
}
```

Gambar 5.2.12.20 Contoh Implementasi *Utils*

13. ***Main***

Main adalah *file entry point* atau titik masuk utama aplikasi, yang pertama kali dijalankan ketika aplikasi dimulai. *Main* memuat konfigurasi awal aplikasi, inisialisasi widget utama, serta pengaturan dasar yang mengontrol perilaku dan tampilan aplikasi secara keseluruhan.

```dart
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const GetMaterialApp(
      debugShowCheckedModeBanner: false,
      home: LogIn(),
    );  // GetMaterialApp
  }
}
```

Gambar 5.2.13.21 Contoh Implementasi *Main*

5.2.2 Implementasi Kueri

Pengimplementasian kueri pada aplikasi yang dibuat adalah untuk mengambil data yang dibutuhkan berdasarkan parameter yang ditentukan oleh pengguna. Dalam proses pembuatan aplikasi, kami mengimplementasikan kueri di beberapa kebutuhan pengolahan data seperti halnya berikut :

1. *Fetch* Alarm Data

```
Future<List<Map<String, dynamic>>> fetchAlarmMarkerData(
    {
      String? status,
      String? site_id,
      String? site_name,
      String? address,
    }) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();

    var query =
    '''
    SELECT
    site_id,
    site_name,
    latitude,
    longitude,
    alamat AS address,
    status
    FROM data_nodeb
    ''';
```

Gambar 5.2.2.1 *Fetch* Alarm Data

Kueri SQL pada kode tersebut digunakan untuk mengambil data dari tabel **`data_nodeb`**, yang berisi informasi terkait lokasi dan status tertentu. Kueri ini memilih kolom-kolom seperti **`site_id`**, **`site_name`**, **`latitude`**, **`longitude`**, **`alamat`** (diubah menjadi **`address`**), dan **`status`**. Fungsi ini bertujuan untuk mengambil data lokasi dan status alarm untuk kemudian digunakan dalam sistem monitoring, di mana data tersebut diolah untuk ditampilkan dalam format alarm seperti penanda pada peta.

2. *Fetch* Alarm Detail Data

```
Future<Map<String, dynamic>> fetchAlarmDetails(String site_id) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();
    var query =
    '''
    SELECT
    type_olt,
    hostname_olt,
    ip_address_olt,
    uplink_olt,
    port_splitter_olt,
    type_ont,
    serial_number_ont,
    ip_address_ont,
    port_ont,
    odp_name,
    odp_longitude,
    odp_latitude,
    odp_address
    FROM data_nodeb
    WHERE site_id = ?
    ''';
```

Gambar 5.2.2.2 *Fetch* Alarm Detail Data

Kueri SQL pada kode tersebut digunakan untuk mengambil detail alarm dari tabel `**data_nodeb**` berdasarkan `**site_id**` yang diberikan sebagai parameter. Kueri ini memilih berbagai informasi teknis yang berkaitan dengan OLT (Optical Line Terminal) dan ONT (*Optical Network Terminal*), seperti tipe OLT (`**type_olt**`), hostname OLT, alamat IP OLT, uplink OLT, port splitter OLT, serta informasi tentang ONT seperti tipe ONT, nomor seri, alamat IP, port, dan detail ODP (*Optical Distribution Point*) seperti nama, koordinat (longitude dan latitude), dan alamat. Kueri ini memfilter data berdasarkan nilai `**site_id**` yang dipilih oleh user.

3. *Fetch* Olo Data

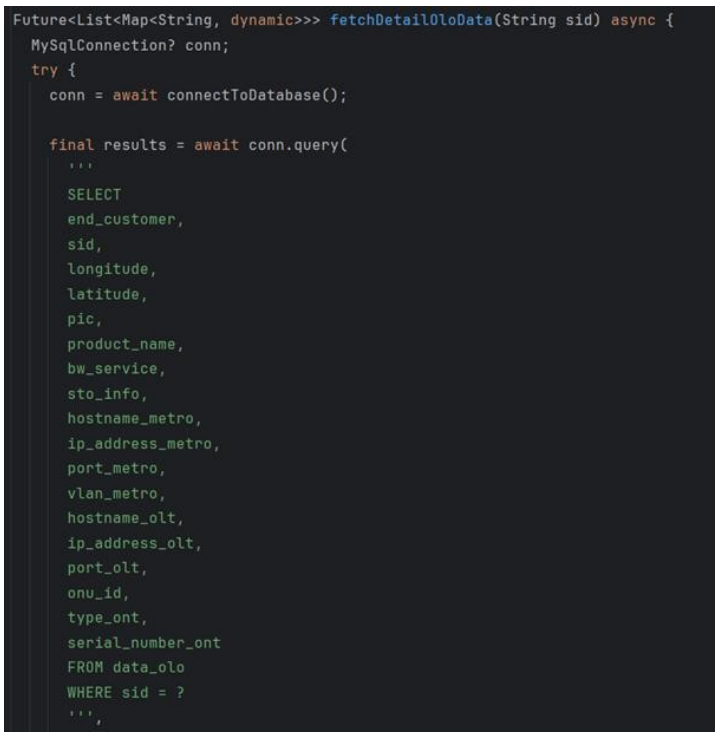
```
Future<List<Map<String, dynamic>>> fetchOloMarkerData({
  String? customer_name,
  String? end_customer,
  String? witel,
  String? sid,
}) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();

    var query =
    '''
    SELECT
    customer_name,
    end_customer,
    witel,
    sid
    FROM data_olo
    ''';
```

Gambar 5.2.2.3 *Fetch* Olo Data

Kueri SQL pada kode tersebut digunakan untuk mengambil data terkait pelanggan dari tabel **`data_olo`**. Kueri ini memilih kolom-kolom seperti **`customer_name`**, **`end_customer`**, **`witel`**, dan **`sid`**. Informasi ini berkaitan dengan detail pelanggan dan wilayah telekomunikasi (witel) untuk keperluan pemantauan, khususnya dalam konteks OLO (*Other Licensed Operator*). Fungsi ini memungkinkan pengambilan informasi pelanggan yang relevan dari database, yang kemudian dapat digunakan untuk menampilkan informasi pelanggan OLO.

4. *Fetch* Olo Detail Data



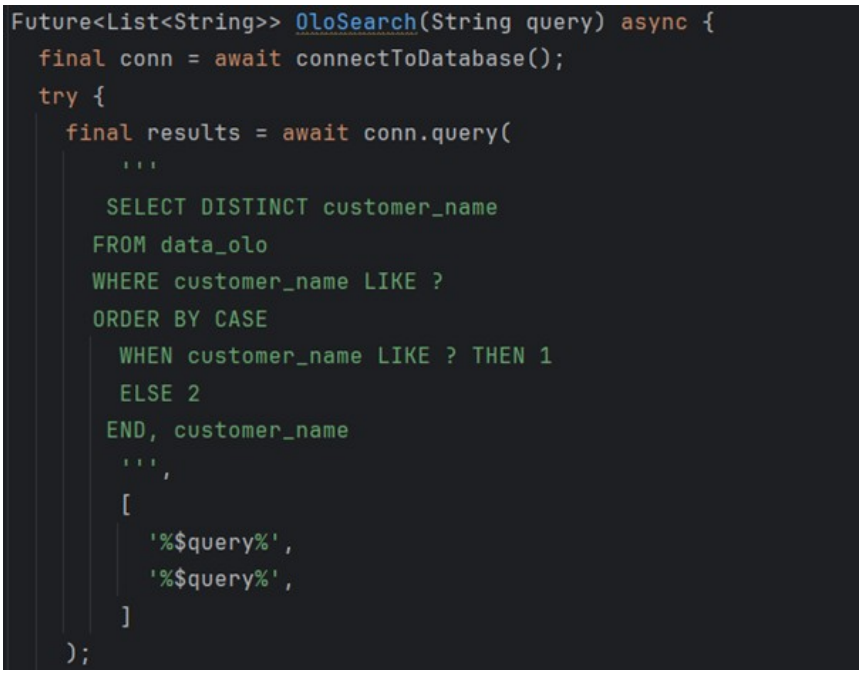
```
Future<List<Map<String, dynamic>>> fetchDetailOloData(String sid) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();

    final results = await conn.query(
      '''
      SELECT
      end_customer,
      sid,
      longitude,
      latitude,
      pic,
      product_name,
      bw_service,
      sto_info,
      hostname_metro,
      ip_address_metro,
      port_metro,
      vlan_metro,
      hostname_olt,
      ip_address_olt,
      port_olt,
      onu_id,
      type_ont,
      serial_number_ont
      FROM data_olo
      WHERE sid = ?
      ''',
```

Gambar 5.2.2.4 *Fetch* Olo Detail Data

Kueri SQL pada kode tersebut digunakan untuk mengambil detail data OLO (*Other Licensed Operator*) dari tabel **`data_olo`** berdasarkan **`sid`** tertentu yang diberikan sebagai parameter. Kueri ini memilih berbagai kolom yang berkaitan dengan informasi pelanggan dan perangkat jaringan, seperti **`end_customer`**, **`sid`**, koordinat lokasi (**`longitude`** dan **`latitude`**), nama produk (**`product_name`**), layanan bandwidth (**`bw_service`**), serta detail teknis terkait perangkat Metro dan OLT, seperti hostname, alamat IP, port, dan informasi VLAN. Selain itu, terdapat informasi terkait ONT (*Optical Network Terminal*), seperti tipe ONT dan nomor serinya. Kueri ini sangat komprehensif dan digunakan untuk mendapatkan informasi lengkap terkait jaringan dan layanan pelanggan berdasarkan **`sid`** pelanggan OLO.

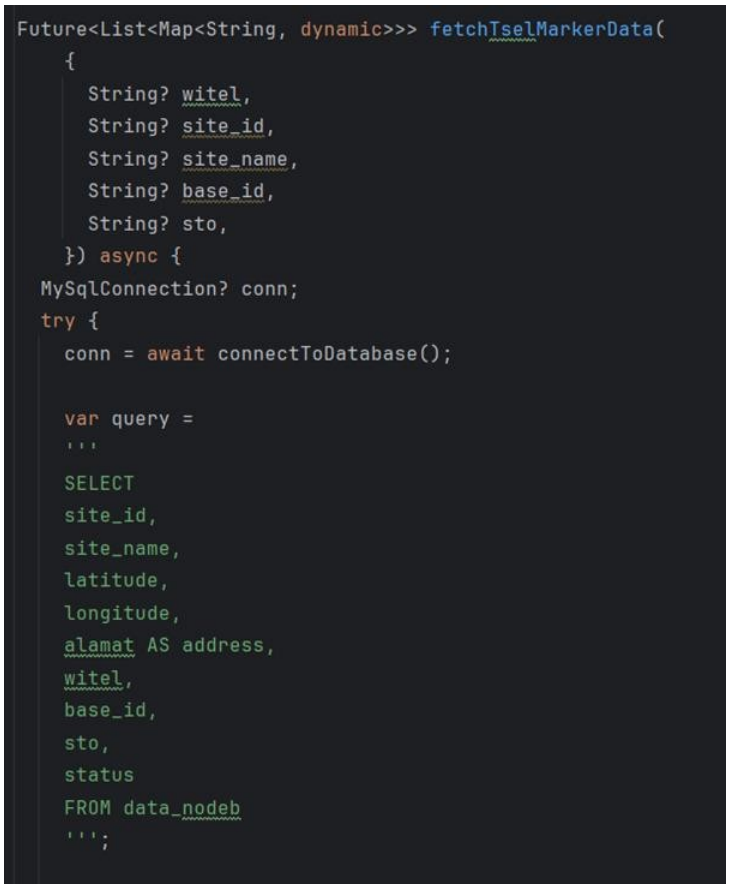
5. *Fetch* Search Olo



Gambar 5.2.2.5 *Fetch* Olo Detail Data

Kueri SQL pada kode tersebut digunakan untuk mencari nama pelanggan dari tabel `**data_olo**` berdasarkan kata kunci yang diberikan dalam parameter `**query**`. Kueri ini memilih nama pelanggan (`**customer_name**`) yang unik menggunakan `**SELECT DISTINCT**` dan memfilter hasilnya berdasarkan kecocokan dengan kata kunci yang dimasukkan menggunakan operator `**LIKE**` (menggunakan wildcard `%` untuk pencarian yang lebih fleksibel). Hasil pencarian diurutkan dengan `**ORDER BY**` menggunakan pernyataan `**CASE**`, yang memberikan prioritas lebih tinggi pada hasil yang lebih relevan (dengan mengatur nilai urutan 1 jika nama pelanggan cocok dengan kata kunci secara langsung, dan 2 jika tidak). Kueri ini digunakan dalam konteks pencarian nama pelanggan yang serupa dengan input dari pengguna, untuk menampilkan hasil yang diurutkan berdasarkan relevansi.

6. *Fetch* Tsel Data



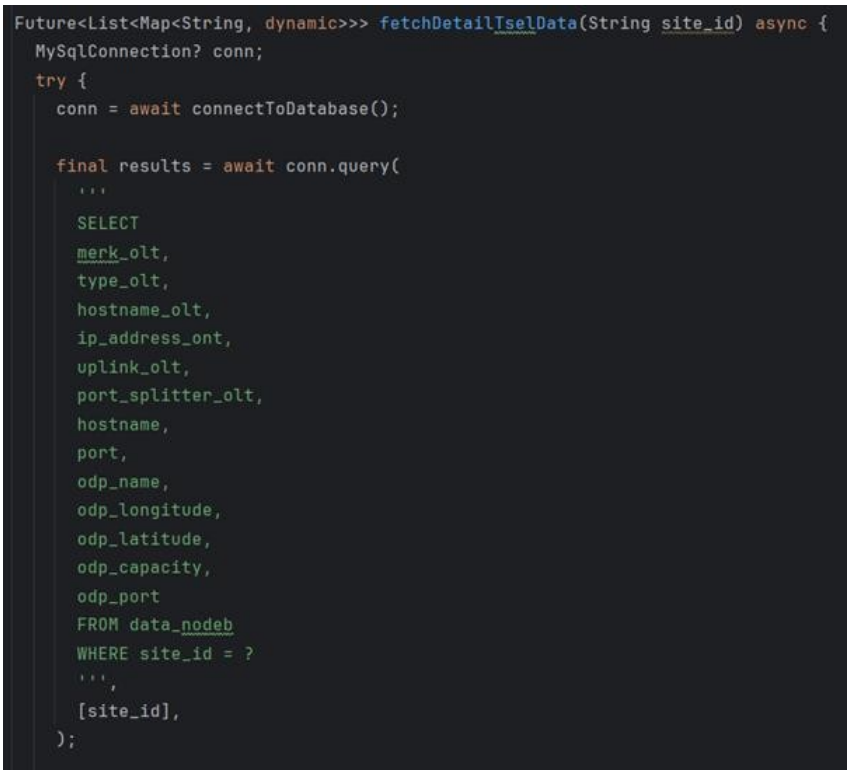
```
Future<List<Map<String, dynamic>>> fetchTselMarkerData(
    {
      String? witel,
      String? site_id,
      String? site_name,
      String? base_id,
      String? sto,
    }) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();

    var query =
    '''
    SELECT
    site_id,
    site_name,
    latitude,
    longitude,
    alamat AS address,
    witel,
    base_id,
    sto,
    status
    FROM data_nodeb
    ''';
```

Gambar 5.2.2.6 *Fetch* Tsel Data

Kueri SQL pada kode tersebut digunakan untuk mengambil data Tsel dari tabel `data_nodeb`, yang berisi informasi lokasi dan status dari berbagai site. Kueri ini memilih kolom-kolom seperti `site_id`, `site_name`, koordinat (`latitude` dan `longitude`), alamat (yang diambil dari kolom `alamat` dan diberi alias `address`), `witel`, `base_id`, `sto`, dan `status`. Data ini digunakan untuk menampilkan atau memantau lokasi-lokasi site pada peta berdasarkan atribut-atribut tersebut.Fungsi ini berperan untuk mengambil data terkait yang sesuai dengan atribut yang diminta dan menampilkannya dalam format yang mudah diakses oleh sistem atau aplikasi terkait.

7. *Fetch* Tsel Detail Data



```
Future<List<Map<String, dynamic>>> fetchDetailTselData(String site_id) async {
  MySqlConnection? conn;
  try {
    conn = await connectToDatabase();

    final results = await conn.query(
      '''
      SELECT
      merk_olt,
      type_olt,
      hostname_olt,
      ip_address_ont,
      uplink_olt,
      port_splitter_olt,
      hostname,
      port,
      odp_name,
      odp_longitude,
      odp_latitude,
      odp_capacity,
      odp_port
      FROM data_nodeb
      WHERE site_id = ?
      ''',
      [site_id],
    );
```

Gambar 5.2.2.7 *Fetch* Tsel Detail Data

Kueri SQL pada kode tersebut digunakan untuk mengambil detail data dari tabel `data_nodeb` berdasarkan nilai `site_id` yang diberikan. Kueri ini memilih informasi teknis terkait OLT (*Optical Line Terminal*) dan ODP (Optical Distribution Point), seperti merk OLT (`merk_olt`), tipe OLT (`type_olt`), hostname OLT, alamat *IP ONT, uplink OLT, port splitter OLT*, serta informasi mengenai ODP, seperti nama, koordinat lokasi (*longitude dan latitude*), kapasitas ODP, dan port ODP. Kueri ini memfilter hasil berdasarkan `site_id`.Data yang diambil digunakan untuk keperluan pemantauan dan pengelolaan infrastruktur jaringan optik, seperti troubleshooting atau optimasi jaringan.

8. *Fetch* Search Tsel

```
Future<List<String>> TselSearch(String query) async {
  final conn = await connectToDatabase();
  try {
    final results = await conn.query(
      '''
      SELECT DISTINCT witel
      FROM data_nodeb
      WHERE witel LIKE ?
      ORDER BY CASE
        WHEN witel LIKE ? THEN 1
        ELSE 2
       END, witel
       ''',
      [
        '%$query%',
        '%$query%',
      ],
    );
```

Gambar 5.2.2.8 *Fetch* Search Tsel

Kueri SQL pada kode tersebut digunakan untuk mencari data `witel` secara unik dari tabel `data_nodeb` berdasarkan kata kunci yang diberikan dalam parameter `*query*`. Kueri ini menggunakan *`SELECT DISTINCT`* untuk memastikan bahwa hanya nilai `witel` yang unik yang diambil, kemudian difilter menggunakan operator `*LIKE*` untuk mencocokkan hasil dengan pola pencarian. Hasil pencarian diurutkan berdasarkan relevansi dengan menggunakan *`ORDER BY CASE`,* di mana `witel` yang lebih cocok dengan *query* ditempatkan di posisi teratas (dengan nilai urutan 1), dan hasil lain di posisi berikutnya (dengan nilai urutan 2). Kueri ini berguna untuk melakukan pencarian data `witel` yang relevan dan menampilkan hasil yang diurutkan berdasarkan tingkat kecocokan dengan input pengguna.

9. *User Login*

```
Future<bool> login(String username, String password) async {
  MySqlConnection? conn;
  try{
    conn = await connectToDatabase();
    var results = await conn.query(
      '''
      SELECT * FROM users
      WHERE username = ?
      AND password = ?
      ''',
      [username,password],
    );
```

Gambar 5.2.2.9 *User Login*

Kueri SQL pada kode tersebut digunakan untuk memverifikasi kredensial login pengguna dengan mencocokkan `*username*` dan `password` yang dimasukkan dengan data yang ada di tabel `*users*`. Kueri ini memilih semua kolom (`*`) dari tabel `*users*` di mana nilai `*username*` dan `password` cocok dengan nilai yang diberikan. Jika pasangan `*username*` dan `*password*` ditemukan dalam *database*, artinya login berhasil dan fungsi akan mengembalikan nilai `*true*`, sedangkan jika tidak ditemukan, login akan dianggap gagal dan mengembalikan `*false*`. Kueri digunakan dalam proses otentikasi pengguna pada aplikasi untuk memastikan bahwa pengguna yang mencoba login memiliki kredensial yang valid.

5.3. Tampilan Antarmuka

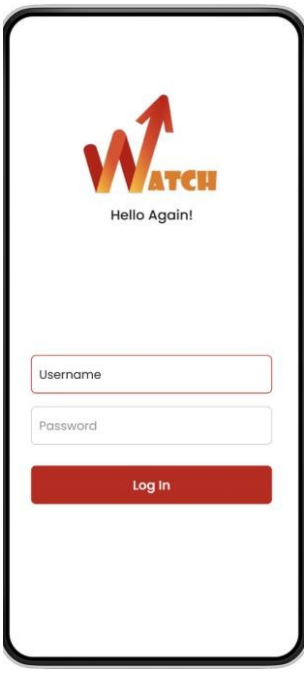
Berdasarkan implementasi yang telah diterapkan dalam *front-end*, maka didapatkan hasil akhir berupa sebuah aplikasi mobile dengan tampilan akhir sebagai berikut. Hal ini dapat dilihat pada Gambar 5.3.1 hingga

1. Tampilan Antarmuka *Splash Screen*

Gambar 5.3.1 Tampilan Antarmuka *Splash Screen*

Tampilan antarmuka *splash screen* saat membuka aplikasi *mobile watch* yang telah dikembangkan.

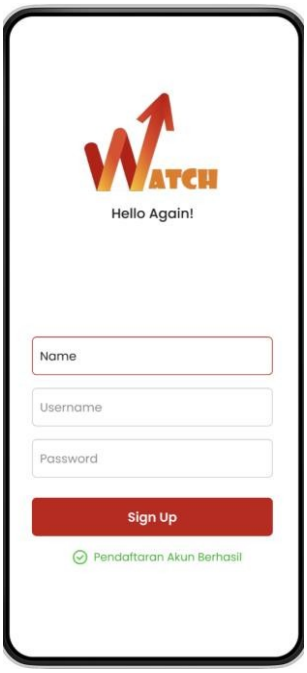
2. Tampilan Antarmuka Halaman *Login*



Gambar 5.3.2 Tampilan Antarmuka Halaman *Login*

Tampilan halaman *login* untuk aplikasi *mobile watch* yang mempunyai 2 *roles,* yakni *user* melakukan input *username* dan *password.*

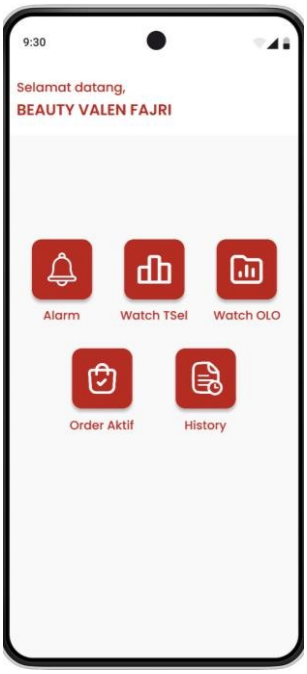
3. Tampilan Antarmuka Halaman *Register*



Gambar 5.3.3 Tampilan Antarmuka Halaman *Register*

Tampilan halaman *register* untuk aplikasi *mobile watch* yang mempunyai 3 *roles,* yakni user melakukan input *name, username* dan*password.*

4. Tampilan Antarmuka *Home*



Gambar 5.3.4 Tampilan Antarmuka *Home*

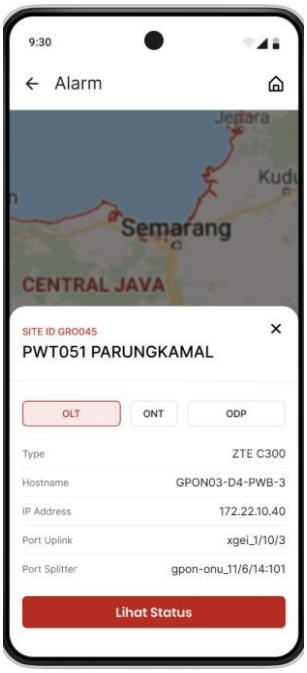
Tampilan *home* untuk aplikasi *mobile watch* yangmempunyai 5 *button,* yakni Alarm, Watch TSEL, Watch OLO, Order Aktif, History.

5. Tampilan Antarmuka Halaman Alarm

Gambar 5.3.5 Tampilan Antarmuka Halaman Alarm

Tampilan halaman alarm untuk aplikasi *mobile watch* yang ditandaisimbol berwarna merah untuk wilayah yang sedang mengalami *trouble*.

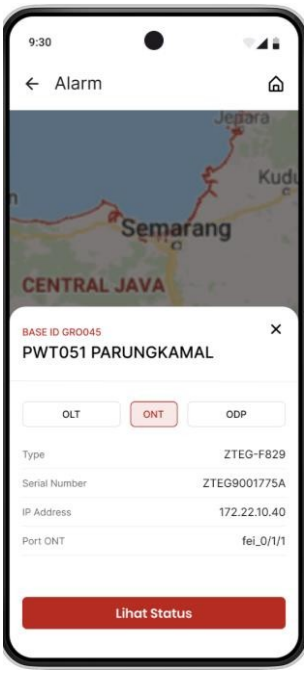
6. Tampilan Antarmuka Halaman Detail Alarm



Gambar 5.3.6 Tampilan Antarmuka Halaman Detail Alarm

Tampilan halaman detail alarm untuk aplikasi *mobile watch* yang mempunyai 3 bagian untuk detail alarm, salah satunya OLT.

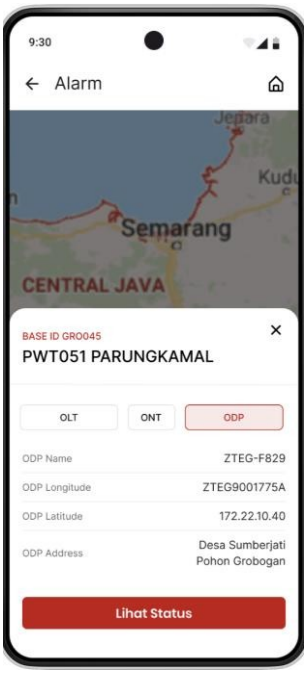
7. Tampilan Antarmuka Halaman Detail Alarm



Gambar 5.3.7 Tampilan Antarmuka Halaman Detail Alarm

Tampilan halaman detail alarm untuk aplikasi *mobile watch* yang mempunyai 3 bagian untuk detail alarm, salah satunya ONT.

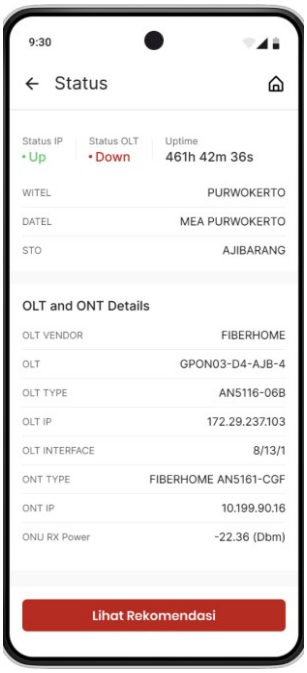
8. Tampilan Antarmuka Halaman Detail Alarm



Gambar 5.3.8 Tampilan Antarmuka Halaman Detail Alarm

Tampilan halaman detail alarm untuk aplikasi *mobile watch* yang mempunyai 3 bagian untuk detail alarm, salah satunya ODP.

9. Tampilan Antarmuka Halaman Status



Gambar 5.3.9 Tampilan Antarmuka Halaman Status

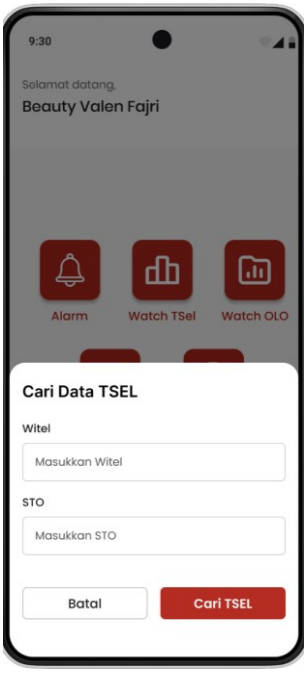
Tampilan halaman status untuk aplikasi *mobile warch.*

10. Tampilan Antarmuka Halaman Rekomendasi

Gambar 5.3.10 Tampilan Antarmuka Halaman Rekomendasi

Tampilan halaman rekomendasi untuk aplikasi *mobile watch*.

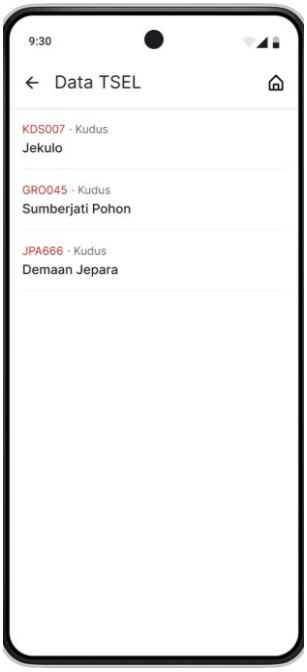
11. Tampilan Antarmuka Halaman *Watch* TSEL



Gambar 5.3.11 Tampilan Antarmuka Halaman *Watch* TSEL

Tampilan halaman Watch TSEL untuk aplikasi *mobile watch* yang menampilkan 2 *search* data TSEL,witel dan STO.

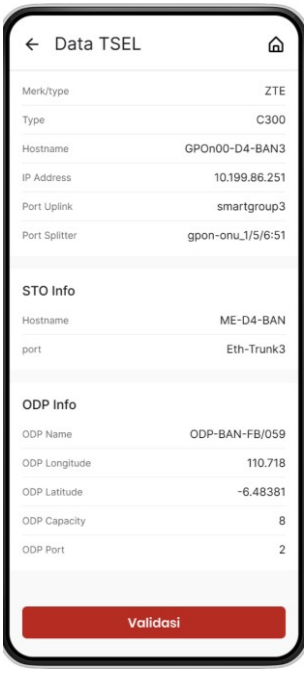
12. Tampilan Antarmuka Halaman Data *Watch* TSEL



Gambar 5.3.12 Tampilan Antarmuka Halaman Data *Watch* TSEL

Tampilan halaman data watch TSEL untuk aplikasi *mobile watch*.

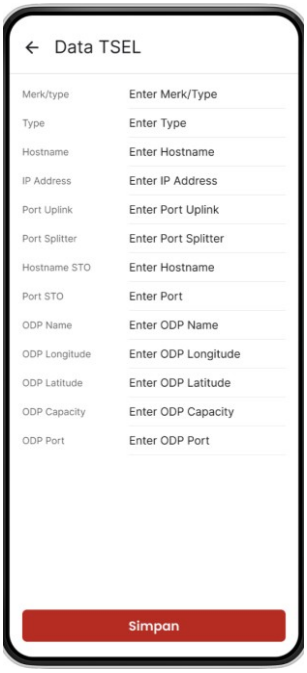
13. Tampilan Antarmuka Halaman Detail Data *Watch* TSEL



Gambar 5.3.13 Tampilan Antarmuka Halaman Detail Data Watch TSEL

Tampilam halaman detail data *watch* TSEL untukaplikas *mobile watch.*

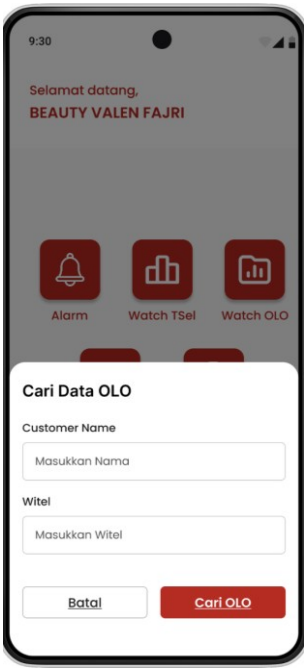
14. Tampilan Antarmuka Halaman Validasi *Watch* TSEL



Gambar 5.3.14 Tampilan Antarmuka Halaman Validasi *Watch* TSEL

Tampilan halaman validasi *watch* TSEL untuk aplikasi *mobile watch* yang didalamnya bisa melakukan edit data mengenai data apa saja yang mengalami perubahan.

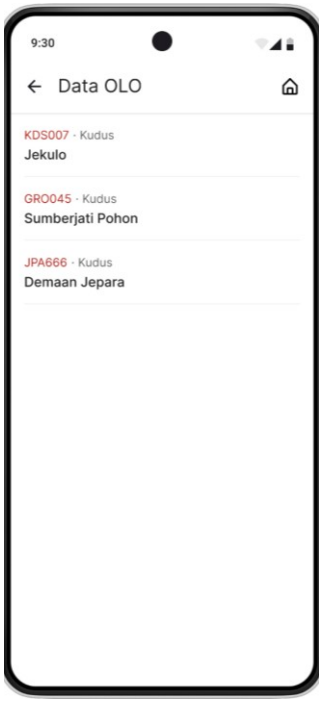
15. Tampilan Antarmuka Halaman *Watch* OLO



Gambar 5.3.15 Tampilan Antarmuka Halaman *Watch* OLO

Tampilan halaman *watch* OLO untuk aplikasi *mobile watch* yang menampilkan 2 *search* data OLO, yakni mengenai customername dan witel.

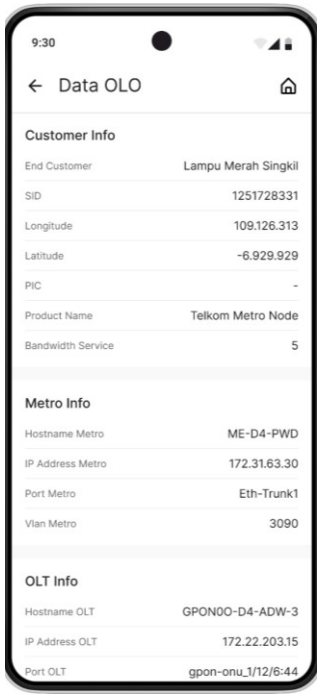
16. Tampilan Antarmuka Halaman Data *Watch* OLO

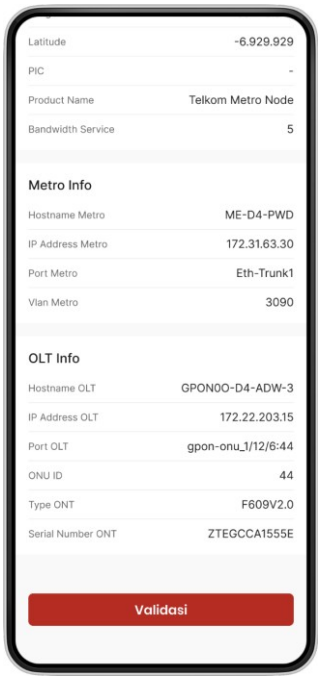


Gambar 5.3.16 Tampilan Antarmuka Halaman Data *Watch* OLO

Tampilan antarmuka halaman halaman data *watch* OLO untuk aplikasi *mobile watch*.

17. Tampilan Antarmuka Halaman Detail Data *Watch* OLO

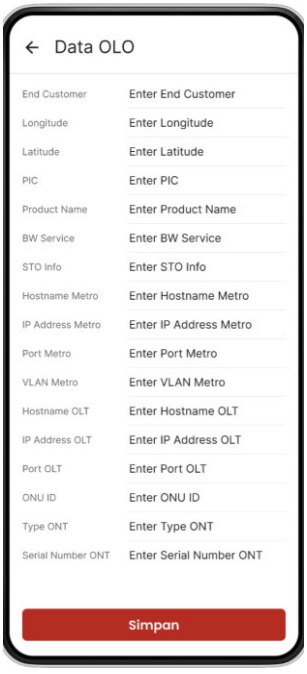




Gambar 5.3.17 Tampilan Antarmuka Halaman Detail Data *Watch* OLO

Tampilan antarmuka halaman detail data *watch* OLO untuk aplikasi m*obile watch*.

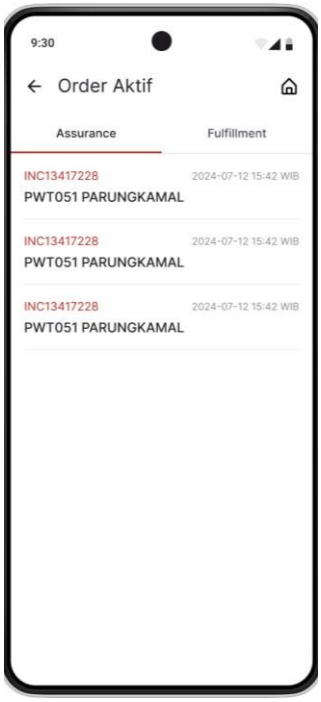
18. Tampilan Antarmuka Halaman Validasi *Watch* OLO



Gambar 5.3.18 Tampilan Antarmuka Halaman Validasi *Watch* OLO

Tampilan halaman validasi *watch* OLO untuk aplikasi *mobile watch* yang didalamnya bisa melakukan edit data mengenai data apa saja yang mengalami perubahan.

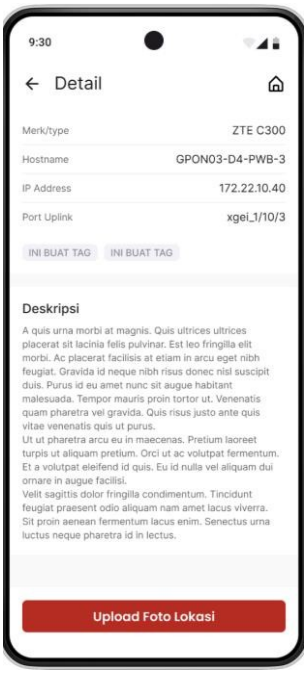
19. Tampilan Antarmuka Halaman Order Aktif *Assurance*



Gambar 5.3.19 Tampilan Antarmuka Halaman Order Aktif *Assurance*

Tampilan antarmuka halaman order aktif *assurance* untuk aplikasi *Mobile Watch* yang memuat tentang data-data yangmasih memiliki status aktif atau terkendala.

20. Tampilan Antarmuka Halaman Detail Order Aktif *Assurance*



Gambar 5.3.20 Tampilan Antarmuka Halaman Detail Order Aktif *Assurance*

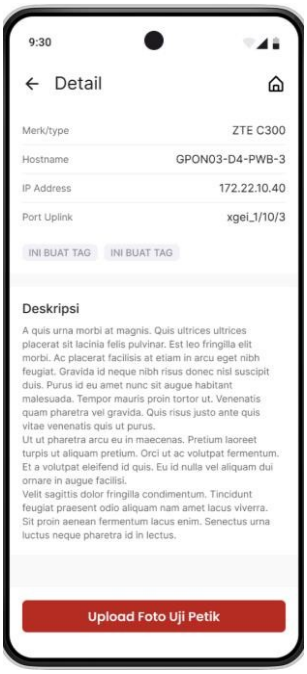
Tampilan halaman detail order aktif *assurance* untukaplikasi *mobile watch.*

21. Tampilan Antarmuka Halaman Order Aktif *Fulfillment*

Gambar 5.3.21 Tampilan Antarmuka Halaman Order Aktif *Fulfillment*

Tampilan halaman order aktif *fulfillment* untuk aplikasi *mobile watch* yang memuat tentang data-data yang masih memiliki statusaktif atau terkendala.

22. Tampilan Antarmuka Halaman Detail Order Aktif *Fulfillment*



Gambar 5.3.22 Tampilan Antarmuka Halaman Detail Order Aktif *Fulfillment*

Tampilan halaman order aktif *fulfillment* untukaplikasi *mobile watch.*

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba dilakukan terhadap prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* dengan dukungan *tomcat website server*. Pengujian dilakukan untuk memastikan kualitas perangkat lunak yang dibangun dan kesesuaian hasil eksekusi perangkat lunak dengan analisis dan perancangan perangkat lunak.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap prototipe antarmuka aplikasi *mobile watch* berbasis *flutter* guna menguji kesesuaian dan ketepatan fungsionalitas dari seluruh sistem aplikasi dengan acuan *website* tomcat.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut ini :

a. Kemampuan Menampilkan *Form*
- *Form Login* : Verifikasi *form login* dapat menampilkan input untuk *username* dan *password*, serta menampilkan pesan kesalahan jika *login* gagal.

b. Kemampuan Menampilkan Data
- Tabel Data Pengguna : Verifikasi aplikasi menampilkan data-data
- Analisis Data : Memastikan aplikasi dapat melakukan analisis sederhana dari data yang ada, seperti laporan penggunaan harian atau mingguan.

c. Notifikasi *Realtime*
- Notifikasi Masuk : Aplikasi harus memberikan notifikasi real-time saat ada perubahan data atau tindakan yang memerlukan perhatian pengguna.
- Sinkronisasi Notifikasi : Pastikan notifikasi yang ditampilkan sesuai dengan data yang ada di *website* Tomcat.

d. Fungsi Pencarian dan Seleksi Data
- Pencarian Data : Pengujian kemampuan aplikasi untuk mencari data tertentu, seperti mencari nama pengguna atau aktivitas tertentu dalam *database*.
- Seleksi Pengguna : Verifikasi aplikasi dapat memilih pengguna atau data tanpa harus mengganti akun atau otorisasi ulang.

e. Kesesuaian Kebutuhan Non-Fungsional
- Akses dari Jaringan Terhubung : Aplikasi harus dapat diakses dan berfungsi normal selama terhubung dengan jaringan yang terhubung internet.
- Antarmuka Pengguna : Uji tampilan antarmuka agar mudah dipahami dan digunakan, dengan desain yang konsisten dan responsif pada berbagai perangkat.
- Kecepatan Akses : Pastikan aplikasi dapat mengakses data dari *database localhost* dengan waktu respon yang cepat.

f. Kesesuaian Data dengan *Website* Tomcat
- Sinkronisasi Data : Data yang ditampilkan pada aplikasi *mobile* harus sinkron dengan data yang terdapat pada *website* Tomcat.
- Konsistensi Data : Verifikasi bahwa perubahan data pada aplikasi *mobile* langsung tercermin pada *website*, dan sebaliknya.

Pengujian dengan kriteria di atas akan memastikan bahwa aplikasi *mobile* yang menggunakan *database localhost* dan acuan *website* Tomcat berfungsi dengan baik dan memenuhi kebutuhan pengguna serta spesifikasi sistem.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai admin ataupun user yang akan menjalankan fitur-fitur dan seluruh kebutuhan fungsional dari sistem. Langkah-langkah untuk setiap kebutuhan fungsional yaitu sebagai berikut :

6.3.1. Admin ataupun User

a. Admin ataupun *User* melakukan *login*.
b. Admin ataupun *User* melihat monitor alarm yang mengalami gangguan.
c. Admin ataupun User melakukan pengecekan status gangguan.
d. Admin ataupun *User* melakukan pengecekan rekomendasi.
e. Admin ataupun *User* melakukan akses pada data TSEL.
f. Admin ataupun *User* melakukan validasi maupun edit data TSEL.
g. Admin ataupun *User* melakukan akses pada data OLO.
h. Admin ataupun *User* melakukan validasi maupun edit data OLO.
i. Admin ataupun *User* melakukan akses pada data order aktif..
j. Admin ataupun *User* melakukan validasi data order aktif.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku sistem aplikasi *mobile watch* terhadap kasus skenario uji coba. Pengujian dilakukan oleh pihak pengembang, pengguna, dan pembimbing lapangan.

Tabel 6.4.1 Hasil Evaluasi Pengujian *Mobile Watch*

Kriteria Pengujian	Hasil Pengujian
Melakukan *login*	Terpenuhi
Melihat Monitor Alarm yang Mengalami Gangguan	Terpenuhi
Melakukan Pengecekan Status Gangguan	Terpenuhi
Melakukan Akses pada Data TSEL	Terpenuhi
Melakukan Validasi maupun Edit Data TSEL	Terpenuhi
Melakukan Akses pada Data OLO	Terpenuhi
Melakukan Validasi maupun Edit Data OLO	Terpenuhi
Melakukan Akses pada Data Order Aktif	Terpenuhi
Melakukan Validasi Data Order Aktif	Terpenuhi

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Setelah melakukan pengembangan aplikasi pada kegiatan Kerja Praktik di Telkom Regional 5 Semarang, dapat disimpulkan hal-hal berikut:

1. Rancangan dan Implementasi Sistem
   Aplikasi *mobile watch* berhasil dirancang dan diimplementasikan dengan mengintegrasikan *database localhost* yang memungkinkan akses dan pengelolaan data secara *real-time*. Solusi ini memberikan kecepatan dan keandalan yang optimal, sehingga mendukung efisiensi operasional di Divisi MSO Telkom Regional 5 Semarang.
2. Adopsi Fitur *Website Tomcat*
   Aplikasi *mobile watch* berhasil mengadopsi fitur-fitur utama dari *website Tomcat*, seperti pemantauan data yang aman dan terstruktur, serta kemampuan manajemen yang efisien. Hal ini menjadikan aplikasi relevan sebagai alat bantu dalam pengelolaan data secara terpusat.
3. Keandalan dalam Berbagai Kondisi Operasional
   Aplikasi *mobile watch* dilengkapi mekanisme untuk tetap berjalan dalam kondisi konektivitas terbatas, menjadikannya solusi yang andal untuk digunakan di berbagai situasi operasional. Dengan ini, aplikasi dapat diakses kapan saja oleh pegawai Divisi MSO Telkom Regional 5 Semarang.

7.2. Saran

Berikut ini adalah saran yang penulis berikan untuk arah perkembangan selanjutnya, yaitu sebaiknya dalam melakukan penerapan *backend* yang terintegrasi dengan *database* MSO agar perubahan dalam aplikasi *mobile watch* dapat terintegrasi dengan sistem - sistem lain.

DAFTAR PUSTAKA

[1] PT Telkom Indonesia, “Telkom Profil dan Sejarah,” 2024, Accessed: Nov. 12, 2024. [Online]. Available: https://www.telkom.co.id/sites/profil-telkom/id_ID/page/profil-dan-riwayat-singkat-22

[2] TECHINASIA, “P R E M I U M C O N T E N T Visual Studio Code: Aplikasi Editor Kode dari Microsoft untuk Windows, Linux, dan OS X Artikel ini merupakan bagian dari konten premium Tech in Asia ID+,” 2024. [Online]. Available: https://id.techinasia.com/visual-studio-code-editor-kode-microsoft

[3] phpMyAdmin’s, “Welcome to phpMyAdmin’s documentation!,” 2012. [Online]. Available: https://docs.phpmyadmin.net/en/latest/

[4] NIAGAHOSTER, “Pengertian jQuery Serta Fungsi dan Contohnya,” 2019. [Online]. Available: https://www.niagahoster.co.id/blog/jquery-adalah/

[5] PHP Manual, “PHP: Introduction - Manual,” 2001.

[6] OpenStreetMap, “What is OpenStreetMap? | OpenStreetMap,” 2004, Accessed: Nov. 12, 2024. [Online]. Available: https://welcome.openstreetmap.org/what-is-openstreetmap/

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

PROFIL

Nama	: Beauty Valen Fajri
Tempat, Tanggal Lahir	: Pamekasan, 18 Februari 2003
Jenis Kelamin	: Perempuan
Telepon	087750500906
Email	: beautyfajri@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	2021
Semester	: 7 (Tujuh)
Peminatan	: *Machine Learning* dan *Mobile Development*

PROFIL

Nama	: Meyroja Jovancha Firoos
Tempat, Tanggal Lahir	: Samarinda, 06 Mei 2003
Jenis Kelamin	: Perempuan
Telepon	082211812031
Email	: zovanchaj@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	2021
Semester	: 7 (Tujuh)
Peminatan	: *Machine Learning* dan *Mobile Development*