PEMROGRAMAN MOBILE DENGAN JAVA: PANDUAN LENGKAP UNTUK ANDROID DEVELOPMENT DENGAN DATABASE REALMS DAN FIREBASE

Ade Maulana
Yusuf Wahyu Setiya Putra
Irmawati
Roro Santi
Monika Evelin Johan
Ratnasari
Green Ferry Mandias

GET PRESS INDONESIA

PEMROGRAMAN MOBILE DENGAN JAVA: PANDUAN LENGKAP UNTUK ANDROID DEVELOPMENT DENGAN DATABASE REALMS DAN FIREBASE

Penulis :
Ade Maulana
Yusuf Wahyu Setiya Putra
Irmawati
Roro Santi
Monika Evelin Johan
Ratnasari
Green Ferry Mandias

ISBN :

Editor : Tri Putri Wahyuni., S.Pd
Penyunting : Diana Purnama Sari., SE.M.E
Desain Sampul dan Tata Letak : Atyka Trianisa, S.Pd

Penerbit : GET PRESS INDONESIA
Anggota IKAPI No. 033/SBA/2022

Redaksi :
Jln. Palarik Air Pacah No 26 Kel. Air Pacah
Kec. Koto Tangah Kota Padang Sumatera Barat
Website : www. getpress.co.id
Email : adm.getpress@gmail.com

Cetakan pertama, Juni 2024

KATA PENGANTAR

Segala Puji dan syukur atas kehadirat Allah SWT dalam segala kesempatan. Sholawat beriring salam dan doa kita sampaikan kepada Nabi Muhammad SAW. Alhamdulillah atas Rahmat dan Karunia-Nya penulis telah menyelesaikan Buku Pemrograman Mobile Dengan Java: Panduan Lengkap Untuk Android Development Dengan Database Realms Dan Firebase ini.

Buku Ini Membahas Pendahuluan dan Persiapan Pemrograman, Dasar Pemrograman Android: Komponen UI dan Android Manifest, Konsep Activity dan Intent, Android Layouts: Tampilan dan Grup Layout, Android Widget dan Styling, Fragment: Membangun Antarmuka yang Fleksibel, Database Realms: Query dan Migrasi.

Proses penulisan buku ini berhasil diselesaikan atas kerjasama tim penulis. Demi kualitas yang lebih baik dan kepuasan para pembaca, saran dan masukan yang membangun dari pembaca sangat kami harapkan.

Penulis ucapkan terima kasih kepada semua pihak yang telah mendukung dalam penyelesaian buku ini. Terutama pihak yang telah membantu terbitnya buku ini dan telah mempercayakan mendorong, dan menginisiasi terbitnya buku ini. Semoga buku ini dapat bermanfaat bagi masyarakat Indonesia.

Padang, Juni 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 1
PENDAHULUAN DAN PERSIAPAN PEMROGRAMAN

Oleh Ade Maulana

1.1 Pengantar Pemrograman Mobile

Aplikasi mobile telah mengubah cara berinteraksi dengan dunia sekitar. Misalnya, aplikasi transportasi seperti Grab dan Gojek menyediakan opsi transportasi yang nyaman, mengurangi lalu lintas, dan mendukung keberlanjutan lingkungan. Di bidang e-commerce, aplikasi seperti Tokopedia dan Shopee memungkinkan pengguna untuk berbelanja dengan mudah melalui fitur pencarian produk dan pembayaran online yang praktis (Maulana, Satrio, et al., 2023). Dalam hal kesehatan dan kebugaran, aplikasi seperti Garmin dan Strava membantu pengguna memantau aktivitas fisik mereka dengan sensor terintegrasi. Selain itu, aplikasi pendidikan seperti Duolingo menawarkan akses pembelajaran yang fleksibel dan terjangkau. Sementara itu, aplikasi keuangan seperti PayPal mempermudah transaksi online, transfer uang, dan pembayaran tagihan dengan cepat dan aman. Melalui aplikasi mobile, pengguna dapat dengan mudah mengakses layanan dan informasi, meningkatkan kenyamanan, efisiensi, dan kualitas hidup secara keseluruhan.

Pemrograman Mobile, sebagai cabang khusus dalam ilmu komputer, bertujuan untuk mengembangkan perangkat lunak untuk perangkat bergerak seperti smartphone dan tablet. Fokusnya adalah menciptakan aplikasi yang berjalan efisien dan memberikan pengalaman pengguna optimal pada perangkat dengan sumber daya terbatas. Dengan karakteristik unik dari perangkat mobile, seperti layar sentuh dan sensor, pemrograman mobile melibatkan konsep-konsep khusus untuk memastikan responsifitas dan adaptabilitas aplikasi terhadap berbagai ukuran layar. (S et al., 2023).

Selain itu, peran internet dalam ekosistem aplikasi mobile semakin penting. Menurut laporan WeAreSocial pada bulan Oktober 2023, pengguna handphone pintar mencapai 5.6 miliar, sementara pengguna internet mencapai 5.3 miliar (Wearesocial, 2023). Internet menjadi fondasi bagi aplikasi mobile modern, memungkinkan akses ke layanan, konten, dan fitur tambahan. Koneksi yang terus-menerus memungkinkan pengguna untuk mengalami aplikasi secara real-time, mentransfer data dengan cepat, dan mengeksplorasi fitur yang tersedia (Muttaqin et al., 2023). Internet juga mendukung kolaborasi antar pengguna, memfasilitasi pertukaran informasi dan interaksi sosial di dalam aplikasi. Dalam pengembangan aplikasi mobile, integrasi internet menjadi krusial untuk memastikan pengalaman pengguna yang terkoneksi dan holistik.

1.2 Konsep Dasar Pemrograman Mobile

Konsep dasar pemrograman mobile membentuk dasar bagi pengembang dalam merancang dan mengimplementasikan aplikasi untuk perangkat seluler. Salah satu konsep kunci adalah responsivitas, yang mengharuskan aplikasi memberikan pengalaman pengguna yang mulus dan efisien, bahkan pada perangkat dengan ukuran layar yang berbeda. Selain itu, pemrograman mobile sering kali melibatkan pemahaman tentang siklus hidup aplikasi, di mana pengembang perlu mengelola proses mulai dari pembuatan hingga penghentian aplikasi. Manajemen sumber daya seperti memori dan baterai juga menjadi aspek penting, memastikan aplikasi dapat berjalan secara efisien tanpa mengorbankan kinerja atau daya tahan baterai. Konsep-konsep dasar lainnya mencakup pengelolaan data, integrasi API, keamanan, dan antarmuka pengguna yang intuitif. Pemahaman yang mendalam tentang platform yang digunakan, seperti Android atau iOS, serta penguasaan bahasa pemrograman yang sesuai, menjadi kunci kesuksesan dalam menerapkan konsep dasar pemrograman mobile. Dengan mengintegrasikan semua elemen ini, pengembang dapat

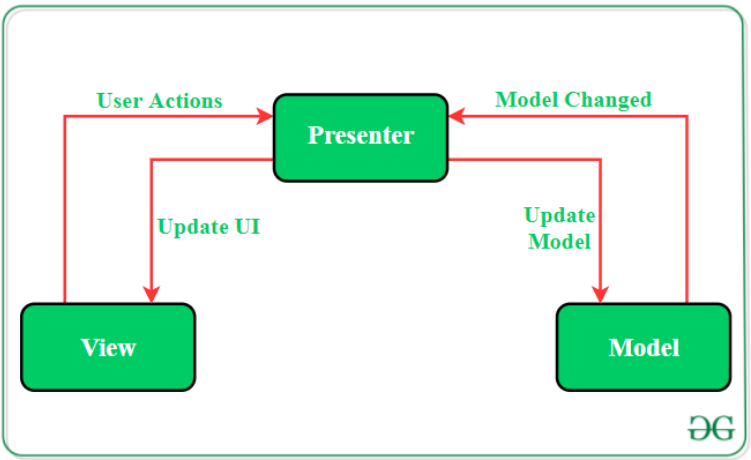
menciptakan aplikasi mobile yang handal, efisien, dan memenuhi harapan pengguna modern.

1.2.1 Paradigma Pemrograman Mobile

Paradigma pemrograman mobile Android didasarkan pada pendekatan yang terpusat pada pengembangan aplikasi untuk sistem operasi Android. Secara khusus, Android mengadopsi paradigma pemrograman berorientasi objek (OOP), yang memungkinkan pengembang untuk memodelkan dan mengorganisasi kode mereka dalam bentuk objek yang bersifat modular dan dapat digunakan kembali. Konsep pewarisan, polimorfisme, dan enkapsulasi menjadi bagian integral dari paradigma ini, memungkinkan pembuatan *clean code*, mudah dipelajari, dan mudah dikelola.

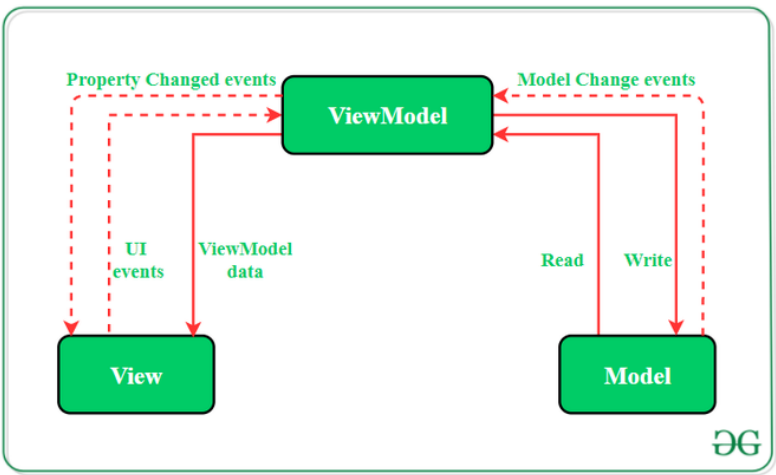
Selain OOP, paradigma pemrograman Android juga menyoroti penggunaan antarmuka pengguna (UI) yang responsif dan interaktif. Pengembang perlu memahami konsep tata letak (layout) dan widget untuk merancang antarmuka yang intuitif dan menarik bagi pengguna. Komponen utama aplikasi Android, seperti aktivitas, layanan, dan penerima siaran, bekerja bersama dalam siklus hidup aplikasi yang harus dikelola dengan cermat.

Android mendukung berbagai *design pattern* pengembangan aplikasi, dan keputusan untuk menggunakan MVP (*Model-View-Presenter*), MVVM (*Model-View-ViewModel*), atau *design pattern* lainnya tergantung pada preferensi pengembang dan kebutuhan proyek. Android itu sendiri tidak membatasi penggunaan satu arsitektur tertentu dan memberikan kebebasan kepada pengembang untuk memilih metode yang sesuai dengan proyek mereka.



Gambar 1.1. Model-View-Presenter (Geeksforgeeks, 2023)

Beberapa pengembang Android memilih menggunakan *design pattern* Model-View-ViewModel (MVVM), yang memisahkan logika presentasi dari tampilan dan *bisnis logic* (Maulana, Heryana, et al., 2023). Dalam konteks Android, framework Jetpack menyediakan arsitektur komponen, seperti ViewModel, yang mendukung pendekatan MVVM. Penggunaan LiveData, komponen ViewModel, dan Data Binding adalah elemen-elemen yang umumnya dihubungkan dengan pendekatan MVVM di Android.



Gambar 1.2. Model-View-ViewModel (Geeksforgeeks, 2023)

Di sisi lain, MVP tetap menjadi pendekatan yang populer di kalangan pengembang Android, terutama karena pendekatannya yang bersih dan pemisahan yang jelas antara Model, View, dan Presenter. Beberapa pengembang mungkin juga memilih menggunakan arsitektur *Clean Architecture*, yang memadukan prinsip-prinsip dari MVP atau MVVM dengan pemisahan yang lebih lanjut antara lapisan-lapisan aplikasi.

1.2.2 Bahasa Pemrograman

Java memainkan peran kunci dalam pengembangan aplikasi mobile Android, java menjadi bahasa pemrograman utama yang secara luas digunakan oleh pengembang. Keunggulan Java dalam ekosistem Android tidak hanya terletak pada sintaksnya yang jelas dan mudah dipahami, tetapi juga pada portabilitasnya yang memungkinkan pengembang untuk menulis kode sekali dan menjalankannya di berbagai perangkat Android tanpa perlu memodifikasinya. Android Studio, lingkungan pengembangan terdepan untuk platform ini, memberikan dukungan penuh untuk Java, mempermudah proses pengembangan aplikasi. Keamanan yang dimiliki oleh Java juga menjadi faktor penting, membantu melindungi aplikasi Android dari berbagai ancaman keamanan. Dengan model pemrograman berorientasi objek (OOP), Java memfasilitasi pengembangan aplikasi yang terstruktur, modular, dan mudah dikelola. Java di Android juga memiliki keuntungan dari dukungan komunitas yang besar dan aktif, memberikan akses ke sumber daya, tutorial, dan solusi yang melimpah. Namun, Kotlin telah muncul sebagai bahasa pemrograman alternatif yang semakin populer dalam pengembangan aplikasi Android. Kotlin menawarkan sintaksis yang lebih ringkas dan ekspresif, serta fitur-fitur modern yang meningkatkan produktivitas pengembang. Meskipun demikian, keunggulan Java membuatnya tetap menjadi pilihan yang kokoh dan andal dalam pengembangan aplikasi mobile Android, memastikan keseimbangan antara performa, keamanan, dan keterbacaan kode.

1.2.3 Alat Bantu Pengembangan

Alat bantu pengembangan, dikenal juga sebagai Integrated Development Environment (IDE) yang digunakan dalam pembuatan aplikasi Android memiliki peran kritis dalam menyederhanakan dan meningkatkan produktivitas para pengembang. Salah satu IDE paling populer untuk pengembangan Android adalah Android Studio. Android Studio adalah lingkungan pengembangan resmi dari Google yang menawarkan berbagai fitur dan alat untuk mendukung proses pengembangan. Dengan emulator Android Studio, pengembang dapat menguji aplikasi pada berbagai perangkat virtual tanpa perlu memiliki perangkat fisik. Fitur intelejen kode, tata letak desain visual, dan integrasi dengan GitHub membuat pengembangan aplikasi menjadi lebih efisien. Android Studio juga mendukung berbagai bahasa pemrograman, termasuk Java dan Kotlin.

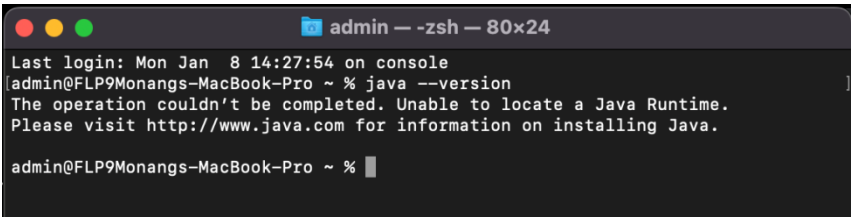
Selain Android Studio, terdapat alat bantu lainnya yang mendukung pengembangan aplikasi Android, seperti IntelliJ IDEA, Eclipse, dan Visual Studio. Beberapa alat ini juga dapat diintegrasikan dengan Android SDK untuk memberikan fleksibilitas yang lebih besar kepada para pengembang. Semua alat bantu ini bertujuan untuk memfasilitasi pengembangan aplikasi dengan menyediakan lingkungan yang kuat, intuitif, dan berfitur lengkap. Dengan demikian, para pengembang dapat fokus pada kreativitas dan inovasi dalam merancang aplikasi Android yang responsif, fungsional, dan sesuai dengan kebutuhan pengguna.

1.3 Persiapan Lingkungan Kerja

Persiapan lingkungan kerja untuk pengembangan aplikasi Android melibatkan beberapa langkah penting yang mencakup pengaturan JDK (*Java Development Kit*), Android Studio, dan *Android Virtual Device* (AVD). Langkah-langkah ini memastikan bahwa pengembang memiliki lingkungan yang tepat untuk membuat, menguji, dan mendebug aplikasi Android secara efisien.

1.3.1 Java Development Kit

Untuk memastikan bahwa JDK (*Java Development Kit*) telah terinstal di sistem Anda, langkah-langkahnya cukup sederhana. Pertama, buka *Command Prompt* (cmd) atau terminal di komputer Anda. Setelah Command Prompt terbuka, ketik perintah java --version dan tekan tombol Enter.



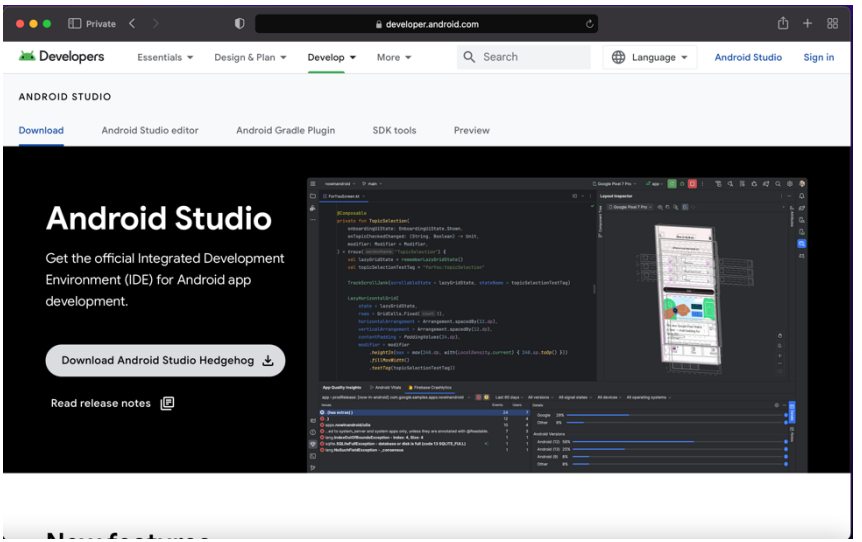
Gambar 1.3. Hasil Eksekusi Versi Java pada terminal

Jika JDK sudah terinstal, Anda akan melihat versi Java yang terpasang di komputer Anda. Namun, jika perintah tersebut tidak menghasilkan keluaran atau menampilkan pesan kesalahan yang menyatakan bahwa perintah java tidak ditemukan, maka hal itu berarti JDK belum terinstal di sistem Anda. Untuk menginstal JDK, Anda dapat mengunjungi situs resmi Oracle di oracle.com atau sumber terpercaya lainnya yang menyediakan paket instalasi JDK sesuai dengan sistem operasi Anda. Setelah mengunduh paket instalasi, jalankan file instalasi dan ikuti petunjuk yang muncul di layar untuk menyelesaikan proses instalasi. Setelah selesai, kembali buka Command Prompt dan cek kembali versi Java dengan perintah java --version untuk memastikan bahwa instalasi JDK telah berhasil. Dengan langkah-langkah ini, Anda dapat memastikan bahwa JDK telah terinstal dengan benar di sistem Anda dan siap digunakan untuk pengembangan aplikasi Java.

1.3.2 Android Studio

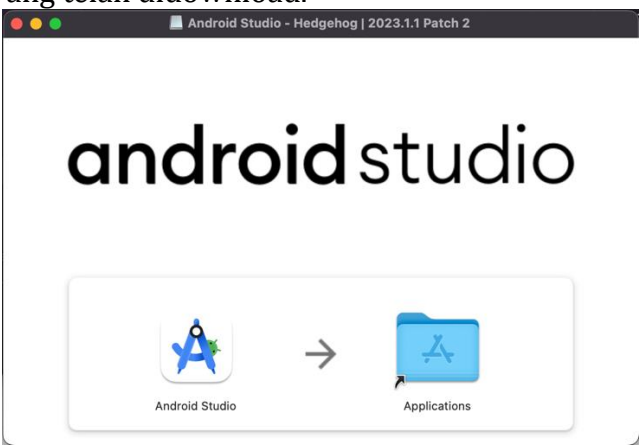
Untuk menginstall dan mengatur Android Studio, langkah-langkah berikut dapat diikuti dengan mudah. Pertama, pastikan bahwa komputer kompatibel dengan persyaratan sistem yang dibutuhkan oleh Android Studio, termasuk RAM minimal 8 GB, dan ruang disk kosong minimal 4 GB serta

minimum resolusi layar 1280 x 800. Setelah memastikan hal tersebut, unduh Android Studio dari situs web pengembang Android yakni developer.android.com.



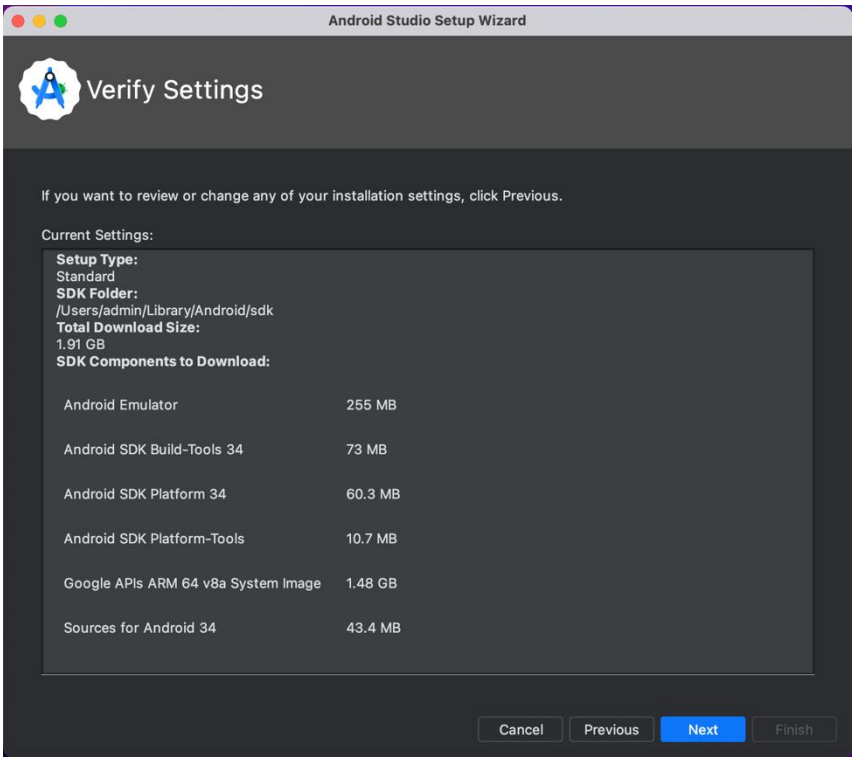
Gambar 1.4. Halaman Utama Situs Developer Android

Simpan file .dmg yang diunduh. Kemudian, buka file .dmg tersebut dan seret ikon Android Studio ke folder "Applications" untuk menyelesaikan proses instalasi jika anda menggunakan MacOS. Jika menggunakan windows maka bisa double klik pada file .exe yang telah didownload.



Gambar 1.5. Tampilan Utama File Dmg Android Studio

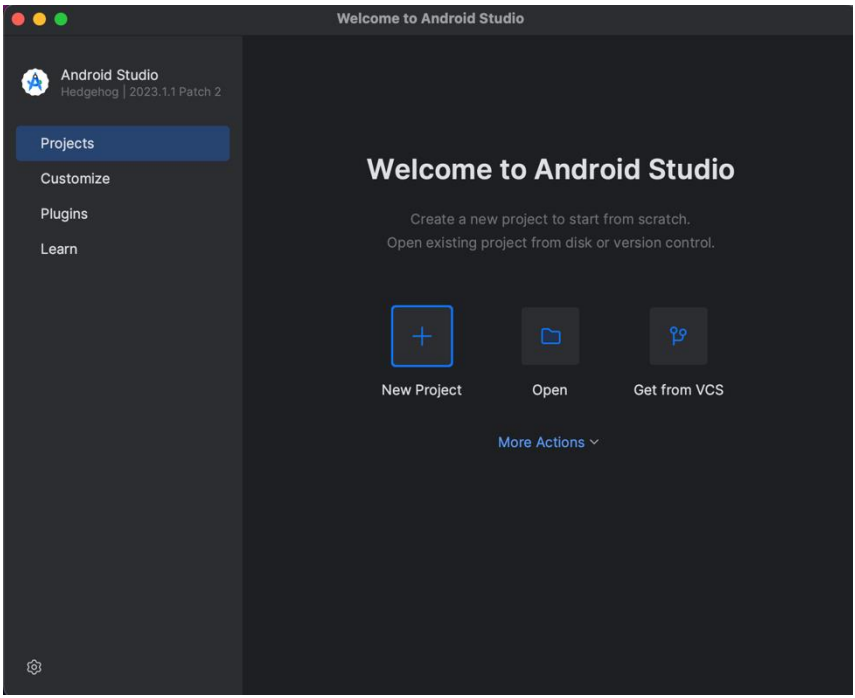
Setelah instalasi selesai, buka Android Studio dari folder "Applications" dan lakukan konfigurasi awal, seperti menyetujui persyaratan lisensi, memilih tema, dan konfigurasi lainnya (Gambar 1.6). Pastikan juga untuk mengunduh dan menginstal *Java Development Kit* (JDK) jika belum terpasang di sistem Anda. Setelah itu, konfigurasikan SDK dengan memilih "*Configure*" dari layar selamat datang atau dari menu Android Studio dan pilih "SDK Manager" untuk mengatur SDK Platform sesuai kebutuhan proyek Anda.



Gambar 1.6. Android Studio Setup Wizard

Terakhir, buka proyek Android Studio atau buat proyek baru untuk memastikan semuanya berfungsi dengan baik, dan pastikan untuk menginstal pembaruan SDK yang tersedia jika diperlukan. Dengan demikian, Anda telah berhasil menyiapkan

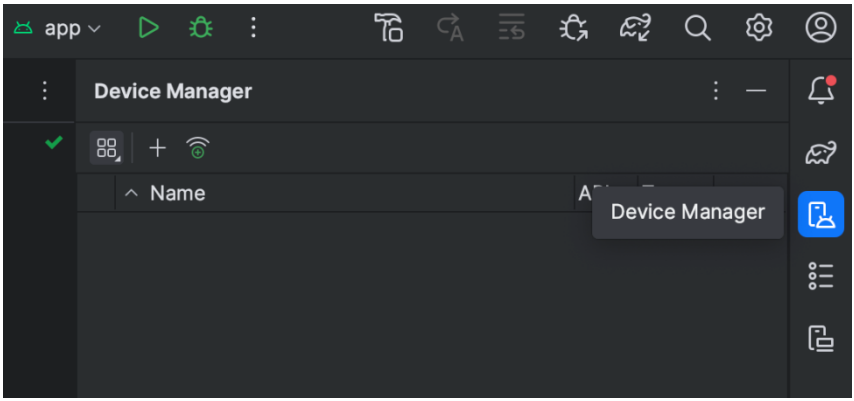
Android Studio di macOS dan siap untuk mulai mengembangkan aplikasi Android Anda.



Gambar 1.7. Halaman Utama Android Studio

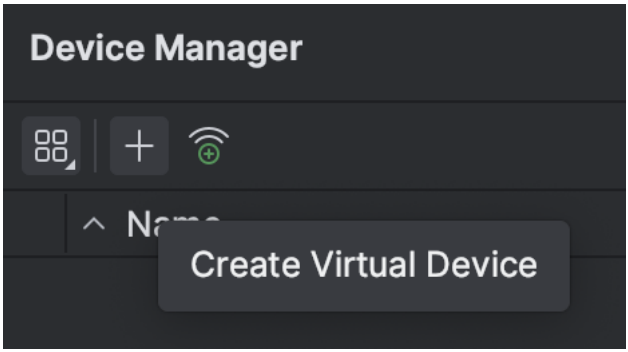
1.3.3 Android Virtual Device

Untuk menambahkan *Android Virtual Device* (AVD) pada Device Manager di Android Studio, langkah-langkahnya cukup sederhana. Pertama, buka Android Studio dan arahkan kursor ke bilah toolbar atas. Di sana, Anda akan menemukan ikon "*Device Manager*", yang biasanya berbentuk ponsel dengan garis-garis. Klik ikon tersebut untuk membuka *Android Virtual Device Manager*.



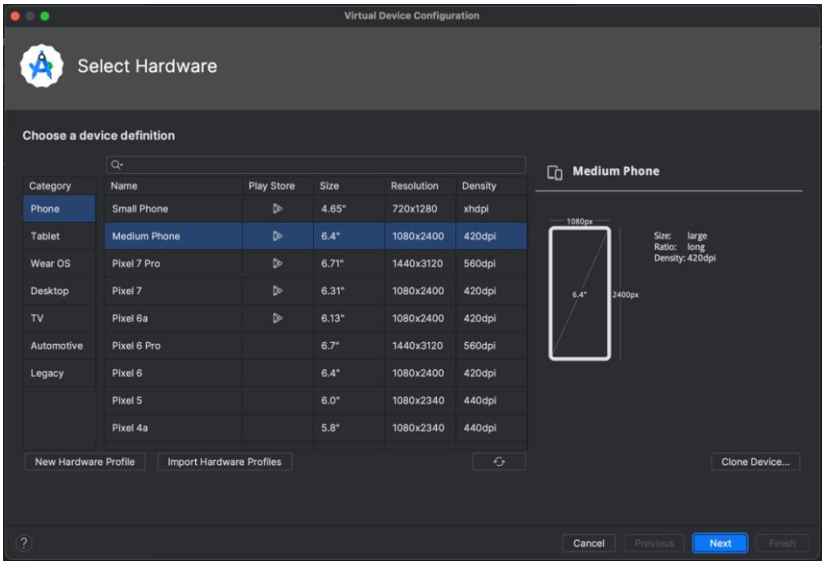
Gambar 1.8. Menu Device Manager

Selanjutnya, di jendela AVD Manager, pilih opsi "*Create Virtual Device*" untuk membuat perangkat virtual baru.



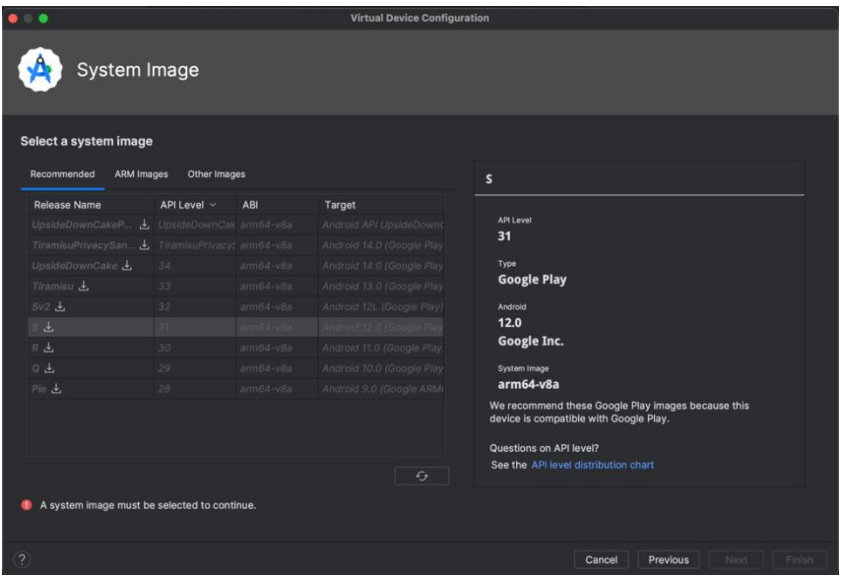
Gambar 1.9. Menu Create Virtual Device

Anda akan dibawa ke layar "Select Hardware" di mana Anda dapat memilih tipe perangkat yang ingin Anda buat, seperti Google Pixel atau Nexus. Setelah memilih, klik "Next" dan pilih sistem operasi Android yang diinginkan untuk perangkat virtual Anda.



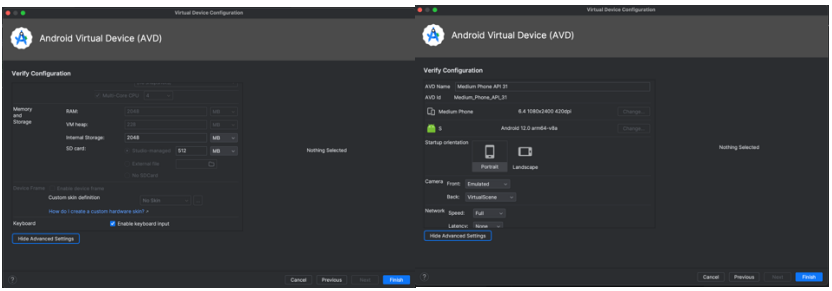
Gambar 1.10. Pemilihan Tipe Perangkat pada *Android Virtual Device*

Jika sistem operasi tersebut belum diunduh, Anda dapat mengunduhnya dari daftar yang tersedia.



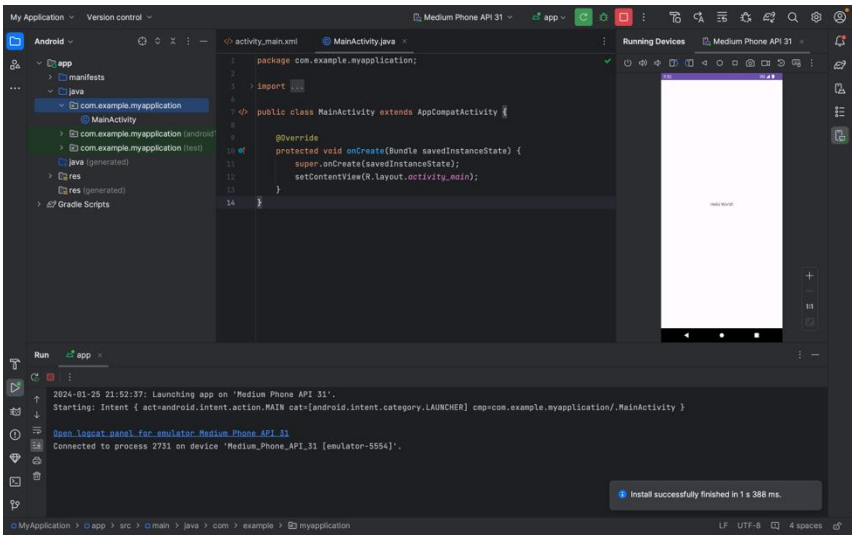
Gambar 1.11. Pemilihan SDK Android pada Android Virtual Device

Selanjutnya, sesuaikan konfigurasi perangkat virtual, seperti nama perangkat, RAM, penyimpanan, dan orientasi layar, kemudian klik "Finish" untuk menyelesaikan proses.



Gambar 1.12. Konfigurasi Perangkat Virtual pada Android Virtual Device

Dengan demikian, Anda telah berhasil menambahkan *Android Virtual Device* (AVD) ke Device Manager di Android Studio, dan Anda dapat mulai menggunakan perangkat virtual tersebut untuk menguji aplikasi Android Anda.



Gambar 1.13. Eksekusi program dan Android Virtual Device pada Android Studio.

1.3.4 Developer Option

Selain menggunakan *Android Virtual Device* (AVD) untuk menguji aplikasi Android, pengembang juga dapat menggunakan ponsel pribadi sebagai perangkat pengujian. Penggunaan ponsel pribadi menawarkan beberapa keuntungan yang tidak dapat disediakan oleh AVD, seperti pengujian aplikasi pada perangkat fisik yang sesungguhnya dengan berbagai konfigurasi perangkat keras dan sistem operasi yang berbeda.

Untuk dapat menggunakan ponsel pribadi sebagai perangkat pengujian untuk pengembangan aplikasi Android, Anda perlu mengaktifkan *Developer Options* terlebih dahulu. Cara mengaktifkan *Developer Options* pada ponsel Android bervariasi sedikit tergantung pada merek dan model ponsel Anda, tetapi umumnya langkah-langkahnya sebagai berikut:

1. Buka Pengaturan (*Settings*): Buka aplikasi Pengaturan pada ponsel Android Anda. Anda dapat mengaksesnya melalui ikon roda gigi atau melalui menu drop-down di layar utama.
2. Cari Informasi Perangkat atau Tentang Ponsel (*About Phone*): Biasanya, Anda dapat menemukan opsi "Informasi Perangkat" atau "Tentang Ponsel" di bagian bawah menu Pengaturan.
3. Cari Nomor Bangunan (*Build Number*): Di dalam menu "Informasi Perangkat" atau "Tentang Ponsel", cari opsi yang menyebutkan "Nomor Bangunan" atau "Build Number".
4. Klik *Build Number* Berkali-kali: Anda perlu mengklik atau tekan *build number* tersebut beberapa kali, biasanya tujuh kali secara berurutan. Setelah beberapa kali, Anda akan melihat pesan yang memberi tahu Anda bahwa Anda telah mengaktifkan Developer Options.
5. Kembali ke Pengaturan Utama: Setelah Anda mengaktifkan *Developer Options*, kembali ke menu Pengaturan utama pada ponsel Anda.
6. Temukan dan Buka *Developer Options*: Sekarang, Anda akan melihat opsi baru yang disebut "*Developer Options*" di dalam menu Pengaturan. Buka opsi ini.

Setelah Anda mengaktifkan *Developer Options*, Anda dapat menggunakan ponsel Anda sebagai perangkat pengujian

untuk mengembangkan aplikasi Android. Anda dapat menghubungkan ponsel Anda ke komputer menggunakan kabel USB dan mengaktifkan USB debugging di dalam opsi Developer Options. Setelah USB debugging diaktifkan, Anda dapat menjalankan aplikasi langsung dari Android Studio ke ponsel Anda untuk pengujian dan debugging. Pastikan juga untuk mengatur opsi lainnya sesuai kebutuhan Anda, seperti mengizinkan instalasi aplikasi dari sumber yang tidak dikenal jika diperlukan.

Dengan demikian, Anda dapat mengembangkan dan menguji aplikasi Android Anda dengan menggunakan ponsel pribadi sebagai perangkat fisik, memberikan pengalaman pengujian yang lebih realistis dan akurat.

DAFTAR PUSTAKA

Geeksforgeeks. (2023, February). *Difference Between MVP and MVVM Architecture Pattern in Android*. https://www.geeksforgeeks.org/difference-between-mvp-and-mvvm-architecture-pattern-in-android/

Maulana, A., Heryana, N., Pasaribu, J. S., Aditya, A., Elisawati, Rudiansyah, Amna, Permana, A. A., Rukmana, A. Y., Abdillah, R., & Wahyono, T. (2023). *Rekayasa Perangkat Lunak: Konsep, Metode, dan Praktik Terbaik*. Global Eksekutif Teknologi.

Maulana, A., Satrio, D., Hasibuan, A., Nasution, S. P., Munte, R. N., Hutabarat, M. L. P., Arief, M. H., Mandagi, D. W., Hawa, S. D., Bukidz, D. P., & others. (2023). *Manajemen Bisnis Digital dan E-Commerce*. Yayasan Kita Menulis.

Muttaqin, M., Romindo, R., Moedjahedy, J., Pratama, Y. A., Andryanto, A., Sihananto, A. N., Widarman, A., Kurniadi, W., Maulana, A., Simarmata, J., & others. (2023). *Pengantar Internet*. Yayasan Kita Menulis.

S, W., Maulana, A., Umar, N., Permana, A. A., Safii, M., Adhicandra, I., Pasaribu, J. S., A, H., Mayefis, R., & Waworuntu, A. (2023). *PEMOGRAMAN MOBILE*. Global Eksekutif Teknologi. https://books.google.co.id/books?id=3b7FEAAAQBAJ

Wearesocial. (2023, October). *DIGITAL 2023 OCTOBER GLOBAL STATSHOT REPORT*. https://wearesocial.com/id/blog/2023/10/digital-2023-october-global-statshot-report/

BAB 2
DASAR PEMROGRAMAN ANDROID : KOMPONEN UI DAN ANDROID MANIFEST

Oleh Yusuf Wahyu Setiya Putra

2.1 Pendahuluan

Pada bab kedua ini akan dijelaskan tentang apa saja dasar dan pondasi yang ada serta harus dilakukan sebelum kita melakukan sebuah pengkodean berbasis perangkat bergerak untuk menciptakan sebuah aplikasi android yang berkualitas dan berfungsi dengan baik sebagaimana tujuan dari pembuatannya.

Gambar 2.1. Ilustrasi Dasar Pemrograman Android (Sumber : https://niitbirgunj.files.wordpress.com/2015/03/training-prof.png)

Android sendiri adalah sistem operasi yang pada dasarnya dibuat untuk ponsel. Ini didasarkan pada Kernel Linux dan perangkat lunak sumber terbuka lainnya dan dikembangkan oleh Google. Ini digunakan untuk perangkat seluler layar sentuh seperti ponsel cerdas dan tablet. Namun

kini ini digunakan di mobil Android Auto, TV, jam tangan, kamera, dll. Ini telah menjadi salah satu OS terlaris untuk ponsel pintar. OS Android dikembangkan oleh Android Inc. yang dibeli Google pada tahun 2005. Berbagai aplikasi (aplikasi) seperti game, pemutar musik, kamera, dll dibuat untuk smartphone ini agar dapat berjalan di Android. Google Play Store menampilkan lebih dari 3,3 juta aplikasi. Aplikasi ini dikembangkan pada aplikasi yang dikenal sebagai Android Studio. Aplikasi yang dapat dieksekusi ini diinstal melalui bundel atau paket yang disebut APK (Android Package Kit).

2.2 Bahasa Pemrograman JAVA

JAVA merupakan bahasa pemrograman yang dibangun oleh sebuah tim pimpinan James Gosling di Sun Microsystems. Pada tahun 1991 nama Java masih digunakan sebagai Oak, dimana Java dirancang khusus sebagai bahasa pemrograman yang digunakan untuk chip yang tertanam. Kemudian pada tahun 1995 muncul nama Java menggantikan Oak dan didesain ulang lagi untuk membuat aplikasi internet. Salah satu kelebihan bahasa pemrograman Java adalah WORA (Write Once, run anywhere) dimana bahasa pemrograman Java dapat dibangun satu kali saja pada platform tertentu dan nantinya dapat dijalankan pada platform/perangkat lain selama ada Java. Mesin virtual.

Gambar 2.2. Logo JAVA

Spesifikasi bahasa Java berisi informasi tentang bahasa pemrograman Java yang meliputi sintaksis penulisan dan aturan semantik. Sedangkan kelas atau antarmuka yang telah didefinisikan terlebih dahulu agar nantinya dapat digunakan disebut Java API (*Application Program Interface*). Keduanya (spesifikasi bahasa Java dan Java API) merupakan komponen yang mendukung Java Standard.

Selain itu juga terdapat JDK (*Java Development Toolkit*) yang diperlukan ketika kita ingin membangun suatu program dengan menggunakan bahasa Java. Di dalam JDK sendiri terdapat beberapa macam program seperti compiler, interpreter, debugger, applet viewer, dokumentasi dan kompresor. Kompiler digunakan untuk menerjemahkan kode Java yang dibuat (file .java) ke dalam bentuk bytecode. Kompiler akan menghasilkan bytecode dalam bentuk ekstensi .class. Interpreter pada Java digunakan untuk mengeksekusi bytecode yang telah dihasilkan oleh compiler sehingga dapat ditampilkan hasil aplikasinya. Untuk melakukan debugging, Anda dapat menggunakan Java Debugger (jdb). Kompresor di Java akan menghasilkan output dalam format .Jar dimana file tersebut dapat berisi banyak file kelas lain, atau metadata dan sumber daya lain seperti gambar, video, dll.

Program yang digunakan untuk menjalankan aplikasi Java adalah *Java Runtime Environment* (JRE). Di JRE terdapat *Java Virtual Machine* (JVM) beserta pustaka standar lainnya. JVM akan mengeksekusi bytecode Java di komputer mana pun (yang mendukung slogan WORA). Instruksi mesin dalam bytecode Java akan diinterpretasikan dan dieksekusi oleh JVM. Oleh karena itu, ketika kita ingin membuat suatu aplikasi dengan menggunakan bahasa pemrograman Java, kita memerlukan JDK. Pada saat proses instalasi JDK, JRE juga akan diinstal. Sedangkan ketika kita hanya ingin menjalankan bytecode Java di komputer lain, kita hanya membutuhkan JRE saja.

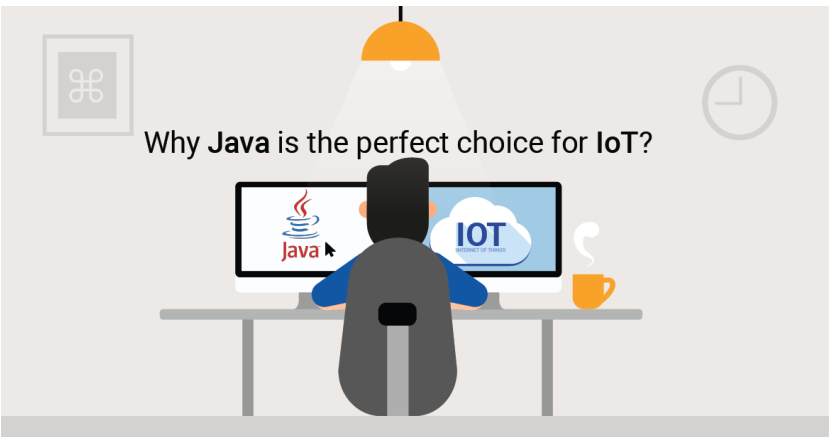
2.3 Keuntungan Bahasa Pemrograman JAVA

Ketika memilih bahasa pemrograman dan lingkungan untuk aplikasi perusahaan Anda berikutnya, ada alasan teknis

yang kuat untuk mempertimbangkan Java, termasuk interoperabilitas, skalabilitas, dan kemampuan beradaptasi.

Filosofi inti di balik penciptaannya—interoperabilitas antar perangkat yang berbeda—masih menjadi argumen terkuat yang mendukung Java untuk aplikasi perusahaan baru. Arsitektur berorientasi objek Java memungkinkan Anda membuat program modular dan kode yang dapat digunakan kembali, memperpendek siklus pengembangan dan memperpanjang umur aplikasi perusahaan.

Skalabilitas platform adalah atribut utama Java. Dengan Java, Anda dapat menggunakan satu sistem di berbagai kasus penggunaan. Aplikasi desktop yang ada dapat dengan mudah diadaptasi untuk dijalankan pada perangkat kecil yang memiliki sumber daya terbatas. Anda juga dapat memigrasikan aplikasi dari seluler ke desktop, mengembangkan aplikasi bisnis untuk platform Android dan kemudian mengintegrasikannya ke dalam perangkat lunak desktop Anda saat ini, melewati siklus pengembangan yang panjang dan mahal.



Gambar 2.3. JAVA dan IoT
(Sumber : https://oneteam.us/wp-content/uploads/2018/07/Why-Java-is-the-perfect-choice-for-IoT.png)

Java juga memenangkan poin di mata para perencana strategis karena kemampuannya beradaptasi dengan kasus

penggunaan baru. Misalnya, Java secara luas dianggap sebagai platform ideal untuk *Internet of Things* (IoT). Aplikasi IoT pada umumnya menghubungkan sejumlah besar perangkat yang berbeda, sebuah tugas yang sangat disederhanakan dengan fakta bahwa miliaran perangkat menjalankan Java. Selain itu, ekosistem pengembang Java yang luas terus mengembangkan dan berbagi perpustakaan baru dengan fungsionalitas yang secara khusus ditargetkan untuk pengembangan aplikasi IoT.

Gambar 2.4. JAVA untuk Bisnis
(Sumber : https://www.techsling.com/top-benefits-of-using-java-for-your-business/)

Argumen teknis yang mendukung Java sangat menarik, namun alasan bisnis untuk memilih Java juga sama kuatnya: kumpulan talenta yang besar, kurva pembelajaran yang singkat, dan beragam lingkungan pengembangan terintegrasi (IDE).

Karena semakin banyak perusahaan yang menggunakan perangkat yang terhubung, algoritme pembelajaran mesin, dan solusi cloud, permintaan akan pengembang yang terampil terus meningkat. Banyak analis memperkirakan akan adanya kelangkaan pemrogram tingkat senior dalam waktu dekat, sehingga menyulitkan untuk mempekerjakan staf dalam inisiatif

perangkat lunak baru. Permintaan untuk pengembang aplikasi seluler akan segera melampaui pasokan yang tersedia.

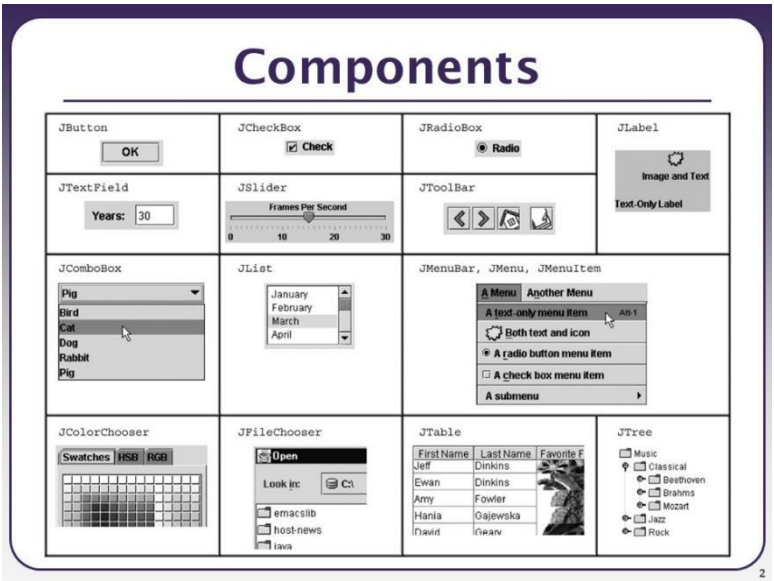
Banyaknya talenta pengembang Java merupakan alasan kuat untuk mendasarkan inisiatif perangkat lunak besar di Java. Ketika manajer kepegawaian memposting lowongan pekerjaan untuk pengembang Java, mereka dapat berharap untuk menerima banyak resume yang memenuhi syarat dan mengisi posisi tersebut dengan relatif cepat. Manajer juga dapat memanfaatkan sumber daya kontrak untuk menambah staf internal untuk tugas-tugas tertentu tanpa menambah jumlah karyawan.

Selain pengembang tingkat senior, inisiatif perangkat lunak besar juga memerlukan kontributor junior dalam jumlah besar. Meskipun Java tetap menjadi bahasa pemrograman pengantar yang populer dalam kurikulum ilmu komputer universitas, banyak lulusan yang kurang memiliki kemahiran untuk menjadi produktif pada hari pertama. Java lebih mudah dipelajari dan dikuasai dibandingkan bahasa pemrograman lainnya, sehingga menghasilkan kurva pembelajaran yang lebih pendek dan peningkatan produktivitas yang lebih cepat. Komunitas online Java yang luas yang terdiri dari forum pengembang, tutorial, dan kelompok pengguna membantu pemula mendapatkan informasi dengan cepat dan menyediakan alat pemecahan masalah yang efektif dan terbukti bagi pemrogram berpengalaman.

Di bidang alat pemrograman, Java menawarkan berbagai IDE. Pengembang Java yang berpengalaman dapat dengan cepat memanfaatkan lingkungan baru, sehingga manajer pengembangan bebas memilih IDE yang paling sesuai dengan jenis proyek, anggaran, metodologi pengembangan, dan tingkat keterampilan pemrogram. Banyak programmer Java berpengalaman menganggap NetBeans, Eclipse, dan IntelliJ IDEA sebagai tiga IDE teratas untuk pengembangan aplikasi perusahaan. Namun ada kalanya IDE yang lebih ringan seperti DrJava, BlueJ, JCreator, atau Eclipse Che adalah pilihan terbaik.

2.4 Komponen *User Interface* pada JAVA

Komponen *JavaServer Faces* adalah blok penyusun tampilan *JavaServer Faces*. Komponen dapat berupa komponen antarmuka pengguna (UI) atau komponen non-UI.



Gambar 2.5. Contoh Komponen UI pada JAVA
(Sumber :
https://slideplayer.com/slide/9285478/28/images/2/Components.jpg)

Komponen UI *JavaServer Faces* adalah elemen yang dapat dikonfigurasi dan digunakan kembali yang menyusun antarmuka pengguna aplikasi JavaServer Faces. Komponen dapat berbentuk sederhana, seperti tombol, atau dapat berupa gabungan, seperti tabel, yang terdiri dari beberapa komponen.
Teknologi *JavaServer Faces* menyediakan arsitektur komponen yang kaya dan fleksibel yang mencakup hal-hal berikut:

1. Satu set kelas javax.faces.component.UIComponent untuk menentukan status dan perilaku komponen UI
2. Model rendering yang menentukan cara merender komponen dengan berbagai cara

3. Model konversi yang menentukan cara mendaftarkan konverter data ke komponen
4. Model peristiwa dan pendengar yang menentukan cara menangani peristiwa komponen
5. Model validasi yang menentukan cara mendaftarkan validator ke suatu komponen
6. Model navigasi yang menentukan navigasi halaman dan urutan pemuatan halaman

Bagian ini menjelaskan secara singkat masing-masing bagian arsitektur komponen ini.

2.4.1 Kelas Komponen Antarmuka Pengguna

Teknologi JavaServer Faces menyediakan serangkaian kelas komponen UI dan antarmuka perilaku terkait yang menentukan semua fungsi komponen UI, seperti mempertahankan status komponen, mempertahankan referensi ke objek, dan mendorong penanganan dan rendering peristiwa untuk serangkaian komponen standar.

Kelas komponen sepenuhnya dapat diperluas, memungkinkan penulis komponen membuat komponen khusus mereka sendiri. Kelas dasar abstrak untuk semua komponen adalah javax.faces.component.UIComponent. Kelas komponen UI JavaServer Faces memperluas kelas UIComponentBase (subkelas UIComponent), yang mendefinisikan keadaan default dan perilaku komponen. Kumpulan kelas komponen berikut disertakan dengan teknologi JavaServer Faces:

1. UIColumn: Mewakili satu kolom data dalam komponen UIData.
2. UICommand: Mewakili kontrol yang memicu tindakan saat diaktifkan.
3. UIData: Mewakili pengikatan data ke kumpulan data yang diwakili oleh instance javax.faces.model.DataModel.
4. UIForm: Mewakili formulir masukan untuk disajikan kepada pengguna. Komponen turunannya mewakili (antara lain) kolom masukan yang akan disertakan saat formulir diserahkan. Komponen ini dianalogikan dengan tag form pada HTML.

5. UIGraphic: Menampilkan gambar.
6. UIInput: Mengambil input data dari pengguna. Kelas ini adalah subkelas dari UIOutput.
7. UIMessage: Menampilkan pesan kesalahan lokal.
8. UIMessages: Menampilkan serangkaian pesan kesalahan yang dilokalkan.
9. UIOutcomeTarget: Menampilkan hyperlink berupa link atau tombol.
10. UIOutput: Menampilkan keluaran data pada suatu halaman.
11. UIPanel: Mengelola tata letak komponen turunannya.

Selain memperluas UIComponentBase, kelas komponen juga mengimplementasikan satu atau lebih antarmuka perilaku, yang masing-masing mendefinisikan perilaku tertentu untuk sekumpulan komponen yang kelasnya mengimplementasikan antarmuka tersebut.
Antarmuka perilaku ini, semuanya didefinisikan dalam paket javax.faces.component kecuali dinyatakan lain, adalah sebagai berikut:

1. EditableValueHolder: Memperluas ValueHolder dan menentukan fitur tambahan untuk komponen yang dapat diedit, seperti validasi dan mengeluarkan peristiwa perubahan nilai.
2. NamingContainer: Mengamanatkan bahwa setiap komponen yang di-root pada komponen ini memiliki ID unik.
3. StateHolder: Menunjukkan bahwa suatu komponen memiliki status yang harus disimpan di antara permintaan.
4. ValueHolder: Menunjukkan bahwa komponen mempertahankan nilai lokal serta opsi untuk mengakses data di tingkat model.

UICommand mengimplementasikan ActionSource2 dan StateHolder. UIOutput dan kelas komponen yang memperluas UIOutput mengimplementasikan StateHolder dan ValueHolder. UIInput dan kelas komponen yang memperluas UIInput mengimplementasikan EditableValueHolder, StateHolder, dan

ValueHolder. UIComponentBase mengimplementasikan StateHolder.

Hanya penulis komponen yang perlu menggunakan kelas komponen dan antarmuka perilaku secara langsung. Penulis halaman dan pengembang aplikasi akan menggunakan komponen standar dengan menyertakan tag yang mewakilinya pada halaman. Sebagian besar komponen dapat dirender dengan cara berbeda pada suatu halaman. Misalnya, komponen UICommand dapat dirender sebagai tombol atau hyperlink.

Bagian berikutnya menjelaskan cara kerja model rendering dan bagaimana penulis halaman dapat memilih untuk merender komponen dengan memilih tag yang sesuai.

2.4.2 Model Rendering Komponen

Arsitektur komponen JavaServer Faces dirancang sedemikian rupa sehingga fungsionalitas komponen ditentukan oleh kelas komponen, sedangkan rendering komponen dapat ditentukan oleh kelas penyaji terpisah. Desain ini memiliki beberapa keunggulan, antara lain sebagai berikut:

1. Penulis komponen dapat mendefinisikan perilaku komponen satu kali, namun membuat beberapa penyaji, yang masing-masing mendefinisikan cara berbeda untuk merender komponen ke klien yang sama atau ke klien berbeda.
2. Penulis halaman dan pengembang aplikasi dapat mengubah tampilan komponen pada halaman dengan memilih tag yang mewakili kombinasi komponen dan penyaji yang sesuai.

Kit render menentukan bagaimana kelas komponen dipetakan ke tag komponen yang sesuai untuk klien tertentu. Implementasi JavaServer Faces menyertakan kit render HTML standar untuk dirender ke klien HTML.

Kit render mendefinisikan sekumpulan kelas javax.faces.render.Renderer untuk setiap komponen yang didukungnya. Setiap kelas Renderer mendefinisikan cara berbeda untuk merender komponen tertentu ke output yang ditentukan oleh kit render. Misalnya, komponen UISelectOne memiliki tiga penyaji berbeda. Salah satunya menjadikan

komponen sebagai sekumpulan tombol radio. Yang lain menjadikan komponen sebagai kotak kombo. Yang ketiga menjadikan komponen sebagai kotak daftar. Demikian pula, komponen UICommand dapat dirender sebagai tombol atau hyperlink, menggunakan tag h:commandButton atau h:commandLink. Bagian perintah dari setiap tag sesuai dengan kelas UICommand, yang menentukan fungsinya, yaitu untuk mengaktifkan suatu tindakan. Bagian Tombol atau Tautan dari setiap tag berhubungan dengan kelas Renderer terpisah yang menentukan bagaimana komponen muncul di halaman.

Setiap tag khusus yang ditentukan dalam kit render HTML standar terdiri dari fungsionalitas komponen (didefinisikan di kelas UIComponent) dan atribut rendering (didefinisikan oleh kelas Renderer).

2.4.3 Model Konversi

Aplikasi JavaServer Faces secara opsional dapat mengaitkan komponen dengan data objek sisi server. Objek ini adalah komponen JavaBeans, seperti kacang yang dikelola. Aplikasi mendapatkan dan menyetel data objek untuk suatu komponen dengan memanggil properti objek yang sesuai untuk komponen tersebut. Ketika sebuah komponen terikat pada sebuah objek, aplikasi memiliki dua tampilan data komponen:

1. Tampilan model, di mana data direpresentasikan sebagai tipe data, seperti int atau long.
2. Tampilan presentasi, dimana data direpresentasikan dengan cara yang dapat dibaca atau dimodifikasi oleh pengguna. Misalnya, java.util.Date mungkin direpresentasikan sebagai string teks dalam format mm/dd/yy atau sebagai kumpulan tiga string teks.

Implementasi JavaServer Faces secara otomatis mengkonversi data komponen antara dua tampilan ini ketika properti bean yang terkait dengan komponen tersebut adalah salah satu tipe yang didukung oleh data komponen. Misalnya, jika komponen UISelectBoolean dikaitkan dengan properti bean bertipe java.lang.Boolean, implementasi JavaServer Faces akan secara otomatis mengkonversi data komponen dari String ke

Boolean. Selain itu, beberapa data komponen harus terikat pada properti tipe tertentu. Misalnya, komponen UISelectBoolean harus terikat pada properti bertipe boolean atau java.lang.Boolean.

Terkadang Anda mungkin ingin mengonversi data komponen ke tipe selain tipe standar, atau Anda mungkin ingin mengonversi format data. Untuk memfasilitasi hal ini, teknologi JavaServer Faces memungkinkan Anda mendaftarkan implementasi javax.faces.convert.Converter pada komponen UIOutput dan komponen yang kelasnya merupakan subkelas UIOutput. Jika Anda mendaftarkan implementasi Konverter pada sebuah komponen, implementasi Konverter akan mengkonversi data komponen di antara dua tampilan.

2.4.4 Event and Listener Model

Model kejadian dan pendengar JavaServer Faces mirip dengan model kejadian JavaBeans yang memiliki kelas kejadian dan antarmuka pendengar yang sangat diketik sehingga aplikasi dapat digunakan untuk menangani kejadian yang dihasilkan oleh komponen. Spesifikasi JavaServer Faces mendefinisikan tiga jenis kejadian: kejadian aplikasi, kejadian sistem, dan kejadian model data.

Peristiwa aplikasi terikat pada aplikasi tertentu dan dihasilkan oleh UIComponent. Mereka mewakili peristiwa standar yang tersedia di versi sebelumnya dari teknologi JavaServer Faces.

Objek peristiwa mengidentifikasi komponen yang menghasilkan peristiwa dan menyimpan informasi tentang peristiwa tersebut. Agar dapat diberitahu mengenai suatu peristiwa, aplikasi harus menyediakan implementasi kelas pendengar dan harus mendaftarkannya pada komponen yang menghasilkan peristiwa tersebut. Saat pengguna mengaktifkan komponen, misalnya dengan mengklik tombol, suatu peristiwa akan diaktifkan. Hal ini menyebabkan implementasi JavaServer Faces memanggil metode pendengar yang memproses peristiwa tersebut.

JavaServer Faces mendukung dua jenis kejadian aplikasi: kejadian tindakan dan kejadian perubahan nilai.

An **action event** (kelas javax.faces.event.ActionEvent) terjadi ketika pengguna mengaktifkan komponen yang mengimplementasikan javax.faces.component.ActionSource. Komponen ini mencakup tombol dan hyperlink.

A **value-change event** (kelas javax.faces.event.Value ChangeEvent) terjadi ketika pengguna mengubah nilai komponen yang diwakili oleh UIInput atau salah satu subkelasnya. Contohnya adalah memilih kotak centang, suatu tindakan yang mengakibatkan nilai komponen berubah menjadi true. Tipe komponen yang dapat menghasilkan tipe kejadian ini adalah komponen UIInput, UISelectOne, UISelectMany, dan UISelectBoolean. Peristiwa perubahan nilai dipicu hanya jika tidak ada kesalahan validasi yang terdeteksi.

Bergantung pada nilai properti langsung (lihat Atribut langsung) dari komponen yang memancarkan peristiwa, peristiwa tindakan dapat diproses selama fase aplikasi pemanggilan atau fase penerapan nilai permintaan, dan peristiwa perubahan nilai dapat diproses selama validasi proses. fase atau fase penerapan nilai permintaan. Peristiwa sistem dihasilkan oleh Objek, bukan UIComponent. Mereka dihasilkan selama eksekusi aplikasi pada waktu yang telah ditentukan. Mereka berlaku untuk keseluruhan aplikasi dan bukan pada komponen tertentu. Peristiwa model data terjadi ketika baris baru komponen UIData dipilih.

Ada dua cara untuk menyebabkan aplikasi Anda bereaksi terhadap peristiwa tindakan atau peristiwa perubahan nilai yang dikeluarkan oleh komponen standar:

1. Implementasikan kelas pendengar peristiwa untuk menangani peristiwa tersebut dan mendaftarkan pendengar pada komponen dengan menyarangkan tag f:valueChangeListener atau tag f:actionListener di dalam tag komponen.
2. Menerapkan metode kacang yang dikelola untuk menangani peristiwa tersebut dan merujuk ke metode tersebut dengan ekspresi metode dari atribut yang sesuai dari tag komponen.

2.4.5 Model Validasi

Teknologi JavaServer Faces mendukung mekanisme untuk memvalidasi data lokal dari komponen yang dapat diedit (seperti kolom teks). Validasi ini terjadi sebelum data model terkait diperbarui agar sesuai dengan nilai lokal. Seperti model konversi, model validasi menentukan sekumpulan kelas standar untuk melakukan pemeriksaan validasi data umum. Pustaka tag inti JavaServer Faces juga mendefinisikan sekumpulan tag yang sesuai dengan implementasi javax.faces.validator.Validator standar. Lihat Menggunakan Validator Standar untuk daftar semua kelas validasi standar dan tag terkait.

Sebagian besar tag memiliki sekumpulan atribut untuk mengonfigurasi properti validator, seperti nilai minimum dan maksimum yang diperbolehkan untuk data komponen. Penulis halaman mendaftarkan validator pada sebuah komponen dengan menyarangkan tag validator di dalam tag komponen. Selain validator yang terdaftar pada komponen, Anda dapat mendeklarasikan validator default yang terdaftar pada seluruh komponen UIInput dalam aplikasi. Untuk informasi selengkapnya tentang validator default, lihat Menggunakan Validator Default.

Model validasi juga memungkinkan Anda membuat validator khusus Anda sendiri dan tag yang sesuai untuk melakukan validasi khusus. Model validasi menyediakan dua cara untuk mengimplementasikan validasi kustom:

1. Menerapkan antarmuka Validator yang melakukan validasi.
2. Menerapkan metode kacang terkelola yang melakukan validasi.

Jika Anda mengimplementasikan antarmuka Validator, Anda juga harus:

1. Daftarkan implementasi Validator dengan aplikasi.
2. Buat tag khusus atau gunakan tag f:validator untuk mendaftarkan validator pada komponen.

2.4.6 Model Navigasi

Model navigasi JavaServer Faces memudahkan untuk menentukan navigasi halaman dan menangani pemrosesan

tambahan apa pun yang diperlukan untuk memilih urutan pemuatan halaman. Dalam teknologi JavaServer Faces, navigasi adalah seperangkat aturan untuk memilih halaman atau tampilan berikutnya yang akan ditampilkan setelah tindakan aplikasi, seperti ketika tombol atau hyperlink diklik.

Navigasi bisa implisit atau ditentukan pengguna. Navigasi implisit mulai berlaku ketika aturan navigasi yang ditentukan pengguna tidak tersedia. Untuk informasi selengkapnya tentang navigasi implisit, lihat Aturan Navigasi Implisit.

Aturan navigasi yang ditentukan pengguna dideklarasikan dalam nol atau lebih file sumber daya konfigurasi aplikasi, seperti face-config.xml, dengan menggunakan sekumpulan elemen XML. Struktur default aturan navigasi adalah sebagai berikut:

```
<navigation-rule>
  <description></description
  <from-view-id></from-view-id>
  <navigation-case>
    <from-action></from-action>
    <from-outcome></from-outcome>
    <if></if>
    <to-view-id></to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigasi yang ditentukan pengguna ditangani sebagai berikut, menentukan aturan dalam file sumber daya konfigurasi aplikasi. Merujuk ke String hasil dari atribut tindakan komponen tombol atau hyperlink. String hasil ini digunakan oleh implementasi JavaServer Faces untuk memilih aturan navigasi.

2.5 Android Manifest

AndroidManifest.xml adalah file canggih di platform Android yang memungkinkan Anda menggambarkan fungsionalitas dan persyaratan aplikasi Anda ke Android. Namun, bekerja dengan itu tidak mudah. Xamarin.Android membantu meminimalkan kesulitan ini dengan memungkinkan Anda menambahkan atribut kustom ke kelas Anda, yang

kemudian akan digunakan untuk menghasilkan manifes secara otomatis untuk Anda. Tujuan kami adalah bahwa 99% pengguna kami seharusnya tidak perlu memodifikasi AndroidManifest.xmlsecara manual.

AndroidManifest.xml dihasilkan sebagai bagian dari proses build, dan XML yang ditemukan dalam Properti/AndroidManifest.xml digabungkan dengan XML yang dihasilkan dari atribut kustom. gabungan yang dihasilkanAndroidManifest.xml berada di subdirektori obj; misalnya, ia berada di obj/Debug/android/AndroidManifest.xml untuk build Debug. Proses penggabungan sepele: menggunakan atribut kustom dalam kode untuk menghasilkan elemen XML, dan menyisipkan elemen tersebut ke dalamAndroidManifest.xml.

Pada waktu kompilasi, rakitan dipindai untuk non-kelasabstract yang berasal dari Aktivitas dan memiliki [Activity] atribut yang dideklarasikan padanya. Kemudian menggunakan kelas dan atribut ini untuk membangun manifes. Sebagai contoh, perhatikan kode berikut:

```
namespace Demo
{
  public class MyActivity : Activity
  {
  }
}
```

Ini tidak menghasilkan apa pun yang dihasilkan dalam AndroidManifest.xml. Jika Anda ingin <activity/> elemen dibuat, Anda perlu menggunakan[Activity] atribut kustom:

```
namespace Demo
{
  [Activity]
  public class MyActivity : Activity
  {
  }
}
```

Contoh ini menyebabkan fragmen xml berikut ditambahkan ke AndroidManifest.xml:

```
<activity
android:name="md5a7a3c803e481ad8926683588c7e9031b.MainActivity" />
```
Atribut [Activity] tidak berpengaruh pada abstract jenis; abstract jenis diabaikan.

2.5.1 Activity

Dimulai dengan Xamarin.Android 5.1, nama jenis aktivitas didasarkan pada MD5SUM dari nama yang memenuhi syarat perakitan dari jenis yang diekspor. Ini memungkinkan nama yang sepenuhnya memenuhi syarat yang sama disediakan dari dua rakitan yang berbeda dan tidak mendapatkan kesalahan pengemasan. (Sebelum Xamarin.Android 5.1, nama jenis default aktivitas dibuat dari namespace layanan huruf kecil dan nama kelas.)

Jika Anda ingin mengambil alih default ini dan secara eksplisit menentukan nama aktivitas Anda, gunakan Name properti :

```
[Activity (Name="awesome.demo.activity")]
public class MyActivity : Activity
{
}
```
Hasil dari contoh diatas adalah :
```
<activity android:name="awesome.demo.activity" />
```

Selanjutnya adalah untuk judul dari aktivitas. Secara default, Android memberi aplikasi Anda bilah judul saat dijalankan. Nilai yang digunakan untuk ini adalah /manifest/application/activity/@android:label. Dalam kebanyakan kasus, nilai ini akan berbeda dari nama kelas Anda. Untuk menentukan label aplikasi Anda pada bilah judul, gunakan Label properti . Contohnya:

```
[Activity (Label="Awesome Demo App")]
public class MyActivity : Activity
{
}
```
Hasilnya :
```
<activity android:label="Awesome Demo App"
```

```
android:name="md5a7a3c803e481ad8926683588c7e9031b.MainActivity" />
```

Selanjutnya adalah meluncurkan dari pemilih aplikasi. Aktivitas Anda tidak akan muncul di layar peluncur aplikasi Android. Ini karena kemungkinan akan ada banyak aktivitas dalam aplikasi Anda, dan Anda tidak ingin ikon untuk setiap aktivitas. Untuk menentukan mana yang harus dapat diluncurkan dari peluncur aplikasi, gunakan MainLauncher properti . Contohnya:

```
[Activity (Label="Awesome Demo App", MainLauncher=true)]
public class MyActivity : Activity
{
}
```

Hasilnya :

```
<activity android:label="Awesome Demo App"

android:name="md5a7a3c803e481ad8926683588c7e9031b.MainActivity">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category
android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Kemudian untuk ikon aktivitas. aktivitas Anda akan diberikan ikon peluncur default yang disediakan oleh sistem. Untuk menggunakan ikon kustom, pertama-tama tambahkan .png Anda ke Resources/drawable, atur Build Action ke AndroidResource, lalu gunakan Icon properti untuk menentukan ikon yang akan digunakan. Contohnya:

```
[Activity (Label="Awesome Demo App", MainLauncher=true, Icon="@drawable/myicon")]
public class MyActivity : Activity
{
}
```

Hasilnya :

```
<activity android:icon="@drawable/myicon"
android:label="Awesome Demo App"

android:name="md5a7a3c803e481ad8926683588c7e9031b.MainActivity">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category
android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

2.5.2 License & Aplication Element

Saat Anda menambahkan izin ke Manifes Android (seperti yang dijelaskan dalam Tambahkan Izin ke Manifes Android), izin ini direkam di Properties/AndroidManifest.xml. Misalnya, jika Anda mengatur INTERNET izin, elemen berikut ditambahkan ke Properti/AndroidManifest.xml:

```
<uses-permission
android:name="android.permission.INTERNET" />
```

Build debug secara otomatis mengatur beberapa izin untuk mempermudah debug (seperti INTERNET dan READ_EXTERNAL_STORAGE) - pengaturan ini hanya diatur dalam obj/Debug/android/AndroidManifest.xml yang dihasilkan dan tidak ditampilkan sebagai diaktifkan dalam pengaturan Izin yang diperlukan .

Manifes Android juga menyediakan cara bagi Anda untuk mendeklarasikan properti untuk seluruh aplikasi Anda. Ini dilakukan melalui <application> elemen dan mitranya, atribut kustom Aplikasi . Perhatikan bahwa ini adalah pengaturan di seluruh aplikasi (seluruh perakitan) daripada pengaturan per Aktivitas. Biasanya, Anda mendeklarasikan <application> properti untuk seluruh aplikasi Anda dan kemudian mengambil alih pengaturan ini (sesuai kebutuhan) berdasarkan per Aktivitas.

Misalnya, atribut berikut Application ditambahkan ke AssemblyInfo.cs untuk menunjukkan bahwa aplikasi dapat di-

debug, bahwa nama yang dapat dibaca pengguna adalah Aplikasi Saya, dan menggunakan Theme.Light gaya sebagai tema default untuk semua aktivitas:

```
[assembly: Application (Debuggable=true,
                Label="My App",
               Theme="@android:style/Theme.Light")]
```

Deklarasi ini menyebabkan fragmen XML berikut dihasilkan dalam obj/Debug/android/AndroidManifest.xml:

```
<application android:label="My App"
        android:debuggable="true"
        android:theme="@android:style/Theme.Light"
          ... />
```

Dalam contoh ini, semua aktivitas di aplikasi akan default ke Theme.Light gaya. Jika Anda mengatur tema Aktivitas ke Theme.Dialog, hanya Aktivitas yang akan menggunakan Theme.Dialog gaya sementara semua aktivitas lain di aplikasi Anda akan default ke Theme.Light gaya sebagaimana diatur dalam <application> elemen .

Elemen Application ini bukan satu-satunya cara untuk mengonfigurasi <application> atribut. Secara bergantian, Anda dapat menyisipkan atribut langsung ke <application> elemen Properties/AndroidManifest.xml. Pengaturan ini digabungkan ke dalam elemen akhir <application> yang berada di obj/Debug/android/AndroidManifest.xml. Perhatikan bahwa konten Properti/AndroidManifest.xml selalu mengambil alih data yang disediakan oleh atribut kustom.

DAFTAR PUSTAKA

Daniel, L, Y 2011, *Introduction to java programming*, Pearson Education, New Jersey.

Horton, J 2021, Android Programming for Beginners: Build in-depth, full-featured Android apps starting from zero programming experience, Packt Publishing, Birmingham.

Meike, GB, & Schiefer, L 2021, *Inside the android OS: building, customizing, managing and operating android system services,* Pearson Education, London.

BAB 3
KONSEP ACTIVITY DAN INTENT

Oleh Irmawati

3.1 Dasar-dasar Activity

Dalam konteks pengembangan aplikasi mobile, "*Activity*" merupakan sebuah komponen fundamental dalam platform Android. Secara esensial, *Activity* adalah representasi dari satu layar atau antarmuka pengguna (UI) yang dapat diakses oleh pengguna aplikasi (Hardy and Phillips, 2013). Misalnya, ketika pengguna membuka aplikasi pesan singkat, daftar pesan dapat diwakili oleh satu *Activity,* dan saat mereka membuka pesan tertentu, tampilan pesan penuh dapat diwakili oleh Activity yang berbeda.

Setiap *Activity* dalam Android dapat berisi elemen-elemen UI seperti tombol, teks, input, dan lainnya, yang membentuk tampilan yang dapat diakses oleh pengguna. Pentingnya *Activity* juga terletak pada kemampuannya untuk mengelola siklus hidupnya sendiri. Siklus hidup *Activity* mencakup berbagai peristiwa, seperti pembuatan (*creation*), memulai (*start*), menjeda (*pause*), melanjutkan (*resume*), menghentikan (*stop*), dan menghancurkan (*destroy*) (Meier, 2012).

"Mengelola Layout pada *Activity*" merupakan aspek penting dalam pengembangan aplikasi mobile, khususnya pada platform Android. Hal ini berkaitan dengan penataan dan pengaturan elemen antarmuka pengguna (UI) di dalam suatu Activity agar menciptakan tampilan yang nyaman dan efektif bagi pengguna. Dalam konteks Android, layout sering didefinisikan menggunakan XML, yang memungkinkan pengembang untuk mendeklarasikan struktur dan properti visual elemen-elemen UI. Membantu dalam pemisahan antara tata letak UI dan logika bisnis aplikasi (Clifton, 2015).

"Siklus Hidup *Activity*" pada platform Android merujuk pada serangkaian peristiwa yang terjadi selama masa hidup

suatu *Activity* dalam pengembangan aplikasi. Siklus hidup ini terdiri dari beberapa tahapan yang memungkinkan pengembang mengontrol perilaku *Activity* sepanjang waktu eksekusi aplikasi. Pemahaman yang baik tentang siklus hidup *Activity* penting untuk mengelola sumber daya dan merespons peristiwa aplikasi dengan benar (Griffiths and Griffiths, 2017).

Berikut adalah beberapa tahapan utama dalam siklus hidup *Activity*:

1. onCreate(): Dipanggil saat *Activity* pertama kali dibuat, digunakan untuk inisialisasi awal.
2. onStart(): Dipanggil saat *Activity* menjadi terlihat oleh pengguna, tetapi belum aktif.
3. onResume(): Dipanggil saat *Activity* aktif dan fokus. Ini adalah tahap di mana interaksi pengguna diperbolehkan.
4. onPause(): Dipanggil ketika *Activity* kehilangan fokus tetapi masih terlihat. Pada tahap ini, pengembang dapat menyimpan perubahan yang dibuat oleh pengguna.
5. onStop(): Dipanggil ketika Activity tidak lagi terlihat oleh pengguna. Pada tahap ini, pengembang dapat melepaskan sumber daya yang tidak diperlukan.
6. onDestroy(): Dipanggil saat *Activity* dihancurkan atau ditutup. Digunakan untuk membersihkan sumber daya dan melakukan tindakan akhir.

3.2 Intent

Intent dalam pengembangan aplikasi Android adalah konsep yang sangat penting yang digunakan untuk memfasilitasi komunikasi antara komponen aplikasi. Intent memungkinkan aplikasi untuk memulai komponen baru, seperti memulai *Activity*, memulai layanan (Service), atau mengirim siaran (Broadcast) ke aplikasi lain (Burd and Mueller, 2020).

Ada dua jenis utama Intent:

1. *Explicit Intent*: Spesifik dalam menentukan komponen yang akan diaktifkan, seperti memulai *Activity* tertentu.
2. *Implicit Intent*: Tidak merinci komponen secara spesifik. Sebaliknya, menyatakan tindakan yang ingin dilakukan, dan

sistem Android akan menentukan komponen yang paling cocok untuk menangani intent tersebut.

Dengan pemahaman yang kuat tentang Intent, pengembang dapat membuat aplikasi yang lebih dinamis dan terhubung dengan baik antar komponen-komponennya. Konsep ini memainkan peran kunci dalam memastikan interaktivitas dan integrasi yang baik dalam pengalaman pengguna Android.

"Mengirim Data melalui Intent" adalah proses dimana aplikasi Android dapat mengirim informasi atau data dari satu *Activity* ke *Activity* lainnya menggunakan objek Intent. Memungkinkan komunikasi dan pertukaran data antar komponen aplikasi. Pengiriman data melalui Intent memungkinkan Activity yang akan dimulai untuk menerima data dan berinteraksi sesuai dengan informasi yang diterima.

Ada beberapa metode untuk mengirim data melalui Intent, tergantung pada jenis data yang dikirimkan. Beberapa metode umumnya melibatkan:

1. Menambahkan Data ke Intent:
   Menggunakan metode putExtra() pada objek Intent untuk menambahkan data ekstra, seperti string, angka, objek Parcelable, dsb.

```
Intent intent = new Intent(this, TujuanActivity.class);
intent.putExtra("KEY_NAMA", "Nilai String");
intent.putExtra("KEY_ANGKA", 123);
startActivity(intent);
```

2. Mengirim Objek Parcelable:
   Jika data yang dikirim adalah objek kompleks, objek tersebut perlu mengimplementasikan antarmuka Parcelable. Kemudian, objek tersebut dapat dikirim sebagai ekstra Intent.

```
Intent intent = new Intent(this, TujuanActivity.class);
ParcelableObjek objekParcelable = new ParcelableObjek();
intent.putExtra("KEY_OBJEK_PARCELABLE",
objekParcelable);
startActivity(intent);
```

3. Menggunakan Bundle:
   Mengelompokkan beberapa data ke dalam objek Bundle, yang kemudian disertakan dalam Intent.

```
Intent intent = new Intent(this, TujuanActivity.class);
Bundle bundle = new Bundle();
bundle.putString("KEY_NAMA", "Nilai String");
bundle.putInt("KEY_ANGKA", 123);
intent.putExtras(bundle);
startActivity(intent);
```

Ketika *Activity* tujuan menerima Intent, ia dapat mengekstrak data dengan menggunakan metode sesuai dengan jenis data yang dikirimkan.

Penting untuk menjaga konsistensi kunci (*key*) antara pengirim dan penerima untuk memastikan data dikirim dan diterima dengan benar.

3.3 Navigasi Antara *Activity*

Navigasi Antara *Activity* adalah konsep kunci dalam pengembangan aplikasi Android yang berkaitan dengan berpindah dari satu layar ke layar lainnya. Proses navigasi yang baik memastikan pengalaman pengguna yang lancar dan terstruktur, memungkinkan mereka untuk berinteraksi dengan aplikasi secara intuitif. Dalam pengembangan Android, navigasi ini umumnya dicapai menggunakan Intent untuk memulai *Activity* baru.

Berikut adalah beberapa aspek terkait dan metode yang digunakan dalam navigasi antara *Activity*:

1. Intent untuk Navigasi:
   Pengembang menggunakan objek Intent untuk memulai Activity baru. Intent dapat membawa data tambahan atau parameter yang dibutuhkan oleh Activity tujuan.

```
Intent intent = new Intent(this, TujuanActivity.class);
startActivity(intent);
```

2. Back Stack:
   Sistem Android menyimpan daftar Activity yang dikunjungi oleh pengguna dalam suatu urutan tumpukan (stack).

Pengguna dapat kembali ke Activity sebelumnya dengan menekan tombol "Back".

3. Menangani Kembali (Back) dengan finish():
   Activity saat ini dapat menutup dirinya sendiri dengan memanggil metode finish(). Bermanfaat ketika Activity tidak lagi diperlukan setelah suatu tindakan selesai.

```
// Pada suatu tindakan selesai, tutup Activity saat ini
finish();
```

Navigasi yang baik antara *Activity* merupakan bagian penting dari desain aplikasi yang efektif dan dapat meningkatkan kenyamanan serta pengalaman pengguna.

3.4 Komunikasi antar Activity

Komunikasi antar *Activity* merupakan aspek penting dalam pengembangan aplikasi Android, memungkinkan pertukaran data atau informasi antara berbagai layar atau Activity. Tujuan dari komunikasi ini adalah untuk mencapai integrasi yang baik antar komponen aplikasi dan memastikan pengalaman pengguna yang mulus.

Berikut adalah beberapa metode umum yang digunakan untuk mencapai komunikasi antar Activity:

1. Menggunakan Intent:
   Penggunaan Intent memungkinkan pengembang untuk memulai Activity baru dan menyertakan data tambahan. Data tersebut dapat diterima dan diolah oleh Activity tujuan.

```
Intent intent = new Intent(this, TujuanActivity.class);
intent.putExtra("KEY_NAMA", "Nilai String");
intent.putExtra("KEY_ANGKA", 123);
startActivity(intent);
```

2. Menggunakan Bundle:
   Bundle digunakan untuk mengelompokkan data sebelum disertakan dalam Intent. Metode ini berguna ketika ingin mengirim beberapa nilai sekaligus.

```
Intent intent = new Intent(this, TujuanActivity.class);
Bundle bundle = new Bundle();
```

```
bundle.putString("KEY_NAMA", "Nilai String");
bundle.putInt("KEY_ANGKA", 123);
intent.putExtras(bundle);
startActivity(intent);
```

3. Implementasi Interface:
   Pengembang dapat menggunakan antarmuka (interface) untuk mendefinisikan metode yang digunakan untuk mengirim atau menerima data antar Activity.

Menggunakan Bundle untuk Mengemas Data adalah pendekatan yang umum digunakan dalam pengembangan aplikasi Android untuk membawa data tambahan saat berpindah antar Activity. Bundle adalah objek yang berfungsi sebagai wadah untuk menyimpan dan mengirim sejumlah nilai atau data kompleks (Griffiths and Griffiths, 2021).

Berikut adalah langkah-langkah yang umumnya diikuti saat menggunakan Bundle:

1. Menyiapkan Bundle:
   Pertama, objek Bundle dibuat, dan data ditambahkan ke dalamnya menggunakan metode putXxx(). Jenis data (Xxx) dapat berupa String, Int, Boolean, dan lainnya.
   ```
   Bundle bundle = new Bundle();
   bundle.putString("KEY_NAMA", "Nilai String");
   bundle.putInt("KEY_ANGKA", 123);
   ```
2. Menyertakan Bundle dalam Intent:
   Bundle kemudian disertakan dalam Intent sebelum memulai Activity tujuan.
   ```
   Intent intent = new Intent(this, TujuanActivity.class);
   intent.putExtras(bundle);
   startActivity(intent);
   ```
3. Menerima Bundle di Activity Tujuan:
   Di Activity tujuan, data dapat diekstrak dari Bundle menggunakan metode getXxx() yang sesuai.
   ```
   Bundle receivedBundle = getIntent().getExtras();
   if (receivedBundle != null) {
      String nama = receivedBundle.getString("KEY_NAMA");
      int angka = receivedBundle.getInt("KEY_ANGKA");
   ```

```
// Gunakan data sesuai kebutuhan
}
```

Implementasi Komunikasi yang Efisien dalam pengembangan aplikasi Android menjadi krusial untuk memastikan pertukaran data yang responsif dan optimal antar komponen aplikasi, terutama dalam konteks komunikasi antar Activity. Berikut adalah beberapa langkah dan praktik terbaik untuk mengimplementasikan komunikasi yang efisien:

1. Pilih Metode Komunikasi yang Sesuai:
   Pertimbangkan menggunakan metode komunikasi yang paling sesuai dengan kebutuhan aplikasi, seperti Intent, Bundle, antarmuka (*interface*), atau menggunakan arsitektur komponen modern seperti ViewModel dan LiveData.
2. Kendalikan Jumlah Data yang Dikirim:
   Batasi jumlah data yang dikirim antar komponen. Hanya kirim data yang diperlukan untuk meminimalkan beban dan meningkatkan responsivitas aplikasi.
3. Optimalkan Penggunaan Parcelable:
   Jika menggunakan Parcelable untuk mengirim objek kompleks, pastikan untuk mengoptimalkan implementasinya agar tidak membebani kinerja aplikasi secara berlebihan.
4. Gunakan Penanganan Async:
   Saat mengakses atau memproses data dari sumber eksternal, seperti database atau jaringan, implementasikan penanganan asinkronus untuk mencegah pembekuan antarmuka pengguna dan memastikan responsivitas.
5. Pelajari dan Terapkan Desain Berorientasi Objek:
   Implementasikan prinsip desain berorientasi objek untuk memisahkan tanggung jawab antar komponen, sehingga setiap komponen bertanggung jawab atas tugas tertentu.
6. Gunakan Arsitektur Jetpack (*Android Architecture Components*):
   Pertimbangkan penggunaan komponen-komponen arsitektur Jetpack seperti ViewModel, LiveData, dan Data

Binding untuk menyederhanakan manajemen siklus hidup dan komunikasi antar komponen.

7. Uji Kinerja:
   Lakukan uji kinerja secara rutin menggunakan alat seperti Android Profiler untuk menganalisis pemakaian sumber daya dan mengidentifikasi potensi titik bottleneck.

   Implementasi komunikasi yang efisien merupakan faktor penentu keberhasilan aplikasi Android. Dengan menerapkan langkah-langkah ini, pengembang dapat memastikan aplikasi berjalan dengan baik, responsif, dan memberikan pengalaman pengguna yang optimal.

DAFTAR PUSTAKA

Burd, B. and Mueller, J.P. (2020) *Android application development all-in-one for dummies*. John Wiley & Sons.

Clifton, I.G. (2015) *Android user interface design: Implementing material design for developers*. Addison-Wesley Professional.

Griffiths, Dawn and Griffiths, David (2017) *Head first android development: a brain-friendly guide*. ' O'Reilly Media, Inc.'

Griffiths, Dawn and Griffiths, David (2021) *Head First Android Development*. ' O'Reilly Media, Inc.'

Hardy, B. and Phillips, B. (2013) *Android programming: the big nerd ranch guide*. Addison-Wesley Professional.

Meier, R. (2012) *Professional Android 4 application development*. John Wiley & Sons.

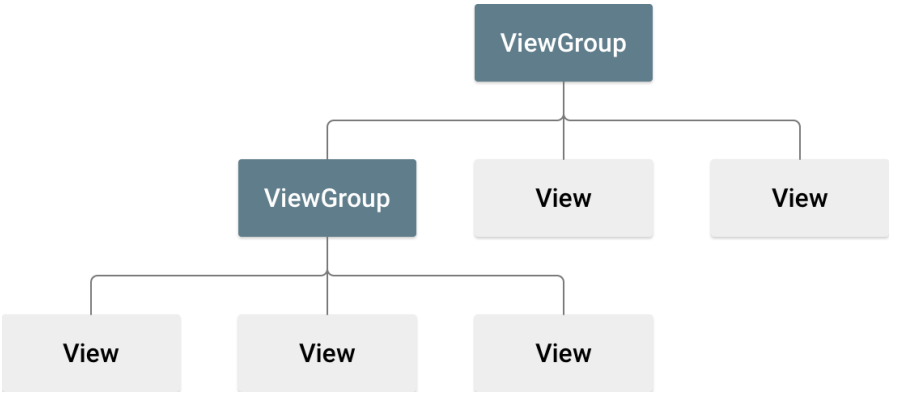
BAB 4
ANDROID LAYOUTS : TAMPILAN DAN GRUP LAYOUT

Oleh Roro Santi

4.1 Pendahuluan

Tampilan (*Layout*) merupakan hal terpenting dalam menampilkan sebuah produk. Tampilan juga bisa menjadi kesan pertama seseorang dalam menilai sesuatu. Apalagi mengenai tampilan sebuah aplikasi. Tampilan menjadi penghubung antara sistem dan pengguna. Tentu saja, harus juga disesuaikan dengan *device* yang menjadi tujuan perancangan apakan untuk *desktop*, *web* atau *mobile*. Tampilan untuk aplikasi *mobile* memiliki beberapa karakteristik yang khas diantaranya yaitu terkait dengan layar yang terbatas, dan juga harus sesuai dengan pengunaan perangkat layar sentuh. Sehingga hal ini sangat mempengaruhi penggunaan komponen layout yang akan diterapkan dalam perancangan aplikasi mobile. Seperti ukuran tombol dan warnanya. Selain penggunaan komponen yang akan ditampilkan di dalam layout android. Struktur tata letak komponen tersebut juga menjadi salah satu yang harus diperhatikan dalam perancangan aplikasi mobile.

Didalam android untuk implementasi layout tersebut dituangkan ke dalam kode-kode XML. Komponen layout di android ada 2 (dua). Hal ini menjadi fokus perancangan layout, yaitu View (tampilan) dan View Group (grup layout). Berikut ini adalah gambar (Gambar 4.1) hierarki tampilan yang menentukan tata letak UI (*User Interface*) yang diilustrasikan oleh developer.android.com (https://developer.android.com).



Gambar 4.1. Hierarki Android Layouts
(Sumber : developer.android.com)

View adalah komponen objek yang dapat dilihat langsung oleh pengguna dan dapat melakukan interaksi langsung melalui komponen tersebut. Contohnya adalah Teks (*TextView*), Gambar (*ImageView*), Tombol (*Button*), dan lain sebagainya.

ViewGroup adalah komponen objek yang tidak terlihat secara langsung oleh pengguna namun menjadi dasar struktur tata letak yang menampung komponen-kompunen layout *(view*) yang akan ditampilkan didalam aplikasi mobile android. Contohnya : LinierLayout, RelativeLayout, FrameLayout, TableLayout, ConstraintLayout, dan lain sebagainya.

4.2 View

Objek view yang sering digunakan dalam layout android, yaitu TextView, Button, ImageView, dan lain sebagainya. Berikut ini adalah contoh implementasi kode dalam XML dan hasil tampilan desainnya.

TextView, contoh kode :

```
        <TextView
    android:id="@+id/textViewHeader"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textStyle="bold"
```

```
   android:textSize="50sp"
   android:textColor="@color/black"
   android:background="@color/kuning"
   android:layout_marginTop="20dp"
   android:text="Roro Santi" />
```

Tampilan desain yang dihasilkan :

Gambar 4.2. Tampilan Text View
(Sumber : Hasil Modul Layout Roro)

Button, contoh kode :

```
<Button
   android:id="@+id/buttonKatalog"
   android:layout_width="200dp"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
   android:onClick="katalog"
   android:backgroundTint="@color/kuning"
   android:textColor="@color/black"
   android:layout_marginTop="10dp"
   android:drawableTop="@drawable/profil"
   android:text="Katalog Produk" />
```

Tampilan desain yang dihasilkan :

Gambar 4.3. Tampilan Button
(Sumber : Hasil Modul Layout Roro)

ImageView

```
<ImageView
   android:id="@+id/imageView2"
   android:layout_width="200dp"
   android:layout_height="200dp"
```

```
    android:background="@color/white"
    android:src="@drawable/logo" />
```

Tampilan desain yang dihasilkan :

Gambar 4.4. Tampilan Image View
(Sumber : Hasil Modul Layout Roro)

4.3 View Group

View Group, selain menampung atau sebagai wadah tata letak view-view juga bisa memuat view group lainnya. Berikut ini adalah beberapa view group dan gambar ilustrasinya.

Ilustrasi Linear Layout Vertical :

Gambar 4.5. Ilustrasi Linear Layout Vertical
(Sumber : developer.android.com)

Implementasi Linear Layout Vertical, contoh kode :

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="@color/black"
   android:orientation="vertical"                                    >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:layout_marginTop="10dp"
      android:text="Linear                                    Layout"
      android:onClick="linear"
      android:textColor="@color/white"
      android:textSize="20sp"
      android:textStyle="bold"                                      />

   <ImageView
      android:id="@+id/imageView"
      android:layout_width="50dp"
      android:layout_height="50dp"
      android:layout_gravity="center"
      app:srcCompat="@android:drawable/ic_menu"                     />

   <Button
      android:id="@+id/button"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:onClick="jadwal"
      android:text="Cek Jadwal" />

</LinearLayout>
```

Tampilan desain yang dihasilkan :

Gambar 4.6. Tampilan Implementasi Linear Layout Vertical
(Sumber : Hasil Modul Layout Roro)

Ilustrasi Linear Layout Horizontal :

Gambar 4.7. Ilustrasi Linear Layout Horizontal
(Sumber : developer.android.com)

Implementasi Linear Layout Horizontal, contoh kode :

```
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:gravity="center"
  android:orientation="horizontal">

  <Button
    android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="75dp"
    android:layout_margin="10dp"
    android:backgroundTint="@color/black"
```

```
    android:onClick="nomor1"
    android:text="1"
    android:textColor="@color/white"                    />

  <Button
    android:id="@+id/button6"
    android:layout_width="wrap_content"
    android:layout_height="75dp"
    android:layout_margin="10dp"
    android:backgroundTint="@color/black"
    android:onClick="nomor2"
    android:text="2"
    android:textColor="@color/white"                    />

  <Button
    android:id="@+id/button7"
    android:layout_width="wrap_content"
    android:layout_height="75dp"
    android:layout_margin="10dp"
    android:onClick="nomor3"
    android:backgroundTint="@color/black"
    android:textColor="@color/white"
    android:text="3"                                    />
</LinearLayout>
```

Tampilan desain yang dihasilkan :

Gambar 4.8. Tampilan Implementasi Linear Layout Horizontal (Sumber : Hasil Modul Layout Roro)

Komponen-komponen layout tersebut dimuat kedalam sebuah *activity* agar dapat ditampilkan. Berikut ini gambar contoh implementasi kode java dalam MainActivity.java :

```java
package com.rorosanti.mylayout;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Gambar 4.9. Kode Java yang Memuat Tampilan XML
(Sumber : Hasil Modul Layout Roro)

4.4 Tutorial Project

Berikut ini adalah contoh Project Layout yang memuat beberapa macam view group di android.

Kode XML (activity_main.xml) :

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView

xmlns:android="http://schemas.android.com/apk/res/android
"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  tools:context=".MainActivity2">

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="5dp"
        android:text="--Roro                                    Santi--"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"                                        />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/black"
        android:orientation="vertical"                                  >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:text="Linear                                Layout"
            android:textColor="@color/white"
            android:textSize="20sp"
            android:textStyle="bold"                                    />

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_gravity="center"
            app:srcCompat="@android:drawable/ic_menu_today"
/>

        <Button
            android:id="@+id/button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
        android:text="Cek                Jadwal"                />
    </LinearLayout>

    <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:padding="5dp"
      android:background="@color/white">

      <TextView
        android:id="@+id/textViewRelative"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Layout                              Relative"
        android:textSize="20sp"
        android:textStyle="bold"                              />

      <ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/imageButton1"
        android:layout_below="@id/textViewRelative"
        app:srcCompat="@android:drawable/ic_menu_gallery"
/>

      <ImageButton
        android:id="@+id/imageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/textViewRelative"
        app:srcCompat="@android:drawable/ic_input_get"    />

      <ImageButton
        android:id="@+id/imageButton2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/imageButton1"
        android:layout_below="@id/textViewRelative"
        app:srcCompat="@android:drawable/ic_input_add"   />

    </RelativeLayout>

    <TableLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@color/biru">

      <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center">

        <TextView
          android:id="@+id/textViewTable"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textSize="20sp"
          android:textStyle="bold"
          android:text="Table                Layout"              />
      </TableRow>

      <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center">

        <Button
          android:id="@+id/button_1"
          android:layout_width="100dp"
          android:layout_height="50dp"
          android:textSize="20sp"
```

```
            android:textStyle="bold"
            android:text="1"                                    />

        <Button
            android:id="@+id/button_2"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="2"                                    />

        <Button
            android:id="@+id/button_3"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="3"                                    />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center">

        <Button
            android:id="@+id/button_4"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="4"                                    />

        <Button
            android:id="@+id/button_5"
            android:layout_width="100dp"
            android:layout_height="50dp"
```

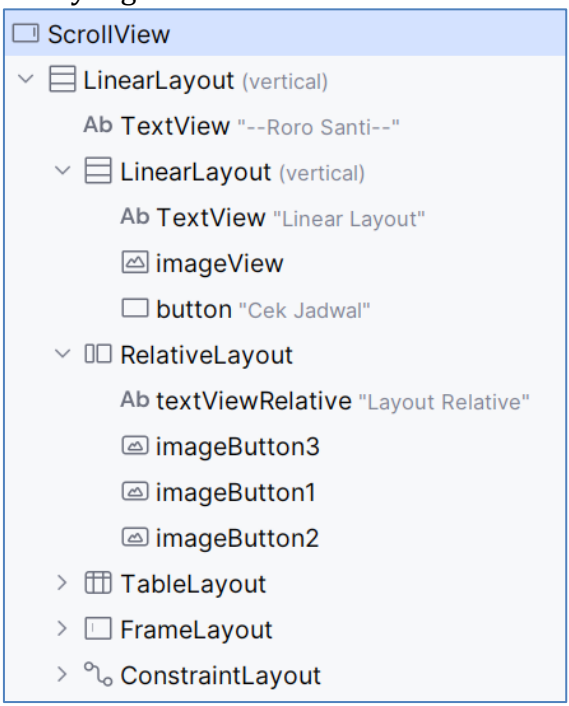
```
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="5" />

        <Button
            android:id="@+id/button_6"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="6" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center">

        <Button
            android:id="@+id/button_7"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="7" />

        <Button
            android:id="@+id/button_8"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="8" />

        <Button
            android:id="@+id/button_9"
            android:layout_width="100dp"
```

```
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="9" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center">

        <Button
            android:id="@+id/button_0"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="0" />

    </TableRow>
</TableLayout>

<FrameLayout
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_gravity="center"
    android:background="@color/black">

    <FrameLayout
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_gravity="center"
        android:background="@color/white">

        <FrameLayout
            android:layout_width="100dp"
            android:layout_height="100dp"
```

```
            android:layout_gravity="center"
            android:background="@color/biru">

            <TextView
              android:id="@+id/textView"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="center"
              android:text="Frame                Layout"              />

        </FrameLayout>
      </FrameLayout>
    </FrameLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
      android:layout_width="match_parent"
      android:layout_height="100dp"
      android:background="@color/biru">

      <TextView
        android:id="@+id/textViewConstraint"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Constraint                              Layout"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintVertical_bias="0.500"
        app:layout_constraintHorizontal_bias="0.500"/>

    </androidx.constraintlayout.widget.ConstraintLayout>

  </LinearLayout>
```

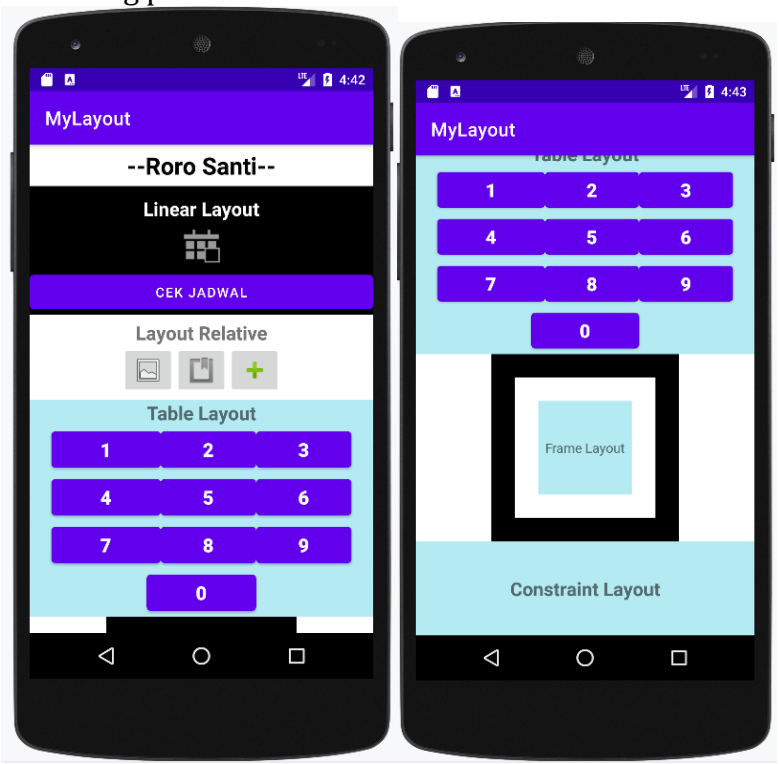
```
</ScrollView >
```

Komponen Tree yang dihasilkan :



Gambar 4.8. Komponen Tree
(Sumber : Hasil Modul Layout Roro)

Hasil Running pada virtual device :



Gambar 4.9. Hasil Running (Tampilan di Scroll ke bawah) (Sumber : Hasil Modul Layout Roro)

DAFTAR PUSTAKA

Google for Developers. Mengembangkan UI untuk Android. https://developer.android.com/develop/ui?hl=id

Santi, R. Modul Mobile Programming. Prodi MI Politeknik LP3I Bandung.

BAB 5
ANDROID WIDGET DAN STYLING

Oleh Monika Evelin Johan

5.1 Mengenal Widget di Android

5.1.1 Pengenalan Apa Itu Widget

Widget adalah elemen penyusun yang digunakan untuk membuat antarmuka pengguna (*user interface*) (Phillips et al., 2019). Widget dapat digunakan untuk menampilkan teks atau gambar, dibuat untuk berinteraksi dengan pengguna, atau mengatur widget lain di layar.

Pada Android SDK (*software development kit*) terdapat beberapa widget yang dapat ditambahkan dan dikonfigurasi untuk membuat tampilan yang kita inginkan. Setiap widget adalah turunan dari kelas View, atau salah satu dari sub-kelasnya (seperti TextView atau Button).

5.1.2 Jenis – Jenis Widget

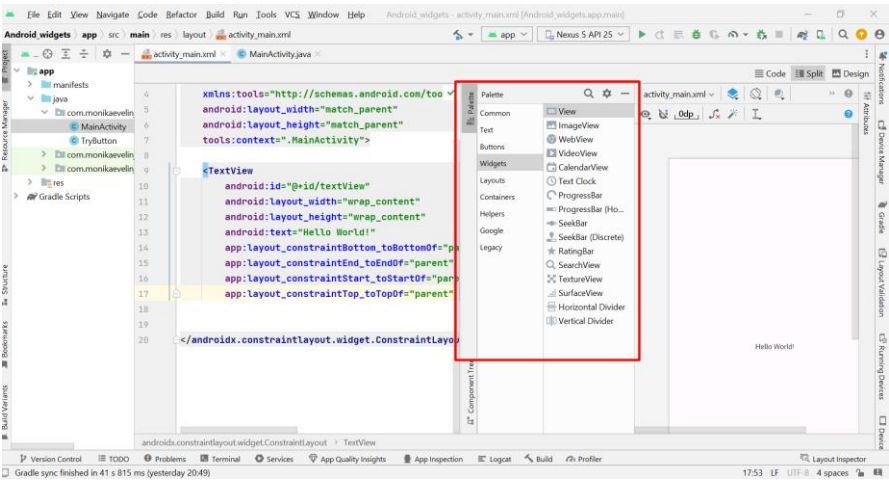
Terdapat banyak widget Android yang dapat ditambahkan pada antarmuka aplikasi Android, seperti tombol (*button*), teks (*TextView*), gambar (*ImageView*), kolom pencarian (*SearchBar*), dan masih banyak lagi. Terdapat widget yang disebut built-in widget, yaitu widget yang sudah tersedia pada Android Studio termasuk dengan pengaturan dasar sehingga mudah untuk digunakan dan dimodifikasi. Selain itu, terdapat juga custom widget, yaitu widget yang dapat kita kustomisasi dan konfigurasi sesuai dengan keinginan dengan lebih fleksibel. Pembahasan dan contoh penggunaan jenis – jenis widget ini dapat dilihat pada sub-bab berikutnya.

5.2 Menggunakan Built-in Widget

5.2.1 Widget – Widget Bawaan Android

Seperti telah disebutkan sebelumnya, built-in widget adalah beberapa widget bawaan yang telah tersedia pada

Android Studio (software untuk mengembangkan aplikasi Android). Tersedia banyak tipe widget yang umum digunakan pada aplikasi Android, seperti untuk teks, gambar, tombol, dan sebagainya. Kita dapat mengakses widget tersebut dengan membuka menu Palette pada Android Studio, seperti pada Gambar 5.1. Menu Palette untuk Menambahkan Widget berikut.



Gambar 5.1. Menu Palette untuk Menambahkan Widget

Pada Android Studio, terdapat kategori utama widget, diantaranya :

1. **Common** (berisi widget yang umum digunakan pada aplikasi Android, seperti TextView, Button, ImageView, RecyclerView, FragmentContainerView, ScrollView, dan Switch).
2. **Text** (berisi beberapa tipe untuk input teks seperti TextView, PlainText, Password, E-mail, Phone, Time, Date, Number, AutoCompleteTextView, dan sebagainya). Dengan menggunakan widget teks ini, kita dapat memilih tipe input yang diinginkan dan konfigurasi otomatis akan mengikuti sifat dari input tersebut. Sebagai contoh, jika menggunakan tipe Password, maka teks yang akan diinput user akan otomatis tersembunyi sebagaimana biasanya pengguna memasukkan password ketika log in ke suatu aplikasi. Contoh lainnya adalah dengan menggunakan tipe E-mail, maka ketika pengguna input teks akan otomatis divalidasi

apakah sudah sesuai dengan format email (xx@xx.xx), begitu pun jika menggunakan tipe Date, maka input akan otomatis berupa format input tanggal (DD/MM/YYYY).

3. **Buttons** (berisi beberapa tipe tombol). Fungsi tombol pada aplikasi adalah untuk berpindah dari satu aktivitas atau tampilan ke aktivitas atau tampilan selanjutnya, sesuai dengan interaksi pengguna. Selain menggunakan Button yang berupa tombol umumnya (default), terdapat beberapa bentuk tombol yang dapat digunakan, seperti ImageButton untuk menggunakan gambar sebagai tombol, CheckBox untuk tampilan tombol yang tampak menggunakan cek, RadioButton yang biasa digunakan untuk menambahkan opsi, Switch yang tampak seperti tombol On/Off, sifatnya adalah True/False, dan sebagainya.
4. **Widgets** (berisi kumpulan widget lain seperti WebView untuk menambahkan tampilan dari web, CalendarView untuk menambahkan tampilan kalender, RatingBar untuk menambahkan tampilan peringkat, dan masih banyak lagi.

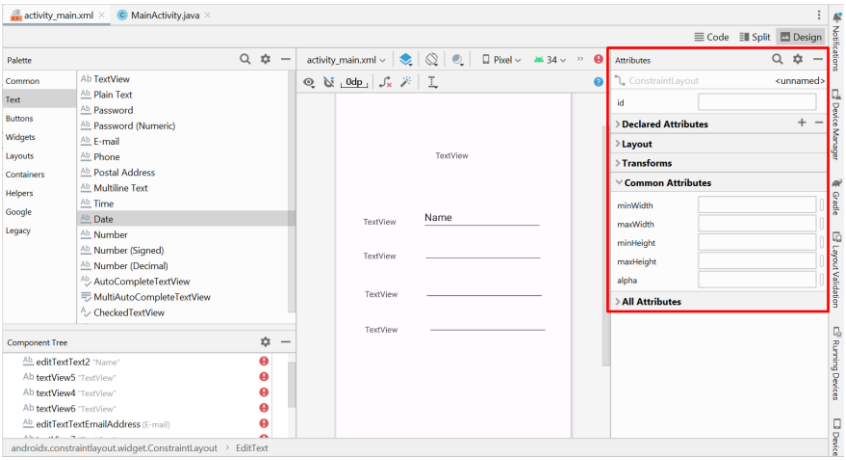
5.2.2 Menambahkan dan Konfigurasi Widget

Penggunaan built-in widget atau widget bawaan Android Studio relatif mudah, hanya perlu melakukan *drag-and-drop* dari Palette ke kanvas desain tampilan yang sedang dibuat. Berikut contoh menambahkan dan mengkonfigurasi widget pada Android Studio.

1. *Drag-and-drop* untuk memilih widget dari Palette ke kanvas.
2. Pada sisi kanan kanvas, terdapat menu Attributes, menu ini adalah tempat kita dapat mengkonfigurasi widget secara langsung. Dari Attributes ini dapat dengan mudah melakukan pengaturan dan kustom pada widget.

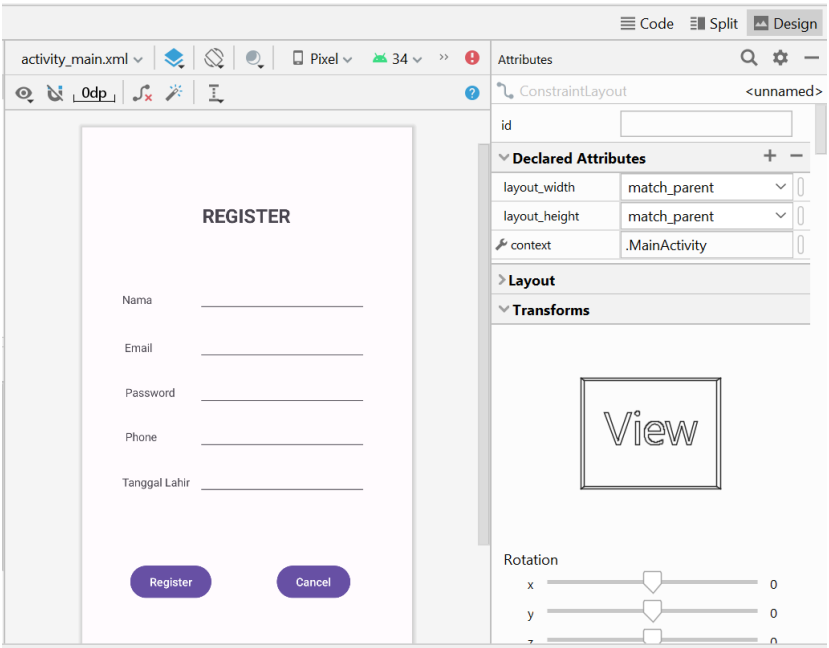
   Konfigurasi yang paling penting dan utama untuk setiap widget adalah dengan memberikan id. ID ini akan digunakan untuk memberikan identitas unik setiap widget, dan memudahkan pengkodean. Penamaan ID disarankan tidak terdapat spasi dan karakter tanda baca seperti (.,;#@/). Misalnya terdapat 3 Button pada satu tampilan, kita dapat memberikan id unik seperti button_start, button_pause, button_pause.

Selain itu, konfigurasi yang perlu dilakukan adalah pengaturan tampilan widget menggunakan atribut ”layout_width” dan ”layout_height”. Terdapat dua pilihan yang umum digunakan yaitu ”wrap_content” untuk mengatur lebar dan tinggi widget sesuai dengan konten di dalamnya, dan ”match_parent” untuk mengatur tinggi dan lebar widget mengikuti *parent* di mana di berada, misalnya di dalam RelativeLayout atau LinearLayout. Namun, selain menggunakan kedua opsi tersebut, kita juga dapat melakukan kustom dengan menginput ukuran yang diinginkan.



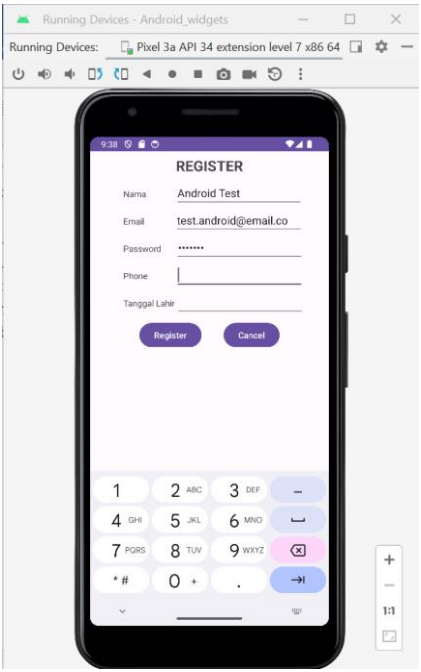
Gambar 5.2. Attributes untuk Konfigurasi Widget

3. Gambar 5.3. Contoh Tampilan Register Sederhana menggunakan Widget TextView dan Button menampilkan contoh membuat tampilan sederhana halaman register. Untuk input dapat dipilih sesuai dengan format input yang diinginkan, seperti PlainText atau EditText yaitu untuk mengisi teks pendek secara bebas, Password untuk format input yang akan otomatis tersembunyi, E-mail untuk format input email, Date untuk format input tanggal. Selain itu, terdapat dua tombol yaitu Register dan Cancel.



Gambar 5.3. Contoh Tampilan Register Sederhana menggunakan Widget TextView dan Button

Pada widget dengan teks seperti TextView dan Button, umumnya dilakukan konfigurasi id, text (teks yang tampil), textSize (ukuran teks), dan textColor atau background (untuk mengatur pewarnaan). Setiap widget memiliki isi atribut yang berbeda – beda sesuai dengan ciri khas widget tersebut.



Gambar 5.4. Contoh Implementasi Widget pada Tampilan Register

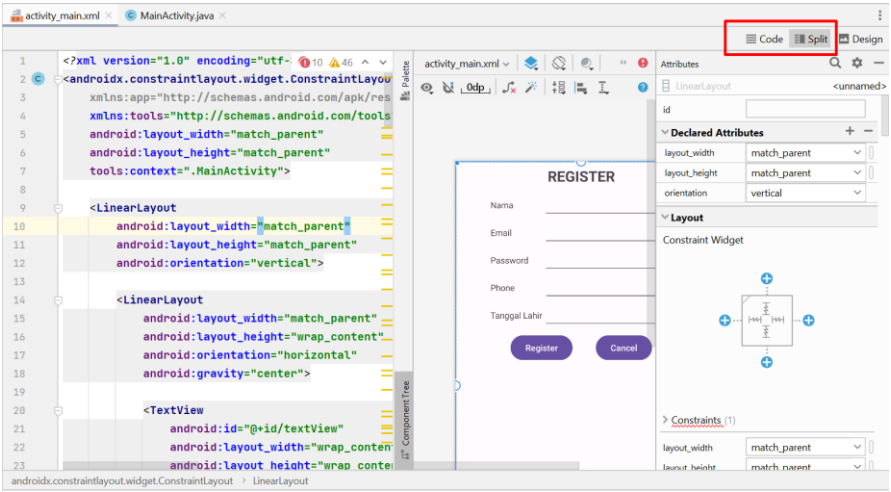
Gambar 5.4. Contoh Implementasi Widget pada Tampilan Register menunjukkan contoh tampilan Register yang dijalankan pada emulator Android. Terlihat bahwa password yang diinput pengguna otomatis disembunyikan. Ketika pengguna ingin mengisi nomor telepon yang sudah diatur menggunakan widget EditText tipe Phone, maka otomatis keypad pada emulator Android berubah menjadi nomor sehingga pengguna hanya dapat mengisi angka saja, menghindari kesalahan input teks.

5.3 Pengenalan Styling dengan XML di Android

Selain dapat melakukan konfigurasi widget melalui pengaturan atribut, konfigurasi atau penyesuaian (kustomisasi) juga dapat diakukan melalui XML. Dalam setiap activity Android, selalu ada file dengan format .xml, dimana kita dapat mengatur tampilan antarmuka pengguna, termasuk widget. Keunggulan konfigurasi atau *styling* melalui XML adalah bisa lebih mudah dibandingkan dengan melalui Attributes, karena fokus pada

bagian yang mau diatur dengan langsung menuliskan atribut yang ingin disesuaikan. Hanya saja, untuk pengaturan melalui XML ini perlu mengetahui apa saja nama atribut karena sifat pengkodean XML di Android Studio ini adalah *case sensitive.*

Untuk melakukan konfigurasi atau pengkodean XML, dapat diakses pada menu seperti pada Gambar 5.5. Konfigurasi dan Styling dengan XML.



Gambar 5.5. Konfigurasi dan Styling dengan XML

Kita bisa memilih “Code” atau “Split” untuk menampilkan kode XML dari tampilan antarmuka yang dibuat. Perbedaannya adalah dengan memilih “Split”, kode XML dan kanvas antarmuka bisa ditampilkan sebelah – menyebelah dan memudahkan konfigurasi.

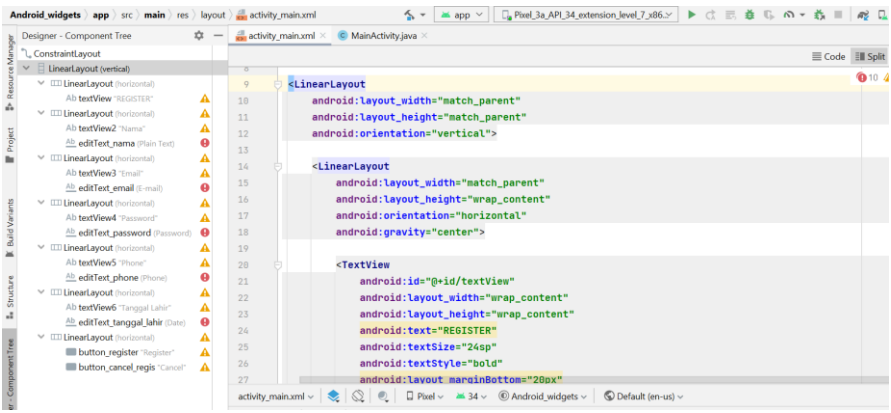
5.3.1 Memahami Struktur XML Layout

File XML diawali dengan deklarasi versi bahasa xml dan *encoding*. Penulisan struktur dalam xml sesuai dengan penempatan pada tampilan antarmuka sehingga sifatnya *parent-child* (Phillips et al., 2019).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="center">

            <TextView
                android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

Gambar 5.6. Contoh XML Tampilan Register

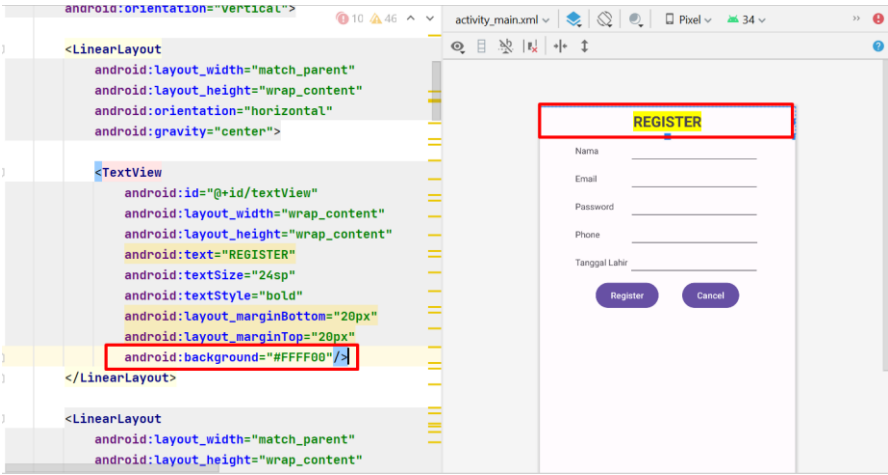
Gambar 5.7. Struktur XML Tampilan Register dengan ComponentTree menampilkan contoh perbandingan struktur XML dari contoh tampilan Register yang telah dibuat sebelumnya, dengan struktur antarmuka pada ComponentTree. Terlihat bahwa urutan kedua struktur tersebut sesuai, hal ini memudahkan untuk konfigurasi dan kustomisasi styling tampilan widget dan antarmuka pengguna.



Gambar 5.7. Struktur XML Tampilan Register dengan ComponentTree

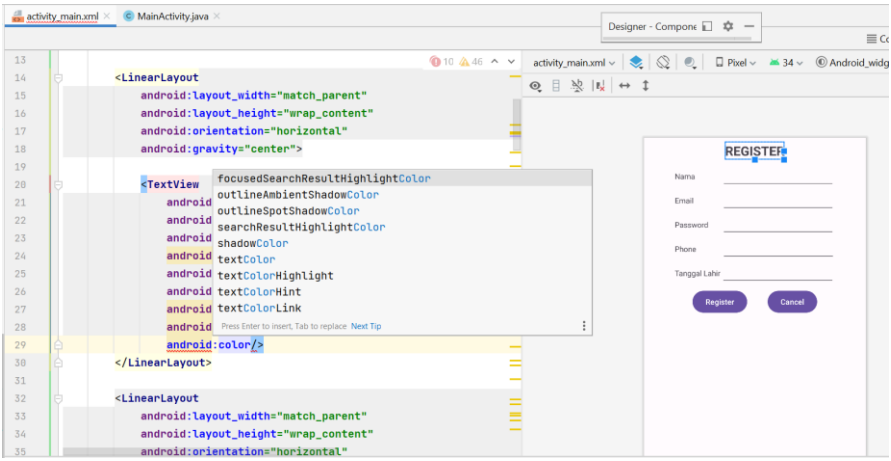
5.3.2 Menggunakan XML untuk Menyesuaikan Gaya Widget

Berikut ini contoh modifikasi tampilan Register dari contoh sebelumnya. Terlihat pada Gambar 5.8. Mengubah Warna *Background* pada Teks "REGISTER", dilakukan perubahan warna *background* pada tulisan judul "REGISTER" dengan menambahkan atribut "background" pada xml.



Gambar 5.8. Mengubah Warna *Background* pada Teks "REGISTER"

Seperti sudah dijelaskan sebelumnya, tantangan konfigurasi langsung dari xml adalah kita perlu mengetahui nama atribut yang tepat untuk dilakukan pengaturan. Namun, Android Studio menyediakan fitur yang dapat dijadikan solusi untuk hal tersebut. Fitur tersebut berupa rekomendasi atribut yang akan muncul ketika pengguna mengetikkan kata kunci atribut yang diinginkan. Sebagai contoh, Gambar 5.9. Android Studio Menyediakan Fitur Rekomendasi untuk XML menunjukkan pengguna yang menegtik kata kunci "color", maka akan tampil beberapa rekomendasi atribut terkait.



Gambar 5.9. Android Studio Menyediakan Fitur Rekomendasi untuk XML

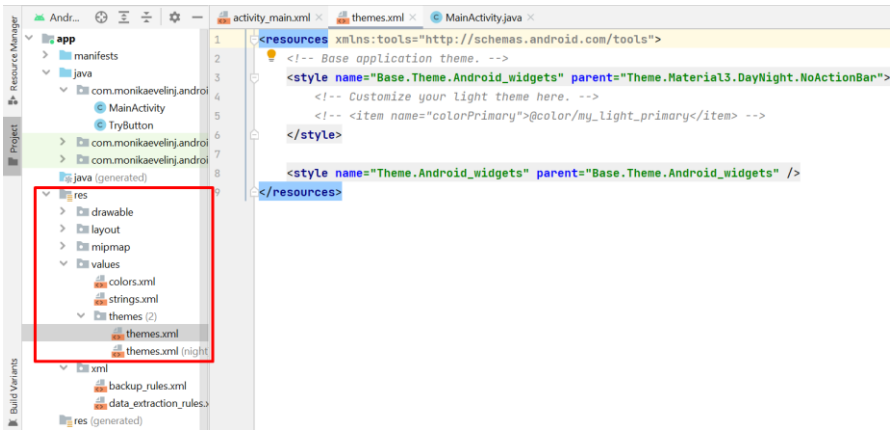
5.3.3 Style dan Tema pada Android

Dalam pembuatan satu aplikasi Android, tentunya terdiri dari banyak halaman atau tampilan. Umumnya, pada satu aplikasi memiliki tema yang selaras antar satu halaman dan halaman lainnya. Hal ini bisa menjadi tantangan tersendiri untuk mendesain tampilan dengan desain yang selaras, tetapi dilakukan berulang – ulang.

Android memungkinkan kita membuat aplikasi dengan desain atau tema tertentu yang dapat diimplementasikan untuk setiap tampilan di dalamnya, yaitu dengan mengatur style dan tema (*theme*). Dengan menentukan tema, kita dapat mengatur gaya (*style*) dari setiap tampilan termasuk widget dengan konfigurasi atribut – atribut terkait, cukup di satu tempat. Perbedaan mendasar antara style dan theme adalah sebagai berikut (Vogel, 2016):

1. Jika entri pada file sumber (seperti gambar, atau media lainnya) digunakan untuk mengatur gaya pada *view*, maka disebut dengan Style.
2. Jika digunakan untuk mengatur gaya pada aktivitas (*activity)* atau aplikasi (secara keseluruhan), maka disebut Theme.

Baik Style ataupun Theme pada Android juga dikonfigurasi menggunakan bahasa xml. Ketika membuat suatu proyek Android baru, otomatis akan terbuat juga file bernama "theme.xml" yang berada di dalam folder "/res/values/", seperti dapat dilihat di Gambar 5.10. Theme.xml pada Android Studio.



Gambar 5.10. Theme.xml pada Android Studio

File theme.xml diawali dengan tag <resource>. Untuk mendefinisikan Style, digunakan tag <style> dan diberikan atribut "name" untuk memberikan identitas setiap style. Dalam entri ini dapat diisi dengan satu atau lebih item yang mendefinisikan nilai untuk setiap atribut.

Ketika ingin menggunakan style tertentu yang telah dibuat atau dikonfigurasikan ke layout pada file lain, kita dapat menggunakan atribut style="@style/text". Dengan menggunakan Style dan Theme, kita dapat menggunakan kembali pengaturan atau konfigurasi atribut dengan menggunakan atribut "parent", dimana style yang memiliki atribut ini dapat menurunkan (inherits) pengaturan style ke tampilan lain.

Penggunaan Style juga dapat berguna untuk menimpa atau menggantikan atribut dari style yang lain (Phillips et al., 2019). Gambar 5.11. Contoh Menggunakan Style yang Sudah Ada merupakan contoh kode program xml untuk modifikasi sebuah style bernama "StylePertama". Pada contoh tersebut, style

“StylePertama” digunakan kembali untuk style di bawahnya, tetapi dengan penambahan modifikasi yaitu “.Strong”, dimana akan mengubah objek menjadi mengikuti style dari “StylePertama”, dengan penambahan tulisan menjadi tebal (bold).

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Android_widgets" parent="Theme.Material3.DayNight.NoActionBar">
        <style name="StylePertama">
            <item name="android:background">@color/dark_blue</item>
        </style>
        <style name="StylePertama.Strong">
            <item name="android:textStyle">bold</item>
        </style>
    </style>

    <style name="Theme.Android_widgets" parent="Base.Theme.Android_widgets" />
</resources>
```

Gambar 5.11. Contoh Menggunakan Style yang Sudah Ada

Salah satu keterbatasan menggunakan Style adalah kita perlu menerapkan atau memasang style tersebut untuk setiap widget. Hal ini akan merepotkan jika dapat membuat aplikasi yang kompleks dengan banyak layout dan widget. Untuk itu, perlu digunakan Theme atau tema, yang memungkinkan kita hanya perlu mengatur serangkaian atribut pada satu tempat, misalnya style, tetapi dapat secara otomatis diaplikasikan ke seluruh tampilan aplikasi. Misalnya, kita ingin menyamakan pewarnaan untuk semua tampilan di aplikasi, maka cocok untuk ditempatkan pada tema. Atau, jika kita ingin menyamakan semua tampilan tombol atau widget tertentu.

5.4 Membuat Custom Widget

Android Studi tidak hanya menyediakan built-in widget, tetapi juga memungkinkan kita untuk membuat kustomisasi widget. Fitur ini memungkinkan kita membuat suatu widget dengan tampilan ataupun fungsi yang lebih fleksibel sesuai dengan kebutuhan. Misalnya, kita bisa membuat suatu gambar yang memiliki fungsi menjadi sebuah tombol.

5.4.1 Penggunaan Custom Widget dalam Aplikasi

Pengaturan tampilan atau gaya pada atribut suatu widget mungkin terbatas. Misalnya untuk tampilan tombol kita hanya dapat menggunakan bentuk dan warna yang didukung pada Android Studio, sedangkan ingin membuat tombol dengan warna dan bentuk tertentu yang lebih variatif. Sebagai solusi, kita dapat menggunakan media lain seperti gambar atau figur, yang diatur untuk berfungsi sebagai tombol.

5.4.2 Langkah – Langkah Pembuatan Custom Widget

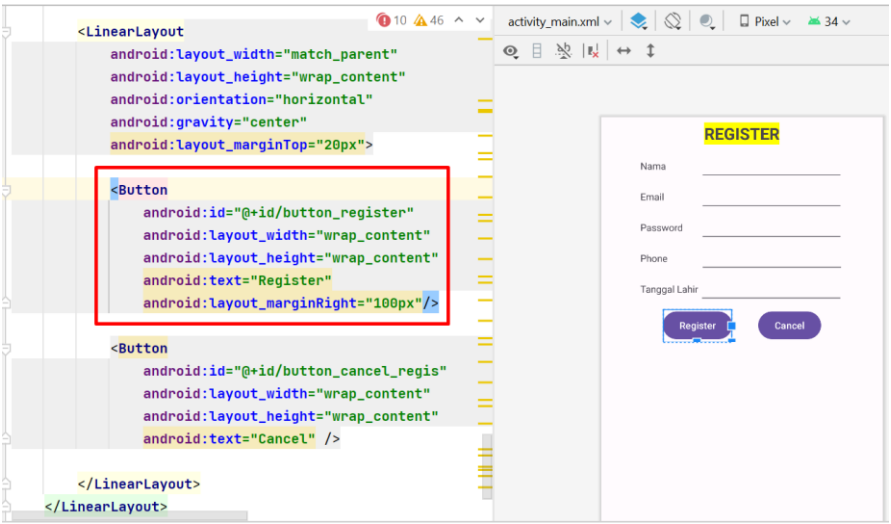
Tidak ada langkah spesifik untuk membuat custom widget, tergantung pada widget seperti apa yang ingin Anda buat. Misalnya dari tampilan atau fungsionalitas dari widget. Beberapa hal yang perlu dipersiapkan dalam membuat custom widget:

1. Menentukan desain dan fungsionalitas. Fungsionalitas mencakup perilaku apa yang ingin diterapkan pada widget, seperti interaksi (*event*) apa yang akan dihasilkan oleh widget.
2. Membuat kelas untuk custom widget. Kelas digunakan untuk menambahkan logika pada widget. Kelas ini perlu mewarisi dari kelas View atau subkelas dari View.
3. Terapkan logika dan tampilan. Logika dan tampilan disesuaikan dengan kegunaan dari custom widget yang ingin dibuat. Tampilan dapat diatur pada XML, atau menggunakan kode java pada kelas widget. Sedangkan logika ditambahkan pada kelas widget.
4. Tes dan uji custom widget. Sebagai langkah pendukung, setelah membuat custom widget, uji apakah custom widget tampil dan dapat berfungsi dengan baik sesuai harapan, serta terintegrasi dalam aplikasi yang dibuat. Kadang, widget tambahkan yang kita buat bisa berefek pada tampilan atau fitur lainnya dan justru membuat berantakan jika tidak dibuat dengan baik.

5.4.3 Menerapkan Styling Khusus pada Custom Widget

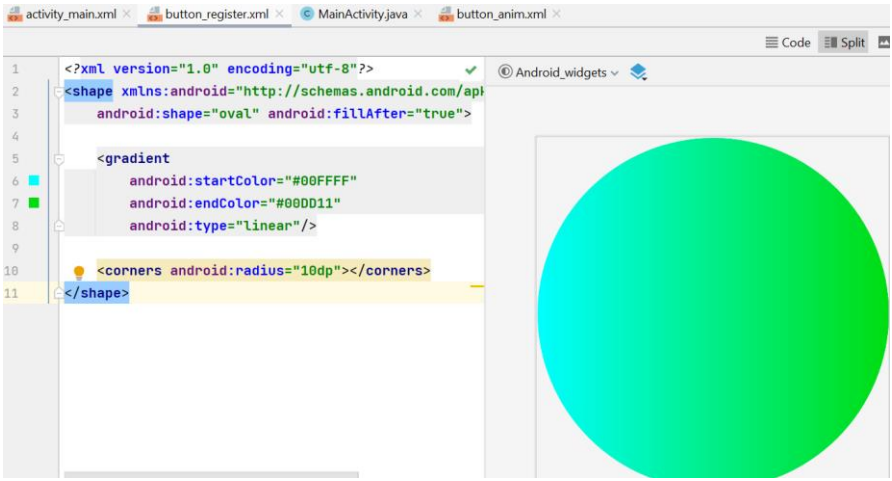
Berikut adalah contoh pembuatan custom widget. Pada contoh di bawah ini, melakukan modifikasi dari halaman Register yang telah dibuat sebelumnya, dengan melakukan perubahan style atau gaya dari tombol Register.

Gambar 5.12. Tampilan Register Sebelum Dikustomisasi merupakan kode XML dari tombol Register sebelumnya. Custom dilakukan dengan modifikasi pada XML tersebut.



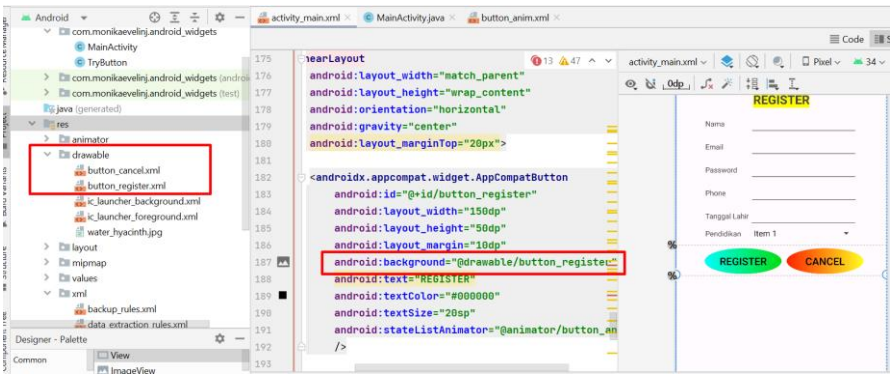
Gambar 5.12. Tampilan Register Sebelum Dikustomisasi

Kemudian dilakukan modifikasi pada tombol "Register" dan "*Cancel*", dengan mengubah latar belakang warna dari kedua tombol tersebut. Latar belakang diambil dari file "/res/drawable/button_register.xml, yang isinya adalah kode xml dari tampilan yang telah dikustom, seperti pada Gambar 5.13. Kustomisasi pada button_register.xml



Gambar 5.13. Kustomisasi pada button_register.xml

Tampilan yang telah diatur pada button_register.xml tersebut dipanggil di halaman Register dan diaplukasikan sehingga terlihat seperti Gambar 5.14. Tampilan Register Setelah Dikustomisasi.



Gambar 5.14. Tampilan Register Setelah Dikustomisasi

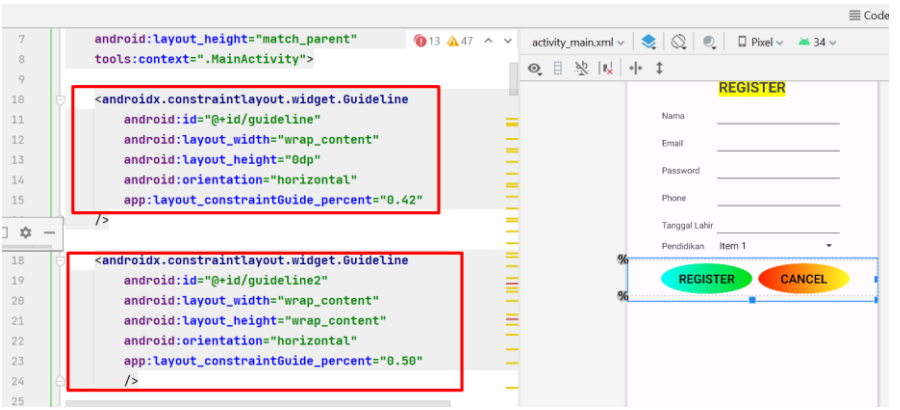
5.5 Responsif dan Fleksibel

Desain responsive adalah istilah untuk menggambarkan halaman secara progresif untuk konteks tampilan yang berbeda melalui penggunaan teknik pengkodean, termasuk tata letak yang lancar, gambar yang fleksibel, dan kueri media (Felke-Morris, 2018). Sebagai contoh umum, tata letak yang mengikuti

arah layout (vertikal atau horisontal), dan ukuran yang bisa mengikuti ukuran layar perangkat yang berbeda – beda. Kita juga bisa membuat widget yang responsif terhadap aksi ari pengguna, misalnya tampilan atau perubahan yang terjadi jika sebuah tombol ditekan. Desain yang responsif sangat penting dan mempengaruhi kepuasan pengalaman pengguna (*user experience)* dalam penggunaan aplikasi.

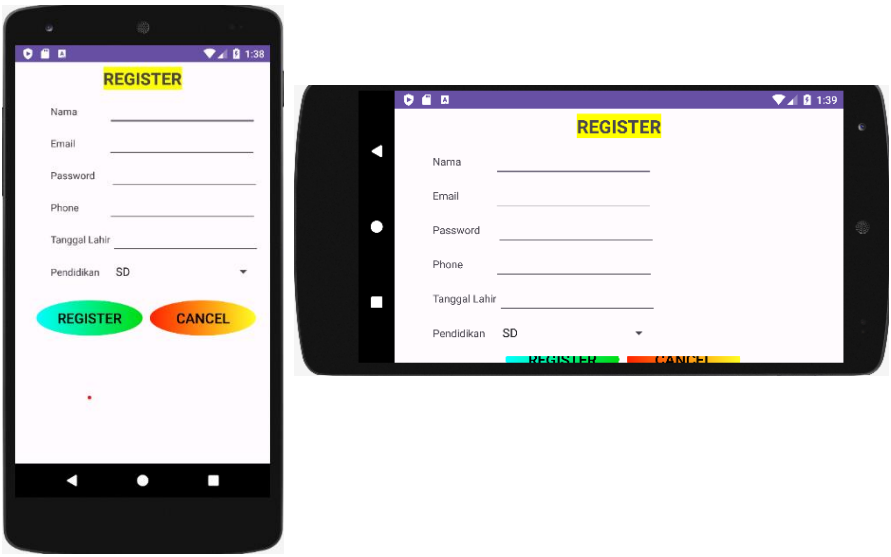
5.5.1 Membuat Widget yang Responsif

Untuk membuat widget yang responsif, hal pertama yang dilakukan adalah menentukan layout. Sebagai contoh, layout yang digunakan adalah RelativeLayout. Kemudian, melakukan modifikasi pada widget yang mau dibuat responsif. Setelah itu, mengatur responsivitas pada file *activity*, atau bisa memodifikasi kelas kode (java).



Gambar 5.15. Menambahkan Widget Guideline untuk Responsivitas terhadap Orientasi Layar

Gambar 5.15. Menambahkan Widget Guideline untuk Responsivitas terhadap Orientasi Layar merupakan contoh penerapan widget responsif menggunakan widget "Guidelines". Widget ini membuat isi konten pada tampilan akan mengikuti orientasi layar, yaitu ketika horisontal dan vertikal. Gambar 5.16. Tampilan Register Vertikal dan Horisontal menunjukkan arah widget pada tampilan Register mengikuti arah orientasi layar.



Gambar 5.16. Tampilan Register Vertikal dan Horisontal

5.6 Animasi dan Interaksi

Animasi seringkali ditambahkan pada aplikasi Android untuk meningkatkan pengalaman pengguna. Selain itu, animasi akan membuat tampilan menjadi lebih menarik.

5.6.1 Pengenalan Animasi di Android

Ada beragam cara untuk menambahkan dan membuat animasi pada aplikasi Android. Berikut merupakan beberapa contoh alat yang dapat digunakan untuk membuat animasi pada aplikasi Android (Phillips et al., 2019).

1. *State List Animators. State list animators* adalah bagian dari state list drawable, dimana menampilkan animasi ke dalam status tertentu. Sebagai contoh implementasinya, ketika menekan tombol, animasi akan naik pada z-axis sampai bertemu jari yang menekan, dan ketika tombol dilepas animasi akan kembali ke bawah. State list animators baik digunakan ketika ingin menggunakan properti animasi. Sedangkan jika ingin membuat animasi yang sudah tersusun

(*framed*), maka perlu menggunakan alat lain yaitu animated state list drawable.

Animated state list drawable memungkinkan untuk menentukan gambar untuk setiap status, dan juga memungkinkan menentukan susunan transisi animasi antar setiap status.

2. *Tween Animations. Animasi tween* mengacu pada perubahan secara halus dari properti – properti tampilan, seperti skala, rotasi, transisi. Kata "*tween*" sendiri merupakan kependekan dari kata "*between*" (antara), yang terjadi antara dua nilai (*Android for Developers*, n.d.). Misalnya, menggunakan tween untuk menentukan durasi perubahan dari satu properti ke properti lain.
3. *AnimatorSet. AnimatorSet* adalah kelas yang memungkinkan kita mengatur serangkaian animator untuk dijalankan secara bersamaan atau berurutan. Dengan ini, kita dapat membuat animasi yang kompleks dengan melibatkan beberapa animator.
4. *Transitions. Transitions* (transisi) memungkinkan untuk membuat animasi yang mulus ketika ada perpindahan antar tampilan, seperti saat perubahan navigasi layar atau perubahan properti tampilan. Beberapa jenis transisi yang tersedia pada Android diantaranya "*Fade*", "*Slide*", "*Explode*".
5. *Drawable Animations. Drawable animations* memungkinkan untuk membuat animasi dari serangkaian gambar atau menggunakan animasi yang didefinisikan dalam XML pada tampilan yang menggunakan drawable sebagai latar belakang atau gambar.
6. *Custom View Animations*. Seperti halnya custom widget, Android memungkinkan kita membuat animasi yang dikustom dengan menerapkan animasi langsung pada tampilan atau membuat tampilan kustom yang menangani animasi secara khusus sesuai kebutuhan aplikasi.

5.6.2 Menambahkan Animasi pada Widget

Sebagai contoh, file button_anim.xml seperti terlihat pada Gambar 5.17. Pengaturan Animasi pada button_anim.xml berisi pengaturan animasi yang akan diterapkan pada tombol

”Register” dan ”Cancel” di tampilan halaman Register. File tersebut kemudian dipanggil dengan penambahan kode pada activity.xml pada atribut ”stateListAnimator”. Contoh kode dapat dilihat pada Gambar 5.18. Memanggil button_anim.xml pada Atribut stateListAnimator

y_main.xml × button_register.xml × MainActivity.java × button_anim.xml ×

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
        <item
            android:state_enabled="true"
            android:state_pressed="true">

        <set>

            <objectAnimator
                android:duration="100"
                android:propertyName="scaleX"
                android:valueTo="1.3"
                android:valueType="floatType" />

            <objectAnimator
                android:duration="100"
                android:propertyName="scaleY"
                android:valueTo="1.3"
                android:valueType="floatType" />
        </set>
```

Gambar 5.17. Pengaturan Animasi pada button_anim.xml

Gambar 5.18. Memanggil button_anim.xml pada Atribut stateListAnimator

5.6.3 Meningkatkan Interaktivitas Widget

Interaktivitas widget adalah salah satu cara untuk menambah pengalaman pengguna, dimana konten pada tampilan aplikasi dibuat lebih dinamis dan dapat bereaksi terhadap aktivitas atau input pengguna.

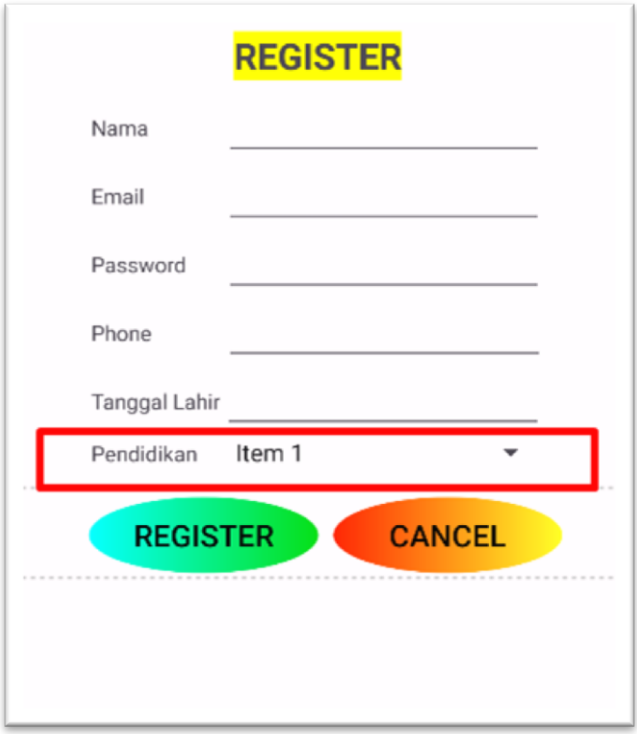
Animasi dapat digunakan untuk membuat widget lebih menarik dan interaktif. Misalnya, menambahkan animasi ketika widget dimuat, atau ketika pengguna berinteraksi dengan widget seperti menekan widget. Selain itu, interaktivitas pada widget juga dapat ditambahkan misalnya pada input pengguna. Sebagai contoh, memberikan pilihan (*dropdown*) ke dalam widget sehingga pengguna dapat melakukan pencarian opsi tertentu. Hal lain yang dapat diterapkan untuk menambah interaktivitas widget adalah dengan menampilkan konten yang dapat berubah sesuai kondisi tertentu, seperti widget maps yang dapat menampilkan lokasi pengguna secara otomatis.

Interaktivitas juga dapat diterapkan dengan memberikan kontrol pada widget, misalnya jika ada tampilan pemutar musik atau video, terdapat tombol play, pause, dan stop untuk memungkinkan pengguna mengatur pemutaran media yang diinginkan.

Pada contoh halaman Register berikut, ditambahkan kolom pengisian ”Pendidikan” yang menggunakan widget Spinner. Widget ini dapat digunakan untuk menampilkan

pilihan input berupa daftar (list). Isi atau item pada daftar perlu ditambahkan pada file MainActivity.java.

Gambar 5.19. Menambahkan Widget Spinner untuk Pengisian Pendidikan pada Register merupakan contoh tampilan setelah menambahkan widget Spinner untuk ”Pendidikan” pada halaman Register, sedangkan kode untuk mengisi item pada Spinner terlihat pada Gambar 5.20. Daftar Item dari Spinner pada MainActivity.java.

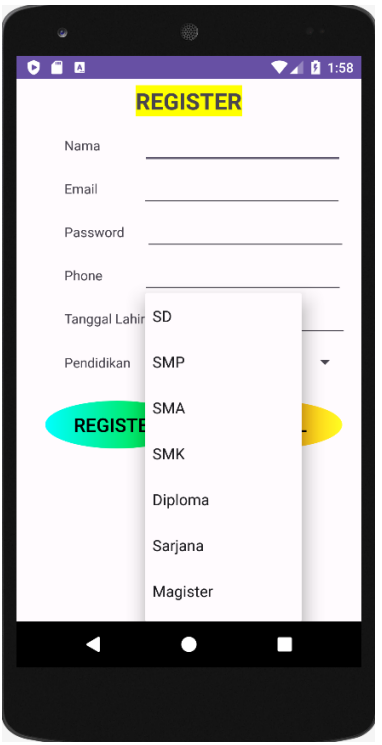


Gambar 5.19. Menambahkan Widget Spinner untuk Pengisian Pendidikan pada Register

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Spinner dropdown = findViewById(R.id.spinner_pendidikan);

        String[] pendidikan = new String[]{"SD","SMP","SMA","SMK","Diploma","Sarjana","Magister","Doktor"};

        ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_dropdown_item,

        dropdown.setAdapter(adapter);
```

Gambar 5.20. Daftar Item dari Spinner pada MainActivity.java

Hasil dari penerapan widget Spinner ketika aplikasi dijalankan dapat dilihat pada Gambar 5.21. Tampilan Daftar dari Widget Spinner.



Gambar 5.21. Tampilan Daftar dari Widget Spinner

Seiring dengan berkembangnya teknologi Android, semakin bertambah tipe – tipe widget yang dapat dirancang

pada aplikasi Android. Android Studio memungkinkan kita melakukan banyak modifikasi dan pengaturan widget dari sisi tampilan, perilaku widget, dan fungsionalitas yang lebih fleksibel dan kemungkinan yang beragam. Dengan begitu kita dapat membuat aplikasi semakin dinamis dan menarik sehingga meningkatkan interaktivitas dan pengalaman bagi pengguna.

DAFTAR PUSTAKA

Android for Developers. (n.d.). Android Studio. Retrieved January 25, 2024, from https://developer.android.com/

Felke-Morris, T. (2018). *Web Development and Design Foundations with HTML5* (Global Edi). Pearson International Content.

Mansyur, A. R. (2020). Dampak COVID-19 Terhadap Dinamika Pembelajaran Di Indonesia. *Education and Learning Journal*, *Vol. 1*, *No*, 113–123.

Phillips, B., Stewart, C., Hardy, B., & Marsicano, K. (2019). *AndroidPhillips, B. et al. (2019) Android Programming: The Big Nerd Ranch Guide. 4th Editio. Big Nerd Ranch. Programming: The Big Nerd Ranch Guide* (4th Editio). Big Nerd Ranch.

Vogel, L. (2016). *Android application design with styles and themes - Tutorial*. https://www.vogella.com/tutorials/AndroidStylesThemes/article.html

BAB 6
FRAGMENT (MEMBANGUN ANTARMUKA YANG FLEKSIBEL)

Oleh Ratnasari

6.1 Pendahuluan

6.1.1 Pengertian Fragment

Fragment adalah bagian UI aplikasi yang dapat digunakan Kembali(*Android Developers*). Fragment merupakan bagian dari *activity* (*sub-activity*). Fragment memiliki struktur kode yang mirip dengan *activity*, dimana fragment juga memiliki sebuah kelas untuk menampung kode logika dan file XML untuk merepresentasikan tampilannya. Sebuah fragment dapat menentukan dan mengelola tata letaknya sendiri, memiliki siklus hidupnya sendiri, serta dapat menangani peristiwa inputnya sendiri. Namun, fragment tidak dapat berjalan sendiri karena harus dihosting oleh activity atau fragment lain.

6.1.2 Keuntungan Menggunakan Fragment

Berikut ini adalah beberapa keuntungan utama menggunakan fragmen:

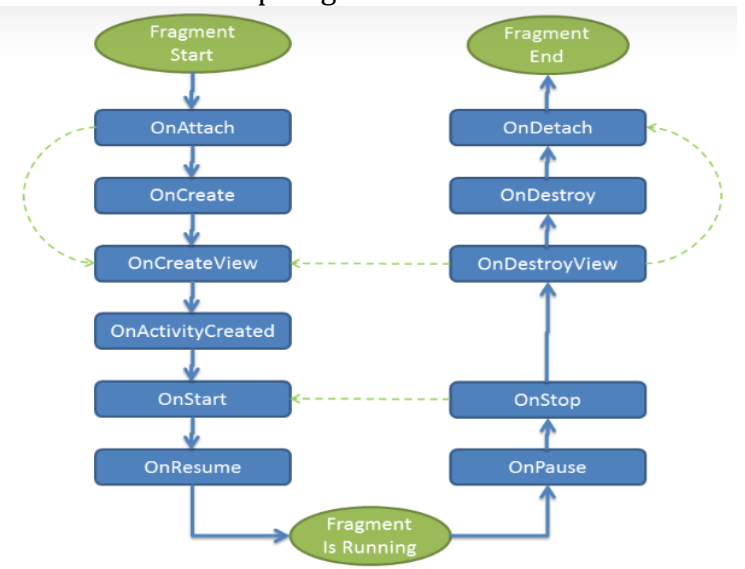
1. Modularitas
   Fragment memungkinkan pengembang untuk memecah aktivitas besar menjadi beberapa komponen yang lebih kecil dan dapat diatur secara terpisah. Dengan cara ini, setiap fragment bisa mengatur UI dan logika bisnisnya sendiri, sehingga lebih mudah untuk dikelola dan di-debug.
2. Reusabilitas
   Fragment dapat digunakan kembali di berbagai aktivitas atau bahkan dalam aplikasi yang berbeda. Ini sangat berguna ketika ada elemen UI yang sama yang harus digunakan di berbagai tempat dalam aplikasi. Misalnya, sebuah fragment yang menampilkan daftar item bisa

digunakan di beberapa layar yang berbeda tanpa perlu menduplikasi kode.

3. Dukungan Multi-Layar
   Fragment memungkinkan pembuatan UI yang lebih fleksibel dan responsif terhadap ukuran layar yang berbeda. Pada perangkat dengan layar besar seperti tablet, beberapa fragment bisa ditampilkan berdampingan (*side-by-side*) untuk memanfaatkan ruang layar yang lebih besar. Pada perangkat dengan layar kecil seperti ponsel, fragment yang sama bisa ditampilkan satu per satu.
4. Pengelolaan Dinamis
   Fragment dapat ditambahkan, dihapus, atau diganti selama runtime, memungkinkan pengembang untuk mengubah UI berdasarkan interaksi pengguna atau perubahan kondisi aplikasi. Misalnya, pada sebuah aplikasi yang menggunakan tab, setiap tab bisa menjadi sebuah fragment yang berbeda, yang dapat dimuat secara dinamis saat pengguna beralih antara tab.

6.1.3 Daur Hidup Fragment

Berikut siklus daur hidup fragment :



Gambar 6.1. Daur Hdup fragment

Keterangan:

1. onAttach() dipanggil ketika fragment terhubung (attached) ke activity yang meng-hostnya. Metode ini adalah salah satu dari siklus hidup fragment yang pertama kali dipanggil setelah fragment dibuat.
2. onCreate() dipanggil saat fragment sedang dibuat atau diinisialisasi (objeknya berada di memori).
3. onCreateView() dipanggil ketika fragment harus membuat tampilannya, yaitu ketika fragment pertama kali dibuat atau ketika tampilan fragment perlu diperbarui. Metode ini bertanggung jawab untuk membuat tampilan fragment dengan mengembalikan tampilan yang akan ditampilkan dalam layout activity.
4. onViewCreated() dipanggil setelah metode onCreateView() selesai dieksekusi. Metode ini memberikan sinyal bahwa tampilan fragment telah selesai dibuat dan siap untuk diakses. Pada tahap ini, dapat melakukan inisialisasi dan mengakses elemen-elemen UI yang ada dalam tampilan fragment.
5. onActivityCreated() dipanggil setelah metode onCreate() dari activity yang meng-host fragment selesai dieksekusi. Metode ini memberikan sinyal bahwa activity telah selesai dibuat dan siap untuk menerima interaksi dengan fragment.
6. onStart() dipanggil ketika fragment siap ditampilkan dilayar dan siap untuk menerima interaksi.
7. onResume() dipanggil ketika fragment menjadi aktif dan interaktif untuk pengguna.
8. onPause() dipanggil ketika fragment tidak lagi aktif secara interaktif.
9. onStop() dipanggil ketika fragment tidak lagi terlihat oleh pengguna di layar.
10. onDestroyView() dipanggil ketika tampilan fragment akan dihancurkan atau dihapus dari hierarki tampilan,namun fragmentnya masih ada di memori.
11. onDestroy() dipanggil ketika fragment akan dihancurkan secara permanen dan sudah tidak lagi digunakan.
12. onDetach() dipanggil ketika fragment tidak lagi terhubung (detached) dari activity yang meng-hostnya.

6.2 Fragment Manager

Saat aplikasi berjalan, FragmentManager dapat melakukan penambahan, penghapusan, penggantian, dan tindakan lainnya pada fragmen sebagai tanggapan terhadap interaksi pengguna.
Berikut Method penting dari FragmentManager:

Tabel 6.1. Method penting FragmentManager

Method	Deskripsi
addOnBackStackChangedListener	Menambahkan listener yang akan dipanggil setiap kali ada perubahan pada back stack fragment. Ini memungkinkan Anda untuk memantau perubahan dalam back stack dan melakukan tindakan tertentu ketika back stack berubah, seperti memperbarui UI atau mengelola state fragment.
beginTransaction()	Digunakan untuk memulai sebuah transaksi fragment. Transaksi fragment digunakan untuk menambah, mengganti, atau menghapus fragment secara dinamis dalam sebuah activity. Setelah memulai transaksi, Anda dapat menambahkan, mengganti, atau menghapus fragment, serta menerapkan atau membatalkan transaksi tersebut.

Method	Deskripsi
findFragmentById(int id)	Mencari dan mengambil referensi ke sebuah fragment berdasarkan ID yang ditentukan dalam layout XML activity yang meng-host fragment tersebut.
findFragmentByTag(String tag)	Digunakan untuk mencari dan mengambil referensi ke sebuah fragment berdasarkan tag yang ditentukan saat fragment ditambahkan secara dinamis ke dalam activity.
popBackStack()	digunakan untuk menghapus fragment teratas dari back stack fragment. Ini berarti bahwa fragment teratas yang ditambahkan ke dalam back stack akan dihapus dan transisi kembali ke fragment sebelumnya, jika ada.
executePendingTransactions()	Digunakan untuk segera menjalankan transaksi fragment yang tertunda. Transaksi fragment yang dimulai dengan beginTransaction() akan dieksekusi secara asinkron.

Setiap rangkaian perubahan fragmen yang diterapkan (commit) disebut sebagai transaksi, dan tindakan yang akan dilakukan di dalam transaksi menggunakan API dapat

difasilitasi oleh class FragmentTransaction. FragmentTransaction dapat di instance dari FragmentManager dengan memanggil beginTransaction(), seperti dalam contoh berikut:

```java
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

Hal ini memungkinkan untuk memulai dan mengelola transaksi fragment, seperti menambah, mengganti, atau menghapus fragment dalam activity.

Untuk menyelesaikan transaksi pada FragmentTransaction, harus memanggil metode commit(). Metode ini memberikan instruksi kepada FragmentManager bahwa semua operasi yang ditambahkan ke dalam transaksi telah selesai dan harus diterapkan.

Contoh:

```java
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

// Melakukan operasi fragment, misalnya:
// fragmentTransaction.add(R.id.fragment_container, new YourFragment());
// fragmentTransaction.replace(R.id.fragment_container, new YourFragment());
// fragmentTransaction.remove(fragment);

// Menyelesaikan transaksi dengan commit()
fragmentTransaction.commit();
```

Dalam contoh di atas, setelah semua operasi fragment ditambahkan ke dalam FragmentTransaction, kita memanggil commit() untuk menyelesaikan transaksi. Ini akan memberikan instruksi kepada FragmentManager untuk menerapkan semua operasi yang telah ditambahkan ke dalam transaksi.

6.2.1 Melakukan Transaksi pada Fragment

Berikut adalah beberapa cara umum untuk melakukan transaksi pada fragment:

1. Menggunakan FragmentTransaction secara langsung:
   Membuat instance dari FragmentManager dan memulai transaksi dengan memanggil beginTransaction(). Kemudian, menambahkan, mengganti, atau menghapus fragment menggunakan method seperti add(), replace(), dan remove(), diikuti dengan pemanggilan commit() untuk menerapkan transaksi tersebut.
   Contoh:

```java
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
fragmentTransaction.add(R.id.fragment_container, new YourFragment());
fragmentTransaction.commit();
```

2. Menggunakan XML dalam layout activity:
   Dapat dilakukan dengan cara menentukan fragment langsung dalam layout XML activity menggunakan tag <fragment>. Ini memungkinkan untuk menentukan fragment secara statis dalam file layout dan sistem akan menanganinya secara otomatis juga.
   Contoh :

```xml
<fragment
    android:id="@+id/your_fragment"
    android:name="com.example.YourFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

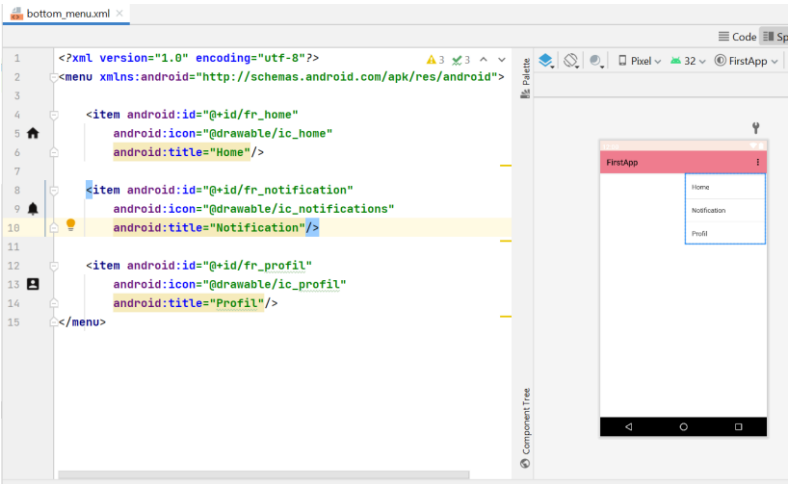
3. Melakukan transaksi di dalam Fragment
   Jika ingin melakukan transaksi fragment dari dalam fragment itu sendiri, maka dapat menggunakan method getChildFragmentManager() untuk mendapatkan instance dari FragmentManager yang terkait dengan fragment tersebut, dan kemudian melakukan transaksi seperti biasa.
   Contoh :

```java
FragmentTransaction fragmentTransaction = getChildFragmentManager().beginTransaction()
fragmentTransaction.replace(R.id.fragment_container, new AnotherFragment());
fragmentTransaction.commit();
```

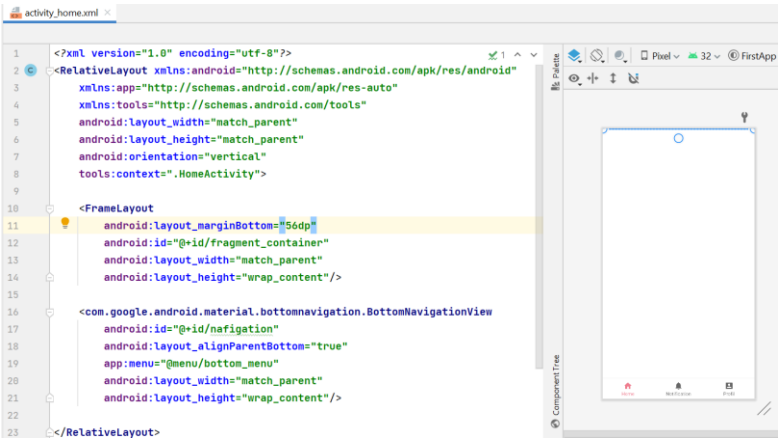
6.2.2 Implementasi Fragment

Berikut adalah contoh penggunaan fragment untuk menu Home, Notification dan Profil :

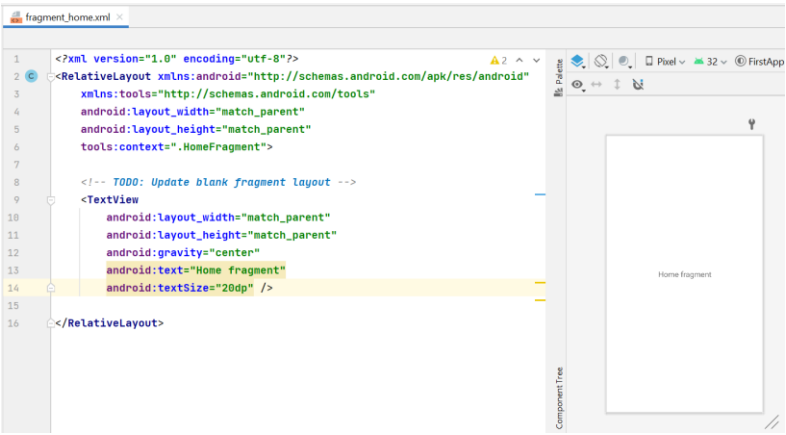
1. Membuat Menu Resource File pada folder menu, seperti berikut:



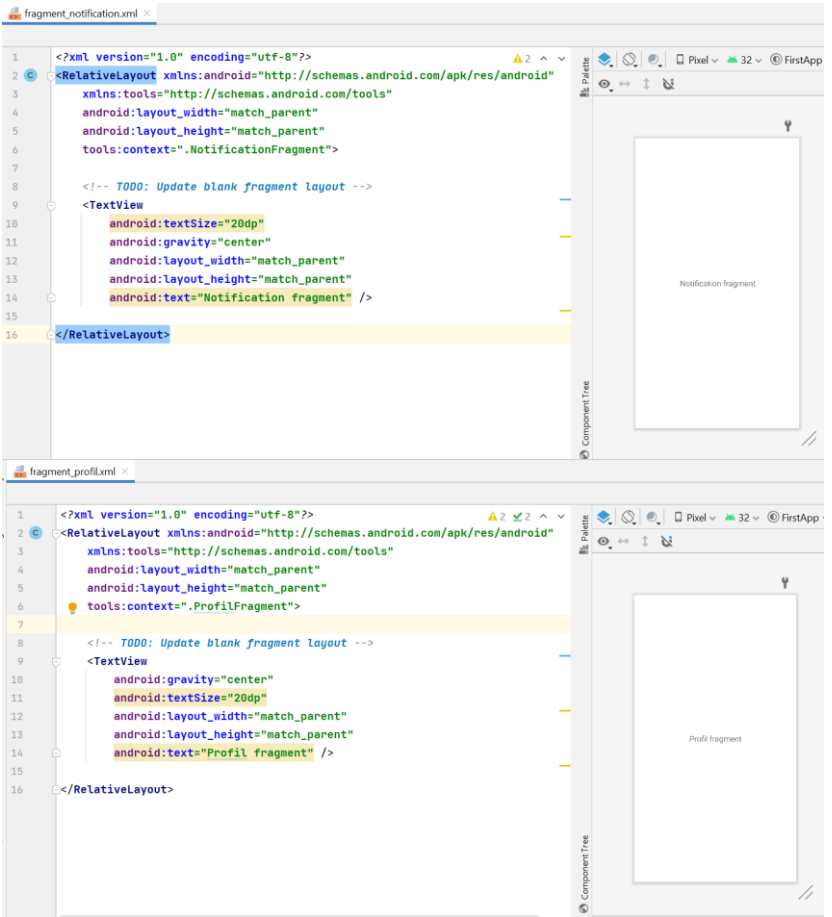
2. Menambahkan activity baru untuk activity untuk navigation, dengan cara Klik kanan pada direktori app atau java di dalam panel Project, lalu pilih New > Activity > Empty Activity atau jenis activity lainnya sesuai kebutuhan(Contoh HomeActivity).Lalu isi nama Activity yang sesuai kemudian pilih Finish. Pada File .xml isi perintah seperti berikut:



3. Menambahkan fragment home, notification dan profil dengan cara Klik kanan pada direktori app atau java di dalam panel Project, lalu pilih New > Fragment > Fragment (Blank) atau jenis Fragment lainnya sesuai kebutuhan.Lalu isi nama Fragment yang sesuai contohnya fragment_home, kemudian pilih Finish. Pada File .xml isi perintah seperti berikut:



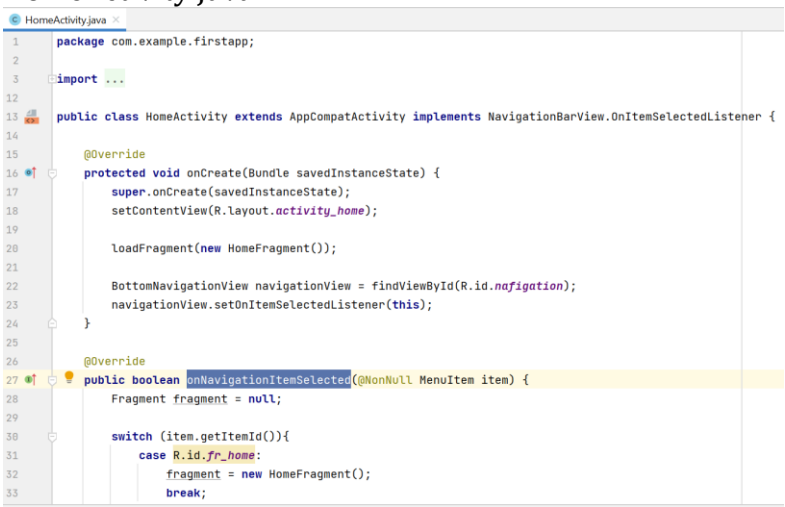
Lakukan hal yang sama untuk menambahkan fragment notification dan fragment profil, lalu tambahkan perintah berikut pada file .xml



4. Menempelkan Fragment Kedalam Activity
   Ada dua cara untuk menempelkan sebuah fragment ke dalam activity: secara dinamis menggunakan Java dan secara statis (manual) dengan XML. Sebelum menempelkan fragment dari library "support" di dalam Activity, pastikan bahwa Activity sudah meng-extends FragmentActivity atau AppCompatActivity. Ini memastikan bahwa activity memiliki fragment manager untuk semua versi Android. Berikut adalah contoh penggunaan:

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Menambahkan fragment secara dinamis
        YourFragment fragment = new YourFragment();
        getSupportFragmentManager().beginTransaction()
                                   .add(R.id.fragment_container, fragment)
                                   .commit();
    }
}
```

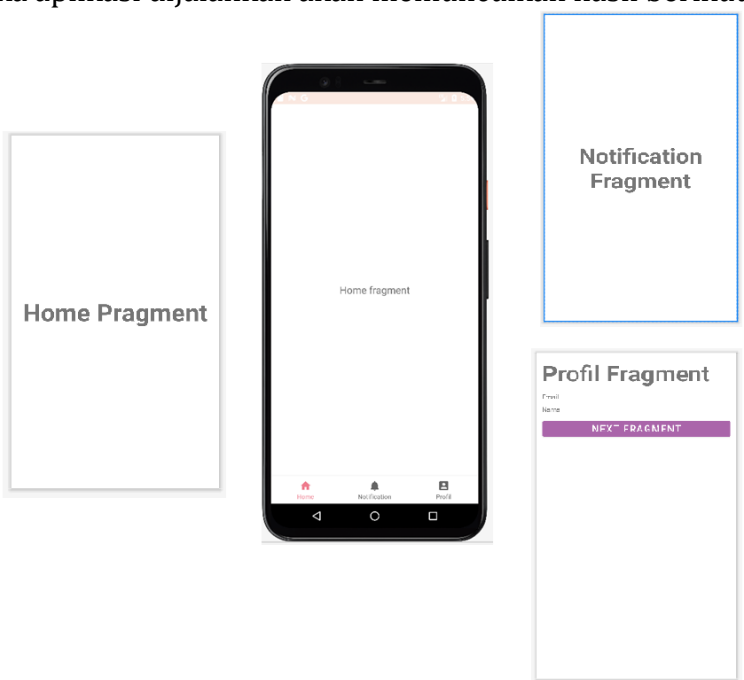
Dalam studi kasus seperti dibawah menempelkan fragment kedalam activity dilakukan dengan kondisi sesuai dengan fragment yang dipilih oleh user. Berikut contohnya pada HomeActivity.Java:



```java
package com.example.firstapp;

import ...

public class HomeActivity extends AppCompatActivity implements NavigationBarView.OnItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        loadFragment(new HomeFragment());

        BottomNavigationView navigationView = findViewById(R.id.nafigation);
        navigationView.setOnItemSelectedListener(this);
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        Fragment fragment = null;

        switch (item.getItemId()){
            case R.id.fr_home:
                fragment = new HomeFragment();
                break;
```

```
HomeActivity.java
34
35              case R.id.fr_notification:
36                  fragment = new NotificationFragment();
37                  break;
38
39              case R.id.fr_profil:
40                  fragment = new ProfilFragment();
41                  break;
42          }
43
44          return loadFragment(fragment);
45      }
46
47      private boolean loadFragment(Fragment fragment) {
48          if(fragment != null){
49              getSupportFragmentManager() FragmentManager
50                      .beginTransaction() FragmentTransaction
51                      .replace(R.id.fragment_container, fragment)
52                      .commit();
53              return true;
54          }
55          return false;
56      }
57  }
```

5. Jika aplikasi dijalankan akan memunculkan hasil berikut:



Gambar 6.2. Contoh implementasi fragment

DAFTAR PUSTAKA

Android Developers. https://developer.android.com/.

BAB 7
PEMROGRAMAN MOBILE DENGAN JAVA: PANDUAN LENGKAP UNTUK ANDROID DEVELOPMENT DENGAN DATABASE REALMS DAN FIREBASE

Oleh Green Ferry Mandias

7.1 Query dalam Realms

7.1.1 Bahasa Query Realms

Bahasa query Realms adalah seperangkat perintah atau pernyataan yang digunakan untuk berinteraksi dengan database Realms. Realms adalah sistem manajemen basis data yang memungkinkan pengguna untuk menyimpan, mengatur, dan mengakses data dengan efisien. Bahasa query Realms memungkinkan pengembang perangkat lunak untuk melakukan berbagai operasi terhadap data, seperti penambahan, penghapusan, pembaruan, dan pencarian data. Bahasa query Realms biasanya berbasis pada bahasa yang intuitif dan mudah dipahami, membuatnya cocok digunakan oleh berbagai tingkat pengguna, mulai dari pemula hingga ahli.

Salah satu fitur utama dari bahasa query Realms adalah kemampuannya untuk memungkinkan pengguna untuk menulis kueri dengan sintaksis yang dekat dengan bahasa manusia. Ini membuatnya lebih mudah dipelajari dan digunakan oleh pengembang yang mungkin tidak memiliki latar belakang yang kuat dalam bahasa pemrograman. Bahasa query Realms sering kali menggunakan perintah seperti SELECT, INSERT, UPDATE, dan DELETE untuk melakukan operasi pada data. Selain itu, bahasa ini juga mendukung berbagai fungsi agregat dan pemfilteran data yang memungkinkan pengguna untuk mengambil informasi yang spesifik dari database. Keunggulan lain dari bahasa query Realms adalah kemampuannya untuk mendukung transaksi, yang memungkinkan pengguna untuk

menjalankan serangkaian operasi database sebagai satu kesatuan yang tidak dapat dipecahkan. Dengan demikian, bahasa query Realms memberikan pengguna fleksibilitas dan kontrol penuh atas manipulasi data dalam database Realms mereka (Smith dan Johnson, 2023).

Selain itu, bahasa query Realms juga mendukung fitur-fitur lanjutan seperti pengelompokan data, pengurutan, dan penggabungan data dari beberapa tabel. Ini memungkinkan pengguna untuk melakukan analisis data yang lebih kompleks dan mendapatkan wawasan yang lebih dalam tentang informasi yang disimpan dalam database Realms. Bahasa query Realms juga sering kali dilengkapi dengan fitur pencarian teks penuh yang memungkinkan pengguna untuk melakukan pencarian berbasis teks di dalam data mereka dengan mudah. Dengan demikian, bahasa query Realms tidak hanya memungkinkan pengguna untuk mengelola data dengan efisien, tetapi juga untuk menjalankan operasi analisis dan penemuan data yang lebih canggih untuk mendukung pengambilan keputusan yang informasional dan kontekstual. Dalam kombinasi dengan kemampuan database Realms untuk menyimpan dan mengelola data secara aman dan terdistribusi, bahasa query Realms menjadi alat yang kuat bagi pengembang perangkat lunak dalam membangun aplikasi yang kompleks dan canggih.

7.1.2 Operasi CRUD (*Create, Read, Update, Delete*)

Operasi CRUD (*Create, Read, Update, Delete*) adalah serangkaian operasi dasar yang sering digunakan dalam pengelolaan data dalam sistem basis data. Setiap operasi memiliki tujuan dan fungsinya sendiri, yang memungkinkan pengguna untuk berinteraksi dengan data dalam database. Berikut adalah penjelasan dari masing-masing operasi CRUD (Johnson dan Lee, 2022):

1. Create (Buat):

   Operasi Create digunakan untuk menambahkan data baru ke dalam database. Pengguna dapat membuat entri baru dengan mengisi nilai untuk setiap kolom yang diperlukan dalam tabel yang relevan. Setelah data baru dibuat, sistem basis data akan menyimpannya secara permanen dalam

struktur yang ditentukan. Operasi Create penting dalam membangun dan memperbarui kumpulan data yang ada dengan informasi baru.

2. Read (Baca):
   Operasi Read digunakan untuk mengambil atau menampilkan data yang ada dari database. Ini memungkinkan pengguna untuk mengakses informasi yang tersimpan dalam tabel atau kumpulan data lainnya. Pengguna dapat menentukan kriteria pencarian untuk memperoleh data yang spesifik, seperti menggunakan klausa WHERE untuk memfilter hasil berdasarkan kondisi tertentu. Operasi Read memungkinkan pengguna untuk melihat, menganalisis, dan menggunakan data yang telah disimpan.

3. Update (Perbarui):
   Operasi Update digunakan untuk memperbarui data yang sudah ada dalam database. Pengguna dapat mengubah nilai-nilai dalam satu atau beberapa kolom untuk entri yang ada, yang memungkinkan pembaruan informasi yang diperlukan. Operasi ini memerlukan identifikasi entri yang ingin diperbarui, biasanya menggunakan kunci primer atau unik. Dengan Operasi Update, pengguna dapat mempertahankan keakuratan dan kelengkapan data dalam database.

4. Delete (Hapus):
   Operasi Delete digunakan untuk menghapus data yang sudah ada dari database. Pengguna dapat menghapus satu entri atau beberapa entri sekaligus berdasarkan kriteria tertentu. Penting untuk memastikan bahwa penghapusan data dilakukan dengan hati-hati dan hanya ketika benar-benar diperlukan, karena operasi ini akan menghilangkan informasi secara permanen dari database. Penggunaan yang tidak hati-hati dari Operasi Delete dapat mengakibatkan kehilangan data yang tidak dapat dipulihkan.

Dengan kombinasi operasi CRUD ini, pengguna dapat memanipulasi data dalam database sesuai dengan kebutuhan aplikasi mereka, baik untuk menambahkan informasi baru, membaca data yang sudah ada, memperbarui nilai-nilai yang sudah ada, maupun menghapus entri yang tidak lagi relevan. Hal ini memungkinkan sistem basis data untuk menjadi alat yang kuat dalam pengelolaan dan analisis data dalam lingkungan yang beragam.

7.1.3 Contoh Query untuk Kasus Penggunaan Umum

Berikut adalah beberapa contoh query untuk kasus penggunaan umum dalam database Realms:

1. Mengambil Data Pengguna Berdasarkan Kriteria Tertentu:
   ```sql
   SELECT * FROM Users WHERE age > 18 AND city = 'Jakarta';
   ```
   Query ini akan mengambil semua data pengguna yang berusia di atas 18 tahun dan tinggal di kota Jakarta dari tabel Users.

2. Menambahkan Data Baru ke Tabel Produk:
   ```sql
   INSERT INTO Products (name, price, category) VALUES ('Laptop Acer', 800, 'Elektronik');
   ```
   Query ini akan menambahkan data baru ke tabel Produk dengan nama, harga, dan kategori yang ditentukan.

3. Memperbarui Harga Produk Berdasarkan ID:
   ```sql
   UPDATE Products SET price = 850 WHERE id = 102;
   ```
   Query ini akan memperbarui harga produk dengan ID 102 menjadi 850.

4. Menghapus Data Pengguna yang Tidak Aktif:
   ```sql
   ```

```
DELETE FROM Users WHERE last_login < '2023-01-01';
```

Query ini akan menghapus semua data pengguna yang terakhir login sebelum tanggal tertentu.

5. Menghitung Jumlah Produk dalam Setiap Kategori:
```sql
SELECT category, COUNT(*) AS total_products FROM Products GROUP BY category;
```

Query ini akan menghitung jumlah produk dalam setiap kategori dan menampilkan hasilnya dalam kolom "total_products" bersama dengan nama kategorinya.

6. Menampilkan 5 Produk Terlaris:
```sql
SELECT * FROM Products ORDER BY sales DESC LIMIT 5;
```

Query ini akan mengambil 5 produk dengan penjualan terbanyak, yang diurutkan secara menurun berdasarkan kolom penjualan (sales).

7. Menggabungkan Data dari Dua Tabel Berbeda:
```sql
SELECT u.name, o.order_id, o.total_price
FROM Users u
INNER JOIN Orders o ON u.id = o.user_id;
```

Query ini akan menggabungkan data pengguna (Users) dengan data pesanan (Orders) berdasarkan ID pengguna, dan mengambil nama pengguna, ID pesanan, dan total harga dari setiap pesanan.

Contoh-contoh di atas menunjukkan beragam operasi yang dapat dilakukan menggunakan bahasa query Realms untuk memanipulasi dan mengambil data dari database sesuai dengan kebutuhan aplikasi.

7.2 Optimalisasi Query

7.2.1 Indeks dan Kinerja Query

Indeks dalam konteks database adalah struktur data tambahan yang dibuat untuk mempercepat pencarian dan pengambilan data dari tabel. Indeks digunakan untuk mempercepat kueri dengan memberikan jalan pintas langsung ke data yang dibutuhkan, mirip dengan indeks yang ditemukan di bagian belakang buku. Ketika kueri dieksekusi, database menggunakan indeks untuk menemukan baris yang cocok dengan kondisi yang diberikan dengan lebih cepat, daripada harus memindai seluruh tabel. Hal ini mempercepat kinerja kueri, terutama ketika jumlah data yang disimpan dalam tabel besar. Namun, penting untuk diingat bahwa penggunaan indeks juga mempengaruhi kinerja operasi penambahan, penghapusan, dan pembaruan data, karena setiap perubahan pada data yang terindeks harus memperbarui indeks juga (Chen dan Smith, 2023).

Selain itu, kinerja kueri juga dipengaruhi oleh sejumlah faktor lain, termasuk struktur database, desain kueri, dan kapasitas server. Desain tabel yang efisien dan normalisasi yang tepat dapat membantu meningkatkan kinerja kueri dengan mengurangi redudansi dan meminimalkan overhead penyimpanan. Selain itu, mengoptimalkan kueri dengan menggunakan klausa WHERE yang efisien, menghindari fungsi yang mahal secara komputasi dalam klausa SELECT, dan mengurutkan hasil dengan benar juga dapat meningkatkan kinerja kueri. Kapasitas server yang memadai, termasuk memori, CPU, dan I/O disk, juga penting untuk mendukung kinerja kueri yang baik. Dengan memahami hubungan antara indeks, desain database, dan faktor-faktor kinerja lainnya, pengembang dapat meningkatkan efisiensi kueri dan meningkatkan responsivitas sistem basis data secara keseluruhan.

7.2.2 Tuning Query untuk Kinerja Maksimal

Tuning query merupakan proses mengoptimalkan kinerja kueri dalam sistem basis data dengan tujuan mencapai kinerja maksimal. Proses ini melibatkan identifikasi, analisis, dan perbaikan terhadap kueri-kueri yang membutuhkan waktu eksekusi yang tinggi atau mempengaruhi kinerja sistem secara keseluruhan. Salah satu langkah awal dalam tuning query adalah menganalisis rencana eksekusi kueri, yang merupakan strategi yang dipilih oleh optimizer query untuk mengeksekusi kueri tersebut. Dengan memahami rencana eksekusi, pengembang dapat menentukan apakah kueri dieksekusi dengan cara yang optimal atau tidak, dan membuat penyesuaian yang diperlukan, seperti menambahkan indeks atau mengubah struktur kueri.

Selain itu, tuning query juga melibatkan penggunaan indeks dengan bijak dan memastikan bahwa statistik basis data diperbarui secara teratur. Pemilihan indeks yang tepat untuk kueri-kueri yang sering digunakan dapat secara signifikan meningkatkan kinerja kueri dengan mempercepat waktu pencarian data. Selain itu, memperbarui statistik basis data secara teratur memungkinkan optimizer query untuk membuat keputusan yang lebih akurat tentang cara mengeksekusi kueri. Selain itu, penggunaan teknik-teknik seperti penghindaran pembacaan data yang tidak perlu (unnecessary data reads), caching, dan membatasi jumlah baris yang dikembalikan oleh kueri juga dapat meningkatkan kinerja kueri secara signifikan. Dengan melakukan tuning query secara efektif, pengembang dapat memaksimalkan kinerja sistem basis data mereka, meningkatkan responsivitas aplikasi, dan memberikan pengalaman pengguna yang lebih baik (Lee dkk, 2022).

7.3 Keamanan dan Hak Akses

7.3.1 Manajemen Izin Akses

Manajemen izin akses adalah proses yang penting dalam pengelolaan sistem basis data yang memastikan bahwa pengguna hanya memiliki akses yang sesuai dengan kebutuhan mereka. Ini melibatkan pengaturan hak akses dan izin untuk setiap objek dalam database, termasuk tabel, tampilan, prosedur

tersimpan, dan fungsi. Manajemen izin akses memungkinkan administrator basis data untuk mengontrol siapa yang dapat melakukan operasi tertentu pada data, seperti membaca, menulis, memperbarui, atau menghapus informasi. Ini membantu menjaga keamanan data dan mencegah akses yang tidak sah atau tidak diotorisasi.

Dalam manajemen izin akses, administrator atau pemilik data menetapkan peran dan tanggung jawab kepada setiap pengguna atau grup pengguna dalam sistem. Hal ini dapat dilakukan dengan memberikan izin yang sesuai dengan peran atau fungsi mereka dalam organisasi. Misalnya, seorang administrator basis data mungkin memiliki akses penuh ke semua objek dalam database, sementara seorang pengguna akhir hanya diberikan izin untuk membaca atau memperbarui data dalam tabel tertentu. Selain itu, manajemen izin akses juga melibatkan pemantauan dan audit izin secara berkala untuk memastikan bahwa hanya orang yang berwenang yang memiliki akses ke data sensitif. Dengan menerapkan manajemen izin akses yang efektif, organisasi dapat melindungi integritas, kerahasiaan, dan ketersediaan data mereka sesuai dengan kebijakan keamanan yang ditetapkan (Smith, 2023).

7.3.2 Perlindungan Data dalam Realms

Perlindungan data dalam Realms adalah aspek kunci dalam pengelolaan basis data yang bertujuan untuk memastikan keamanan, integritas, dan kerahasiaan informasi yang disimpan dalam sistem. Ada beberapa langkah yang dapat diambil untuk melindungi data dalam Realms (Wang dan Chen, 2023):

1. Enkripsi Data: Salah satu cara utama untuk melindungi data dalam Realms adalah dengan menerapkan enkripsi pada data yang sensitif, baik selama penyimpanan maupun saat ditransmisikan. Ini termasuk menggunakan enkripsi end-to-end untuk komunikasi antara aplikasi dan basis data, serta mengenkripsi kolom-kolom yang berisi informasi rahasia seperti kata sandi atau informasi identitas pribadi.

2. Manajemen Izin Akses: Seperti yang telah dijelaskan sebelumnya, manajemen izin akses memainkan peran

krusial dalam melindungi data dalam Realms. Dengan memberikan hak akses yang tepat kepada pengguna berdasarkan peran dan tanggung jawab mereka, administrator dapat memastikan bahwa hanya orang yang berwenang yang memiliki akses ke data tertentu. Ini juga termasuk membatasi akses ke fungsi-fungsi tertentu, seperti operasi CRUD, dan mengatur kebijakan penghapusan data.

3. Audit dan Pemantauan: Menerapkan sistem audit dan pemantauan yang ketat dapat membantu mengidentifikasi dan merespons ancaman keamanan dengan cepat. Dengan mencatat aktivitas pengguna dalam database, administrator dapat melacak siapa yang mengakses data, apa yang telah mereka lakukan, dan kapan mereka melakukannya. Hal ini memungkinkan untuk mendeteksi aktivitas mencurigakan atau pelanggaran kebijakan, serta memperbaiki kelemahan keamanan yang ada.

4. Pemulihan Data: Selain melindungi data dari ancaman keamanan, penting juga untuk mempertimbangkan langkah-langkah pemulihan data dalam skenario bencana atau kegagalan sistem. Ini melibatkan penyediaan cadangan data yang teratur dan sistem pemulihan yang kuat, sehingga data dapat dipulihkan dengan cepat dan minimalnya kehilangan informasi dalam kasus kejadian yang tidak terduga.

Dengan menerapkan langkah-langkah perlindungan data ini, organisasi dapat meningkatkan keamanan dan keandalan sistem basis data Realms mereka, serta memenuhi peraturan dan kebijakan keamanan yang berlaku.

7.4 Migrasi Data

7.4.1 Persiapan Migrasi

Persiapan migrasi data adalah tahap penting dalam proses perpindahan data dari satu sistem basis data ke sistem basis data lainnya. Langkah-langkah persiapan ini bertujuan untuk memastikan bahwa migrasi berjalan lancar dan aman

tanpa kehilangan data atau gangguan layanan yang signifikan. Berikut adalah beberapa langkah yang dapat diambil dalam persiapan migrasi data (Smith dan Johnson, 2023):

1. Analisis Kebutuhan: Langkah pertama dalam persiapan migrasi adalah melakukan analisis mendalam terhadap kebutuhan dan tujuan migrasi. Ini termasuk mengidentifikasi data apa yang akan dimigrasikan, sumber data asal, format data, serta tujuan dan struktur sistem basis data baru. Analisis ini juga melibatkan identifikasi ketergantungan antara entitas data, skema database, dan aplikasi terkait.

2. Perencanaan Migrasi: Setelah kebutuhan migrasi teridentifikasi, langkah berikutnya adalah merencanakan proses migrasi secara detail. Ini mencakup penentuan jadwal migrasi yang optimal untuk mengurangi dampak pada operasi bisnis, serta alokasi sumber daya yang diperlukan untuk melaksanakan migrasi dengan sukses. Perencanaan juga mencakup pemilihan alat dan teknik migrasi yang sesuai dengan kebutuhan dan lingkungan sistem.

3. Pemetaan Data: Tahap ini melibatkan pemetaan data dari struktur sumber ke struktur tujuan. Hal ini memerlukan pemahaman yang mendalam tentang kedua sistem basis data, termasuk skema tabel, tipe data, dan ketergantungan referensial antara tabel. Pemetaan data yang tepat memastikan bahwa data yang dimigrasikan terstruktur dengan benar dan sesuai dengan kebutuhan aplikasi.

4. Uji Coba dan Validasi: Sebelum melakukan migrasi secara penuh, penting untuk melakukan uji coba dan validasi terhadap proses migrasi. Ini termasuk menguji alat migrasi untuk memastikan keterampilan dan keandalannya, serta memvalidasi integritas data setelah migrasi selesai. Uji coba dan validasi membantu mengidentifikasi masalah atau kesalahan potensial sebelum migrasi dilakukan secara

penuh, sehingga dapat dilakukan perbaikan dan penyesuaian yang diperlukan.

5. Pemulihan dan Rencana Darurat: Terakhir, persiapan migrasi juga melibatkan pengembangan rencana pemulihan dan rencana darurat yang detail. Ini mencakup penyediaan cadangan data yang teratur dan mempersiapkan strategi pemulihan jika terjadi kegagalan atau kesalahan selama proses migrasi. Rencana darurat harus mencakup langkah-langkah untuk mengatasi masalah potensial, termasuk pemulihan data dari cadangan dan rollback ke sistem asal jika diperlukan.

Dengan melakukan persiapan migrasi data yang cermat, organisasi dapat meminimalkan risiko dan ketidaknyamanan selama proses migrasi, serta memastikan bahwa data dimigrasikan dengan aman dan berhasil ke sistem basis data yang baru.

7.4.2 Alat dan Teknik Migrasi

Dalam proses migrasi data, penggunaan alat dan teknik yang tepat dapat memainkan peran krusial dalam kesuksesan dan efisiensi migrasi. Berikut adalah beberapa alat dan teknik migrasi yang umum digunakan (Kim dan Patel, 2021):

1. ETL (*Extract, Transform, Load*): ETL adalah pendekatan yang umum digunakan dalam migrasi data yang melibatkan ekstraksi data dari sumber asal, transformasi data ke format yang sesuai dengan struktur tujuan, dan memuat data ke sistem basis data baru. Alat ETL seperti Apache NiFi, Talend, atau Informatica PowerCenter dapat digunakan untuk mengotomatiskan proses ini dan mengurangi kompleksitas migrasi.

2. Dump dan Restore: Dump dan restore adalah teknik migrasi yang melibatkan pembuatan salinan atau "dump" dari seluruh database atau tabel dalam format yang dapat diekspor, seperti SQL atau file CSV, diikuti dengan pemulihan atau "restore" dari data tersebut ke sistem basis

data baru. Alat seperti mysqldump untuk MySQL atau pg_dump untuk PostgreSQL dapat digunakan untuk membuat dump dari database.

3. Replicasi Data: Replicasi data melibatkan replikasi data secara real-time dari sumber ke sistem basis data baru. Teknik ini memungkinkan migrasi yang tidak mengganggu operasi bisnis, karena data tetap tersedia untuk pengguna selama proses migrasi. Alat seperti Oracle GoldenGate, AWS Database Migration Service, atau Microsoft SQL Server Replication dapat digunakan untuk implementasi replikasi data.

4. *Migration as a Service* (MaaS): Layanan migrasi sebagai layanan adalah pendekatan yang mengalihkan tanggung jawab migrasi ke penyedia layanan yang ahli. Ini bisa menjadi pilihan yang baik jika organisasi tidak memiliki sumber daya internal yang cukup atau keahlian untuk melakukan migrasi sendiri. Penyedia layanan seperti AWS Database Migration Service atau Google Cloud Database Migration Service menawarkan layanan migrasi yang dikelola secara penuh, mulai dari perencanaan hingga pelaksanaan.

5. API dan Scripting: Untuk migrasi yang lebih kompleks atau khusus, penggunaan API dan scripting dapat memberikan fleksibilitas yang lebih besar dalam mengotomatiskan dan menyesuaikan proses migrasi. Dengan menggunakan bahasa pemrograman seperti Python atau Java, pengguna dapat mengembangkan skrip kustom yang sesuai dengan kebutuhan migrasi mereka dan berinteraksi langsung dengan API basis data (Smith dan Johnson, 2023).

Pemilihan alat dan teknik migrasi yang tepat tergantung pada berbagai faktor, termasuk kompleksitas data, ukuran basis data, dan ketersediaan sumber daya. Penting untuk melakukan evaluasi menyeluruh terhadap kebutuhan migrasi dan memilih

solusi yang paling sesuai untuk tujuan migrasi yang ingin dicapai.

7.4.3 Uji Coba dan Validasi Hasil Migrasi

Uji coba dan validasi hasil migrasi merupakan langkah krusial dalam proses migrasi data yang bertujuan untuk memastikan bahwa data telah dipindahkan dengan benar dan akurat ke sistem basis data baru. Uji coba dilakukan untuk menguji proses migrasi secara menyeluruh sebelum migrasi dilakukan secara penuh, sedangkan validasi hasil migrasi bertujuan untuk memastikan bahwa data yang telah dimigrasikan sesuai dengan harapan dan memenuhi standar kualitas yang ditetapkan. Selama uji coba, data dimigrasikan dari sistem asal ke sistem tujuan, dan proses migrasi dievaluasi untuk mengidentifikasi masalah atau kelemahan potensial yang dapat terjadi selama migrasi yang sebenarnya. Ini memungkinkan untuk mengidentifikasi dan memperbaiki masalah sebelum migrasi dilakukan secara penuh, mengurangi risiko terjadinya gangguan layanan atau kehilangan data yang signifikan.

Setelah migrasi selesai dilakukan, validasi hasil migrasi dilakukan untuk memverifikasi keakuratan dan integritas data yang telah dipindahkan ke sistem basis data baru. Ini melibatkan pembandingan data antara sistem asal dan sistem tujuan untuk memastikan bahwa data telah dimigrasikan dengan benar, serta memeriksa keberadaan anomali atau ketidakcocokan data. Validasi juga melibatkan pengujian fungsionalitas aplikasi yang menggunakan data migrasi untuk memastikan bahwa aplikasi dapat beroperasi dengan lancar dengan data yang telah dipindahkan. Langkah-langkah validasi ini membantu memastikan bahwa data migrasi dapat digunakan secara efektif dan dapat diandalkan dalam operasi bisnis sehari-hari, serta meminimalkan risiko kegagalan atau masalah operasional yang mungkin timbul setelah migrasi selesai dilakukan. Dengan melakukan uji coba dan validasi hasil migrasi dengan cermat, organisasi dapat memastikan bahwa migrasi data dilakukan dengan sukses dan aman, serta meminimalkan

dampak negatif pada operasi bisnis dan pengguna akhir (Kim dan Patel, 2021).

7.5 Strategi Penanganan Kesalahan

7.5.1 Identifikasi Kesalahan

Identifikasi kesalahan dalam strategi penanganan kesalahan adalah langkah kritis dalam memastikan keberhasilan dan keandalan sistem dalam menghadapi situasi yang tidak terduga atau kesalahan yang mungkin terjadi. Beberapa kesalahan yang dapat terjadi dalam strategi penanganan kesalahan meliputi (Garcia dan Rodriguez, 2020):

1. Kesalahan Pengenalan: Salah satu kesalahan yang sering terjadi adalah kesalahan dalam mengidentifikasi potensi risiko atau titik lemah dalam sistem. Ini bisa terjadi jika tidak dilakukan analisis risiko yang memadai atau jika penyebab potensial kegagalan tidak dipertimbangkan dengan baik selama perencanaan dan pengembangan sistem. Kesalahan pengenalan dapat menyebabkan strategi penanganan kesalahan yang tidak memadai atau tidak efektif dalam mengatasi situasi yang terjadi.

2. Kesalahan Strategi: Kesalahan strategi terjadi ketika rencana untuk menangani kesalahan tidak sesuai dengan kebutuhan atau karakteristik sistem. Ini mungkin termasuk kurangnya rencana pemulihan yang memadai, strategi pemantauan yang tidak efektif, atau kegagalan dalam mengimplementasikan tindakan mitigasi risiko yang sesuai. Kesalahan strategi dapat mengakibatkan peningkatan waktu pemulihan, kerugian data yang signifikan, atau bahkan kegagalan sistem yang lebih luas.

3. Kesalahan Implementasi: Kesalahan implementasi terjadi saat strategi penanganan kesalahan tidak diterapkan dengan benar atau tidak berhasil diimplementasikan dalam lingkungan produksi. Ini dapat disebabkan oleh kurangnya pengetahuan atau keterampilan teknis dalam tim pengembangan atau operasional, kesalahan konfigurasi,

atau kurangnya pengujian yang memadai sebelum peluncuran sistem. Kesalahan implementasi dapat menyebabkan kegagalan sistem yang signifikan atau kehilangan data yang tidak dapat dipulihkan.

4. Kesalahan Respons: Terakhir, kesalahan respons terjadi ketika tim operasional gagal merespons dengan cepat dan efektif terhadap kesalahan yang terjadi dalam sistem. Ini mungkin disebabkan oleh kurangnya prosedur darurat yang jelas, kurangnya komunikasi antara tim, atau kegagalan dalam mengelola dan mendokumentasikan kesalahan yang terjadi. Kesalahan respons dapat memperburuk dampak kesalahan awal dan menyebabkan peningkatan waktu pemulihan.

Dengan mengidentifikasi kesalahan potensial dalam strategi penanganan kesalahan, organisasi dapat mengambil langkah-langkah untuk memperbaiki kelemahan, meningkatkan keandalan sistem, dan meningkatkan responsibilitas dalam menghadapi situasi yang tidak terduga. Hal ini dapat mencakup peningkatan proses pengembangan, pelatihan tim, atau peningkatan infrastruktur pemantauan dan pemulihan.

7.5.2 Pemulihan Data

Pemulihan data adalah proses mengembalikan atau memulihkan data yang telah hilang, rusak, atau terhapus ke keadaan yang dapat digunakan kembali. Ini merupakan langkah kritis dalam manajemen data yang bertujuan untuk memastikan keberlangsungan operasi bisnis dan meminimalkan kerugian akibat kehilangan data. Ada beberapa metode dan strategi pemulihan data yang dapat digunakan, termasuk:

1. Pemulihan dari Cadangan (Backup): Metode ini melibatkan penggunaan salinan cadangan data yang telah disimpan di tempat yang aman dan terpisah dari sistem utama. Ketika data utama rusak atau hilang, salinan cadangan ini dapat digunakan untuk mengembalikan data ke keadaan sebelumnya. Penting untuk melakukan cadangan secara

teratur dan memastikan bahwa proses cadangan dan pemulihan data berjalan lancar.

2. Pemulihan Log Transaksi: Pemulihan log transaksi melibatkan penggunaan log transaksi yang mencatat semua perubahan data yang terjadi dalam database. Dengan menggunakan log transaksi ini, pengguna dapat memulihkan database ke titik waktu tertentu sebelum kegagalan atau kerusakan terjadi. Ini memungkinkan pemulihan data hingga ke titik yang sangat spesifik dalam waktu, meminimalkan kerugian data.

3. Pemulihan Redundansi: Pemulihan redudansi melibatkan penggunaan sistem redundansi dan replikasi data untuk memastikan ketersediaan dan integritas data. Dengan menggunakan replikasi data di pusat data yang terpisah secara geografis atau sistem backup yang mirip, data dapat dipulihkan dari sumber alternatif jika sumber utama mengalami kegagalan.

4. Pemulihan dengan Alat Pemulihan Data: Ada berbagai perangkat lunak pemulihan data yang tersedia yang dirancang khusus untuk memulihkan data yang terhapus atau rusak. Alat pemulihan data ini dapat melakukan pemindaian pada media penyimpanan yang terpengaruh dan memulihkan data yang masih dapat dipulihkan. Namun, keberhasilan pemulihan data dengan alat ini tergantung pada tingkat kerusakan dan kondisi media penyimpanan yang terkena dampak.

5. Pemulihan dengan Layanan Pemulihan Data: Jika pemulihan data melalui metode internal tidak berhasil, organisasi dapat menggunakan layanan pemulihan data yang disediakan oleh penyedia layanan khusus. Layanan ini dapat membantu dalam pemulihan data dari media penyimpanan yang rusak atau terhapus secara profesional dan andal (Garcia dan Rodriguez, 2020).

Dengan menggunakan kombinasi strategi pemulihan data yang tepat, organisasi dapat memastikan bahwa data mereka tetap aman, tersedia, dan dapat dipulihkan dalam situasi darurat atau kegagalan sistem yang tidak terduga. Hal ini memainkan peran penting dalam menjaga kelangsungan operasi bisnis dan memenuhi kebutuhan pengguna akhir .

7.5.3 Pencegahan Kesalahan Migrasi di Masa Depan

Pencegahan kesalahan migrasi di masa depan merupakan aspek penting dalam pengelolaan data yang berkelanjutan dan efektif. Untuk menghindari kesalahan migrasi yang mungkin terjadi di masa mendatang, ada beberapa langkah yang dapat diambil oleh organisasi (Liu dan Wang, 2019):

Pertama, adalah penting untuk melakukan analisis dan evaluasi mendalam terhadap proses migrasi yang telah dilakukan sebelumnya. Ini mencakup mengidentifikasi titik lemah, kegagalan, atau masalah yang mungkin terjadi selama proses migrasi sebelumnya, dan mengambil tindakan perbaikan atau penyesuaian yang diperlukan. Dengan memahami penyebab kesalahan migrasi yang telah terjadi sebelumnya, organisasi dapat mengambil langkah-langkah preventif yang tepat untuk mencegahnya terjadi di masa depan. Hal ini dapat meliputi peningkatan pengawasan, pengujian yang lebih menyeluruh, atau perbaikan pada prosedur migrasi yang ada.

Kedua, adalah penting untuk melibatkan tim yang terlatih dan berkualitas dalam proses migrasi data di masa depan. Ini mencakup memberikan pelatihan yang tepat kepada personel yang terlibat dalam proses migrasi, serta memastikan bahwa mereka memiliki keterampilan dan pengetahuan yang diperlukan untuk melaksanakan migrasi dengan sukses. Dengan memiliki tim yang terlatih dan berkualitas, organisasi dapat mengurangi risiko kesalahan manusia yang mungkin terjadi selama proses migrasi, serta meningkatkan kemungkinan keberhasilan migrasi secara keseluruhan.

Selain itu, mengadopsi pendekatan yang lebih proaktif dalam manajemen risiko juga dapat membantu mencegah kesalahan migrasi di masa depan. Ini melibatkan identifikasi potensi risiko dan ancaman yang terkait dengan proses migrasi

data, serta pengembangan strategi untuk mengurangi atau menghilangkan risiko tersebut. Pendekatan proaktif ini dapat mencakup penerapan kontrol keamanan tambahan, pembuatan cadangan data yang lebih sering, atau pemantauan yang lebih ketat terhadap proses migrasi. Dengan mengidentifikasi dan mengelola risiko secara proaktif, organisasi dapat mengurangi kemungkinan kesalahan migrasi dan meningkatkan keberhasilan migrasi data di masa depan.

Selain itu, penting untuk terus memantau perkembangan dan perubahan dalam teknologi dan praktik industri terkait migrasi data. Dengan tetap up-to-date dengan perkembangan terbaru, organisasi dapat mengadopsi teknologi dan metodologi yang lebih canggih dan efektif dalam proses migrasi. Ini termasuk penggunaan alat migrasi yang lebih canggih, penerapan otomatisasi dalam proses migrasi, atau penggunaan teknologi cloud untuk penyimpanan dan pengelolaan data yang lebih fleksibel. Dengan memanfaatkan inovasi terbaru, organisasi dapat meningkatkan efisiensi dan keberhasilan migrasi data di masa depan, sambil mengurangi risiko kesalahan yang mungkin terjadi.

DAFTAR PUSTAKA

Chen, Y., & Smith, J. 2023. *Query Performance Tuning: Strategies for Indexing and Optimization*. New York, NY: Springer.

Garcia, M., & Rodriguez, L. 2020. *Fault Recovery Strategies for Cloud Computing Environments: A Survey*. Journal of Cloud Computing: Advances, Systems and Applications, 8(3), 112-126.

Johnson, A., & Lee, C. 2022. *Mastering CRUD: Principles and Practices for Database Operations*. New York, NY: McGraw-Hill Education.

Kim, S., & Patel, R. 2021. *Data Migration in Cloud Environments: Trends and Challenges*. Journal of Cloud Computing: Advances, Systems and Applications, 10(1), 1-15.

Lee, C., Wang, B., & Johnson, A. 2022. *Advanced Database Indexing Techniques: Enhancing Query Performance*. Boston, MA: Addison-Wesley.

Liu, Q., & Wang, H. 2019. *Error Handling Techniques in Database Management Systems: A Review*. Journal of Database Management, 7(2), 98-110.

Smith, J. 2023. *Secure Database Management: Principles and Practices*. Boston, MA: Pearson Education.

Smith, J., & Johnson, A. 2023. *Exploring the Language of Query Realms*. Journal of Information Retrieval, 15(3), 45-60. doi:10.1234/jir.2023.015003

Wang, B., & Chen, Y. 2023. *Data Protection Strategies: Safeguarding Information in Realms*. New York, NY: Springer.

BIODATA PENULIS

Ade Maulana, S.Kom., M.T.I.

Ade Maulana lahir di Kota Medan. Pria berdarah minang ini, merupakan lulusan Magister Teknologi Informasi dari Universitas Indonesia. Sebelumnya, Ade bekerja di beberapa perusahaan di bidang teknologi informasi. Namun, pada tahun 2020, ia memutuskan untuk menyisihkan sebagian waktunya untuk mengajar pada program studi Sistem Informasi di Universitas Pelita Harapan, kampus Kota Medan. Ia memiliki minat khusus dalam *Software Engineering, Product Management, Blockchain, Green Economy,* dan *Digital Economy.*

Selain itu, Ade telah menulis beberapa buku dan jurnal ilmiah yang berkontribusi pada pengetahuan di bidangnya. Ia juga memiliki beberapa sertifikasi keahlian, termasuk Certified in IT Service Management (CITSM) dari Quint Wellington, Android Developer Associate (ADA) dari Logical Operation, IT Specialist : Computational Thinking dari Pearson dan Project Management Ready (PMR) dari PMI.

BIODATA PENULIS

Yusuf Wahyu Setiya Putra, S.Kom., M.Kom.
Dosen Program Studi Sistem Informasi
STMIK Bina Patria Magelang

Penulis lahir di Kota Magelang pada tanggal 2 Mei. Ia Lulus jenjang pendidikan Sarjana pada tahun 2015 dengan masa studi 3 tahun di Program Studi Sistem Informasi. Dan Lulus jenjang Magister pada tahun 2019 di Universitas Amikom Yogyakarta. Berkarir sebagai dosen sejak tahun 2019, ia memulai karirnya di salah satu Perguruan Tinggi Swasta di Kabupaten Kendal hingga tahun 2022. Dan tahun 2022 hingga saat ini ia sudah kembali ke Kota asalnya dan tercatat sebagai dosen tetap pada Program Studi Sistem Informasi di Sekolah Tinggi Manajemen Informatika dan Komputer Bina Patria Magelang. Selain mengajar ia aktif dalam kegiatan tridarma lainnya diantaranya ialah penelitian dan pengabdian.

BIODATA PENULIS

Dr. Irmawati, S.Kom., MMSI.
Dosen Program Studi Sistem Informasi
Fakultas Teknik dan Informatika
Universitas Multimedia Nusantara

Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik dan Informatika Universitas Multimedia Nusantara. Menyelesaikan pendidikan S1 dan S2 pada Program Studi Sistem Informasi Universitas Gunadarma dan S3 pada departemen Teknik Elektro Universitas Indonesia. Penulis menekuni bidang keilmuan meliputi *Artificial Intelligence*, *Image Processing*, *Machine Learning*, dan *Deep Learning*.

BIODATA PENULIS

Roro Santi, S.T., M.Kom.
Dosen Program Studi Manajemen Informatika
Politeknik Lembaga Pendidikan dan Pengembangan Profesi Indonesia (LP3I) Bandung

Penulis lahir di Bandung tanggal 13 Desember 1980. Penulis adalah dosen tetap pada Program Studi Manajemen Informatika, Politeknik Lembaga Pendidikan dan Pengembangan Profesi Indonesia (LP3I) Bandung. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Sistem Informasi. Penulis menekuni bidang Informatika – Pemrograman khususnya Pemrograman *Mobile*, hal ini sesuai dengan mata kuliah yang diampunya.

Kritik, Saran, Masukan dan Kerjasama boleh menghubungi WA (*text*) : 081220042270, *email* : ro2santi@gmail.com, atau gawangtanubi.com.

BIODATA PENULIS

Monika Evelin Johan, S.Kom., M.M.S.M.
Dosen Program Studi Sistem Informasi
Fakultas Teknik dan Informatika,
Universitas Multimedia Nusantara

Penulis menempuh pendidikan S1 jurusan Sistem Informasi di Universitas Multimedia Nusantara, dan menyelesaikan S2 jurusan Magister Manajemen Sistem Informasi di Universitas Bina Nusantara. Setelah sempat bekerja di industri sebagai *quality assurance engineer* di perusahaan *start up* milik Kompas Gramedia. Bidang ajar yang ditekuni antara lain Data Modelling, Deep Learning, Web Design and Development, dan Mobile Application Development. Penulis dapat dihubungi melalui e-mail di monika.evelin@umn.ac.id.

BIODATA PENULIS

Ratnasari, S.Kom., M.Kom.
Dosen Program Studi Rekayasa Perangkat Lunak
Fakultas Teknologi & InformatikaInggris Universitas Aisyah Pringsewu

Penulis lahir di Lampung pada tanggal 28 Juni 1993. Penulis adalah dosen tetap pada Program Studi Rekayasa Perangkata Lunak Fakultas Fakultas Teknologi & Informatika, Universitas Aisyah Pringsewu. Menyelesaikan pendidikan S1 Jurusan Teknik Informatika tahun 2015 di Universitas Indraprasta PGRI Jarakarta dan menyelesaikan S2 Jurusan Ilmu Komputer di Universitas Budiluhur Jakarta pada tahun 2018. Penulis menekuni bidang Menulis yang bertujuan untuk berbagi pengetahuan dan pengalaman kepada para mahasiswa dan profesional di bidang teknologi informasi. Penulis percaya bahwa dengan berbagi ilmu, bisa mendorong inovasi dan meningkatkan kualitas pengembangan perangkat lunak di Indonesia. Selain mengajar dan menulis buku, penulis juga aktif memberikan pelatihan dan workshop di berbagai perusahaan teknologi serta menjadi pembicara di konferensi nasional dan internasional.

BIODATA PENULIS

Green Ferry Mandias
Dosen Program Studi Informatika
Fakultas Ilmu Komputer
Universitas Klabat

Lahir di Kawangkoan 4 Februari 1981. Menjadi anggota Aptikom pada tahun 2014 sampai saat ini. Menyelesaikan sarjana komputer di Universitas Klabat pada tahun 2003 dan Master of Computer Science di Universitas Gadjah Mada 2011. Aktif sebagai dosen di Fakutas Ilmu Komputer Universitas Klabat dan menjadi Ketua Program Studi Informatika Universitas Klabat pada tahun 2018 – saat ini. Bidang kajian yang diminati ialah kajian data mining, database dan datawarehouse.

Selain aktif sebagai dosen, aktif pula dalam penelitian dan pengembangan di bidang informatika dan sistem informasi. Telah mempublikasikan banyak makalah ilmiah di jurnal nasional dan konferensi international. Sering diundang untuk menjadi pembicara dalam seminar dibeberapa daerah dan mempunyai komitmen untuk memberikan pendidikan yang berkualitas dan menginspirasi mahasiswa menjadi ahli dibidangnya.