SKRIPSI

PERANCANGAN APLIKASI BERBAGI INFORMASI SEPUTAR PERKULIAHAN ANTAR MAHASISWA PADA *SMARTPHONE* BERBASIS ANDROID

Disusun dan diajukan oleh:

ACHMAD CHAEDAR

D421 15 310

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2021

LEMBAR PENGESAHAN SKRIPSI

PERANCANGAN APLIKASI BERBAGI INFORMASI SEPUTAR PERKULIAHAN ANTAR MAHASISWA PADA *SMARTPHONE* BERBASIS ANDROID

Disusun dan diajukan oleh

ACHMAD CHAEDAR

D421 15 310

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika Fakultas Teknik Universitas Hasanuddin pada tanggal 17 Maret 2021
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping
Dr. Ir. Ingrid Nurtanio, M.T.	A. Ais Prayogi Alimuddin, S.T., M.Eng
Nip. 19610813 198811 2 001	Nip. 19830510 201404 1 001

Ketua Program Studi,

Dr. Amil Ahmad Ilham, S.T., M.IT
Nip. 19731010 199802 1 001

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: ACHMAD CHAEDAR
NIM	: D421 15 310
Program Studi	: TEKNIK INFORMATIKA
Jenjang	: S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

PERANCANGAN APLIKASI BERBAGI INFORMASI SEPUTAR PERKULIAHAN ANTAR MAHASISWA PADA *SMARTPHONE* BERBASIS ANDROID

Adalah karya tulis saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 23 Maret 2021

Yang Menyatakan

ACHMAD CHAEDAR

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya sehingga tugas akhir yang berjudul "Perancangan Aplikasi Berbagi Informasi Seputar Perkuliahan antar Mahasiswa pada Smartphone Berbasis Android" ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari bahwa dalam penyusunan dan penulisan laporan tugas akhir ini tidak lepas dari bantuan, bimbingan serta dukungan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan tugas akhir. Oleh karena itu, penulis dengan senang hati menyampaikan terima kasih kepada:

1) Tuhan Yang Maha Esa atas semua berkat, karunia serta pertolongan-Nya yang tiada batas, yang telah diberikan kepada penulis disetiap langkah dalam pembuatan program hingga penulisan laporan skripsi ini;

2) Kedua orang tua penulis, Bapak Ir. Baharuddin Nur dan Ibu Dr. Hj. Sitti Rabiah, M.Hum., saudara-saudara penulis yaitu Achmad Zulfikar, Achmad Zulakbar, dan Achmad Kautsar serta keluarga yang senantiasa memberikan kekuatan, inspirasi, motivasi, bimbingan moral, materi, kepercayaan, dan kasih sayang yang tidak terbatas kepada penulis;

3) Ibu Dr. Ir. Ingrid Nurtanio, M.T., selaku pembimbing I yang telah banyak memberi bimbingan, inspirasi, motivasi, dan masukan yang bermanfaat kepada penulis;

4) Bapak A. Ais Prayogi Alimuddin, S.T., M.Eng., selaku pembimbing II yang telah banyak memberi keyakinan, perhatian, bimbingan, motivasi, dan masukan yang bermanfaat kepada penulis;

5) Bapak Dr. Indrabayu, S.T., M.T., M.Bus.Sys., dan Bapak Iqra Aswad, S.T., M.T. selaku dosen penguji yang telah memberikan saran sehingga laporan skripsi ini menjadi lebih baik;

6) Bapak Dr. Amil Ahmad Ilham, S.T., M.IT., Ph.D., selaku Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin atas bantuan dan bimbingannya selama masa perkuliahan penulis;

7) Ibu Dr. Ir. Ingrid Nurtanio, M.T., selaku dosen pembimbing akademik yang telah memberikan bimbingan selama masa perkuliahan penulis;

8) Bapak Robert dan Bapak Zainuddin serta segenap staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran penyelesaian tugas akhir penulis;

9) Teman-teman Hypervisor FT-UH atas dukungan dan semangat yang diberikan selama ini;

10) Diri penulis sendiri atas pencapaian, kerja keras, dan semangat pantang menyerah dalam menyelesaikan skripsi dan studi.

11) Serta seluruh pihak yang tidak sempat disebutkan satu persatu yang telah banyak meluangkan tenaga, waktu, dan pikiran selama penyusunan laporan tugas akhir ini.

Akhir kata, penulis berharap semoga Tuhan Yang Maha Esa berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu selanjutnya. Aamiin.

Wassalam

Makassar, Maret 2021

Penulis

ABSTRAK

Smartphone berbasis Android dinilai sudah cukup baik dalam menunjang aktivitas perkuliahan mahasiswa. Namun, aplikasi-aplikasi yang telah ada seperti sosial media yang dinilai masih terbatas dalam menunjang penyebaran informasi antar mahasiswa. Oleh sebab itu, dibutuhkan aplikasi yang dapat menunjang penyebaran informasi antar mahasiswa, terutama informasi perkuliahan. Dengan memanfaatkan *Model-View-ViewModel* (MVVM), *Feature Driven Development* (FDD), serta *Automation Testing*, pengembangan aplikasi berbagi informasi seputar perkuliahan antar mahasiswa pada *smartphone* berbasis Android dapat dilakukan dengan cepat, terstruktur, dan mudah untuk dilakukan pemeliharaan. Subjek penelitian yang terlibat adalah mahasiswa Departemen Teknik Informatika Universitas Hasanuddin sebanyak 20 orang. Objek yang akan diteliti adalah aspek kelayakan aplikasi, desain aplikasi, kemudahan dalam penggunaan, manfaat, hingga keseluruhan aplikasi. Aplikasi yang dikembangkan adalah fitur grup dengan akses pada informasi yang terkelompokkan menjadi 4 yaitu mata kuliah, kontak dosen, tugas, serta informasi yang relevan dengan perkuliahan seperti beasiswa dan lain sebagainya. Informasi pada grup dikelola secara kolaboratif dengan sistem keanggotaan admin grup & anggota grup. Pengujian aplikasi yang dilakukan secara otomatis dan sistematis dengan metode *white box testing* dan *black box testing* berhasil mencakup keseluruhan *internal* aplikasi dan fitur dengan tingkat keberhasilan pengujian sebesar 100%. Hasil analisis kerja sistem dinilai oleh penguji dengan sistem skor skala 1 sampai dengan 10 dan aplikasi yang dikembangkan berhasil mendapatkan rata-rata nilai dari kelima aspek sebesar 8,51 dengan kategori sangat layak.

Kata Kunci : android, penyebaran informasi, mahasiswa, *feature driven development, automation testing*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Peran teknologi komunikasi saat ini menjadi sangat penting karena banyaknya tuntutan kebutuhan akan pertukaran informasi yang cepat dan tepat. Teknologi komunikasi yang berkembang saat ini telah memungkinkan manusia untuk terhubung satu sama lain tanpa dibatasi oleh jarak, ruang, dan waktu. Penyatuan berbagai fungsi dari alat-alat komunikasi telah menyatu dalam sebuah alat komunikasi yang bernama *smartphone*. Smartphone merupakan telepon seluler dengan kemampuan lebih mulai dari resolusi, fitur, hingga komputasi termasuk adanya sistem operasi *mobile* di dalamnya.

Berdasarkan data dari StatCounter menunjukkan bahwa sistem operasi Android adalah sistem operasi smartphone yang paling banyak digunakan di Indonesia dimulai pada tahun 2014. Pada akhir tahun itu, Android menguasai pangsa pasar hampir 60 persen. Sedangkan pada akhir tahun 2015, jumlah pengguna Android naik menjadi 74 persen. Pada tahun 2018, Android stabil meningkat pada 89% hingga angka 92% pada tahun 2019. Kemudian menurut penelitian yang dilakukan oleh Daeng dkk. (2017), *smartphone* berbasis Android dinilai sudah cukup baik dalam menunjang aktivitas perkuliahan mahasiswa. Namun, aplikasi-aplikasi yang telah ada seperti sosial media yang dinilai masih terbatas dalam menunjang penyebaran informasi antar mahasiswa.

Pada penelitian ini dijelaskan bagaimana mengembangkan sebuah aplikasi yang dapat digunakan oleh mahasiswa untuk membagikan dan mendapatkan informasi secara efisien khususnya dalam hal perkuliahan. Gambaran sistem yang ingin diciptakan yaitu sebuah papan informasi dalam bentuk *digital* yang dapat diisi secara kolaboratif oleh sesama mahasiswa. Untuk meningkatkan efisiensi dalam penyimpanan informasi yang ada, terdapat 4 wadah informasi yaitu jadwal kuliah, daftar tugas, daftar kontak dosen, maupun daftar informasi lainnya yang terkait tentang perkuliahan.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah pada tugas akhir ini adalah:

1. Bagaimana cara mengembangkan aplikasi berbagi informasi seputar perkuliahan antar mahasiswa pada *smartphone* berbasis Android yang tepat guna?
2. Bagaimana cara mengukur kelayakan aplikasi berbagi informasi seputar perkuliahan antar mahasiswa yang dikembangkan?

1.3 Tujuan Penelitian

Penelitian ini bertujuan :

1. Untuk menciptakan aplikasi berbagi informasi seputar perkuliahan bagi mahasiswa yang lebih tepat guna serta efisien dari aplikasi berbagi informasi yang telah ada sebelumnya.

2. Untuk meningkatkan kualitas belajar mahasiswa dalam menjalani perkuliahan dengan meringankan beban memori mahasiswa dalam mengingat segala informasi yang terkait perkuliahan.

1.4 Manfaat Penelitian

Manfaat dari tugas akhir ini adalah :

1. Bagi Masyarakat : Membantu para mahasiswa dalam menunjang kualitas belajar mereka dengan cara meningkatkan efisiensi penggunaan *smartphone* untuk mengakses informasi terkait perkuliahan mereka.
2. Bagi Peneliti : Diharapkan dapat menambah wawasan mengenai pengembangan aplikasi berbagi informasi antar mahasiswa berbasis Android.
3. Bagi Pendidikan : Mampu menjadi bahan referensi mengenai aplikasi *smartphone* yang dapat menunjang perkuliahan.

1.5 Batasan Masalah Penelitian

Batasan masalah dalam penelitian ini adalah:

1. Fitur aplikasi difokuskan pada informasi yang terkait seputar perkuliahan seperti jadwal kuliah, tugas, maupun kontak dosen.
2. Pembuatan aplikasi ini ditujukan untuk *smartphone* berbasis Android.
3. Perangkat lunak yang akan digunakan dalam manajemen akun dan *database* adalah Firebase.
4. Bahasa pemrograman yang akan digunakan adalah Kotlin.

1.6 Sistematika Penulisan

Untuk memberikan gambaran singkat mengenai isi tulisan secara keseluruhan, maka akan diuraikan beberapa tahapan dari penulisan secara sistematis, yaitu:

BAB I PENDAHULUAN

Bab ini menguraikan secara umum mengenai hal yang menyangkut latar belakang, perumusan masalah dan batasan masalah, tujuan, manfaat, dan sistematika penulisan dalam penelitian Perancangan Aplikasi Berbagi Informasi Seputar Perkuliahan antar Mahasiswa pada Smartphone Berbasis Android.

BAB II TINJAUAN PUSTAKA

Bab ini berisi teori-teori tentang hal-hal yang berhubungan dengan penelitian dan pengembangan seperti Android Studio, Kotlin, serta Firebase.

BAB III METODE PENELITIAN

Bab ini berisi tentang tahapan, waktu dan lokasi, instrumen penelitian, perancangan, serta analisis kerja sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil penelitian serta pembahasan yang disertai tabel hasil penelitian.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan yang didapatkan berdasarkan hasil penelitian yang telah dilakukan serta saran-saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Android

Android adalah sistem operasi untuk telepon seluler berbasis Linux sebagai kernelnya. Saat ini Android menjadi pesaing utama dari produk smartphone lainnya seperti Apple dan Blackberry karena Android memiliki beberapa kelebihan daripada smartphone lain, yaitu:

- Android bersifat *open source* yang artinya pengembang bebas untuk mengembangkan aplikasi pada *platform* ini.
- Lengkap. Android menyediakan peralatan untuk membangaun perangkat lunak yang sangat lengkap dibanding dengan *platform* lain.
- Bebas. Android adalah *platform mobile* yang tidak memiliki batasan dalam mengembangkan aplikasinya. Tidak ada lisensi dalam mengembangkan aplikasi Android (Pandu, 2014).

2.2 Android Studio

Menurut Android Developer (2020), Android Studio adalah sebuah *Integrated Development Environment* (IDE) resmi untuk pengembangan aplikasi Android, berdasar IntelliJ IDEA. Selain penyunting kode dan alat pengembang yang kuat dari IntelliJ, Android Studio menawarkan lebih banyak fitur yang meningkatkan produktivitas saat membuat aplikasi Android, seperti:

- Sistem *build* berbasis *Gradle* yang fleksibel
- Sebuah lingkungan terpadu pengembangan untuk semua perangkat Android
- Alat dan *framework* pengujian yang ekstensif
- *Lint* untuk mengetahui kinerja, kegunaan, kompabilitas versi dan masalah lainnya
- Dukungan terhadap *C++* dan *NDK (Native Development Kit)*

2.3 Kotlin

Kotlin pertama kali diperkenalkan oleh sebuah perusahaan software bernama JetBrains yang juga dikenal dengan IntelliJ Software pada tahun 2011. Saat itu Kotlin belum begitu populer hingga Kotlin diumumkan sebagai bahasa baru untuk JVM. Pada tanggal 17 Mei 2017, Kotlin akhirnya diumumkan oleh Google sebagai bahasa resmi yang mendukung pengembangan aplikasi Android selain Java di Google I/O 2017 (Samojło, 2019). Sejak itu, Kotlin menjadi semakin populer di komunitas pengembang dan untuk pertama kalinya memasuki 50 bahasa teratas pada bulan Juni. Karena popularitas Kotlin, banyak aplikasi dari perusahaan rintisan terpanas dan perusahaan Fortune 500 seperti Gojek, Pinterest, Twitter, Airbnb, Netflix, dll. Menggunakan Kotlin untuk membangun aplikasi. Kotlin di bawah lisensi Apache 2.0, juga gratis & *open source*. Kotlin mirip dengan bahasa Swift untuk pengembangan Apple.

Nama Kotlin didapat dari pulau Kotlin yang terletak di Rusia. Kotlin adalah bahasa pemrograman dengan tipe statis yang dapat dijalankan di Java Virtual Machine (JVM). Masyarakat memprediksikan bahwa masa depan Kotlin

tidak hanya terbatas pada aplikasi Android saja tetapi akan ditingkatkan lebih jauh untuk platform iOS juga (Patel, 2019). Kotlin juga dapat digunakan untuk pengembangan backend. Sintaks Kotlin jauh lebih ringkas dan Kotlin dapat dioperasikan dengan bahasa pemrograman Java yang berarti sintaks Kotlin juga berfungsi dengan Java.

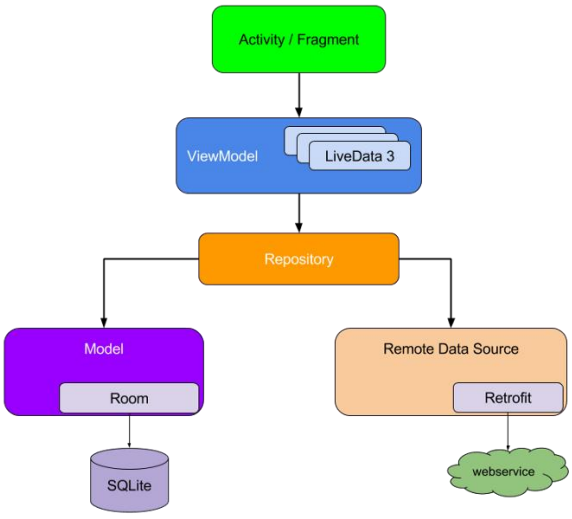
Salah satu fitur terbaik di Kotlin yang telah dibahas oleh banyak developer adalah jenis *Nullable* dan *NonNull* yang diimplementasikan pada Kotlin untuk menghindari kesalahan *NullPointerExeceptions* yang biasanya terjadi selama eksekusi program (Yugandhar, 2019). Kotlin masih dalam pengembangan berat karena Kotlin masih cukup baru, oleh karena itu developer berharap Kotlin akan terus berkembang seiring waktu. Kotlin berfungsi dengan Android Studio versi 3.0 ke atas.

2.4 *Model-View-ViewModel* (MVVM)

Menurut Putra (2019), *Model-View-ViewModel* atau disingkat MVVM adalah salah satu *Architectural Patterns* yang membagi tanggung jawab kepada tiga komponen, yaitu *Model*, *View* dan *ViewModel*. *View* bertanggung jawab untuk semua hal yang berhubungan dengan *UI* seperti menampilkan *loading*, *dialog*, *toast* dll. *Model* merupakan komponen yang bertanggung untuk menyediakan data yang dibutuhkan. Terakhir *ViewModel* merupakan komponen inti dari *Architectural Patterns* ini, tugasnya menyimpan dan mengambil data dari *Model* untuk nantinya ditampilkan oleh *View*.

MVVM adalah satu dari sekian banyaknya *Architectural Patterns* yang ada. Banyak lagi yang lainnya seperti MVC (*Model-View-Controller*), MVI (*Model-View-Intent*), MVP (*Model-View-Presenter*) serta masih banyak lagi. MVP dan MVVM merupakan pola yang paling sering digunakan dalam pengembangan aplikasi Android.

Pada Google I/O 2017 yang lalu, Google memperkenalkan *library* baru *Architecture Component*. *Library* ini menyediakan beberapa komponen yang mendukung pengembangan aplikasi Android dengan pattern MVVM. Sejak itu juga, Google merekomendasikan *pattern* MVVM untuk pengembangan aplikasi Android.



Gambar 2.1 **Cara Kerja MVVM**

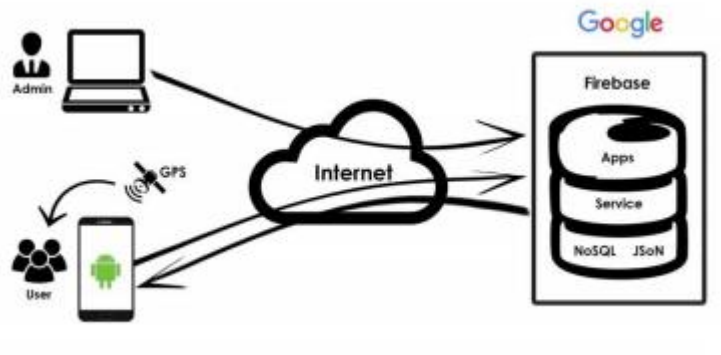
Penjelasan dari cara kerja MVVM adalah sebagai berikut:

1. *Activity* bertanggung jawab sebagai *View*.
2. *View* akan melakukan observasi terhadap data yang disimpan di *ViewModel*. Apabila terjadi perubahan *data* pada *ViewModel*, maka *View* bertanggung jawab untuk melakukan pembaharuan pada antarmuka sesuai dengan data.
3. *ViewModel* menyimpan *data* berupa *LiveData* agar *View* dapat melakukan observasi.
4. *ViewModel* berkomunikasi dengan *Repository* (*Model*) untuk mendapatkan data atau perubahan data dan melakukan update terhadap data yang dimiliki.
5. *Repository* bertanggung jawab untuk mengatur sumber data yang dibutuhkan. *Data* bisa didapatkan baik dari *server* maupun dari *database* lokal menggunakan SQLite.

2.5 Firebase

Firebase memiliki beberapa produk, yaitu autentikasi, *database*, fungsi, serta *backend* sebagai layanan (*Backend as a Service*). Layanan ini menyediakan pengembang aplikasi *API* yang memungkinkan aplikasi data yang akan disinkronisasi di klien dan disimpan pada *cloud* Firebase ini. Firebase menyediakan *library* untuk berbagai *client platform* yang memungkinkan integrasi dengan Android, iOS, JavaScript, Java, Objective-C dan NodeJS dan

dapat juga disebut sebagai layanan DbaaS (*Database as a Service*) dengan konsep *realtime*. Firebase digunakan untuk mempermudah pengembang dan pemrograman dalam mengembangkan aplikasi-aplikasinya.



Gambar 2.2 **Arsitektur Sistem Firebase**

1. Firebase Authentication

Firebase Authentication adalah sebuah fitur oleh Firebase yang dirancang untuk mengelola akun pengguna dimulai dari mendaftar, masuk dan memverifikasi kredensial pengguna selama di mana Firebase Authentication akan menyimpan semua email dan kata sandi pengguna. Firebase Authentication dapat menyimpan *token* pengguna apabila pengguna telah login sebelumnya untuk masuk kembali secara otomatis. Pada saat pengguna melakukan *logout*, token pengguna akan dihapus untuk mencegah masuk kembali secara otomatis otomatis. Selain itu, Firebase Authentication dapat memperbarui kredensial pengguna seperti mengganti *e-mail* & kata sandi.

2. Firebase Realtime Database

Semua data Firebase Realtime Database disimpan sebagai objek JSON. Bisa dianggap *database* sebagai JSON *tree* yang di-*host* pada *cloud*. Tidak seperti *database* berbasis SQL, Firebase Realtime Database berbasis NoSQL sehingga tidak ada tabel atau rekaman. Ketika ditambahkan ke JSON *tree*, data akan menjadi simpul dalam struktur JSON yang ada. Meskipun basis data menggunakan JSON *tree*, data yang tersimpan dalam basis data bisa diwakili sebagai tipe bawaan tertentu yang sesuai dengan tipe JSON yang tersedia untuk membantu dalam menulis lebih banyak kode yang bisa dipertahankan.

Ada empat metode untuk menulis data ke Firebase Realtime Database:

Metode	Penggunaan umum
`setValue()`	Menulis atau mengganti data ke jalur yang didefinisikan, seperti `users/<user-id>/<username>`.
`push()`	Tambahkan ke daftar data. Setiap kali Anda memanggil `push()`, Firebase akan menghasilkan ID unik, seperti `user-posts/<user-id>/<unique-post-id>`.
`updateChildren()`	Memperbarui beberapa kunci untuk jalur yang didefinisikan tanpa mengganti semua data.
`runTransaction()`	Memperbarui data kompleks yang bisa rusak karena pembaruan bersamaan.

Gambar 2.3 **Fungsi penulisan pada Firebase Realtime Database**

Untuk operasi tulis dasar, *setValue*() bisa digunakan untuk menyimpan data ke referensi yang ditetapkan, menggantikan data yang ada di jalur tersebut.

Ada dua tipe pengambilan data pada *Firebase Realtime Database yaitu ValueEventListener* dan *ChildEventListener*:

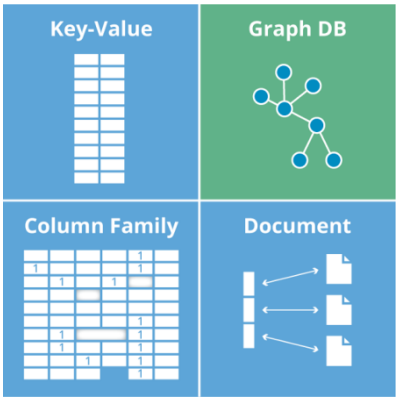
Listener	Callback kejadian	Penggunaan biasa
`ValueEventListener`	`onDataChange()`	Membaca dan mendengarkan perubahan untuk seluruh konten jalur.
`ChildEventListener`	`onChildAdded()`	Mengambil daftar item atau mendengarkan penambahan daftar item. Disarankan untuk digunakan dengan `onChildChanged()` dan `onChildRemoved()` untuk memantau perubahan daftar.
	`onChildChanged()`	Mendengarkan perubahan pada item dalam daftar. Gunakan dengan `onChildAdded()` dan`onChildRemoved()` untuk memantau perubahan daftar.
	`onChildRemoved()`	Mendengarkan item yang dibuang dari daftar. Gunakan dengan `onChildAdded()` dan`onChildChanged()` untuk memantau perubahan daftar.
	`onChildMoved()`	Gunakan dengan data diurutkan untuk mendengarkan perubahan dalam prioritas item.

Gambar 2.4 **Fungsi pengambilan pada Firebase Realtime Database**

Untuk menambahkan listener kejadian, gunakan metode *addValueEventListener*() atau *addListenerForSingleValueEvent*(). Untuk menambahkan *listener* kejadian anak, gunakan metode *addChildEventListener*(). Metode *onDataChange*() untuk membaca cuplikan statis konten pada jalur tertentu, seperti yang telah ada pada saat kejadian. Metode ini terpicu satu kali 10 ketika listener terpasang dan terpicu lagi setiap kali terjadi perubahan data, termasuk anaknya. *Callback* meneruskan cuplikan yang berisi semua data di lokasi tersebut, termasuk data anak. Jika tidak ada data, cuplikan yang dikembalikan adalah *null*. Metode *onDataChange*() dipanggil setiap kali terjadi perubahan data pada referensi database yang ditetapkan, termasuk perubahan ke anaknya. (Firebase, 2020).

2.6 NoSQL

Menurut Thoiba (2019), NoSQL adalah sebuah database yang bersifat non-SQL atau non-relasional, dengan kata lain tidak mengenal relasi antar tabel. Perbedaan mendasar dengan SQL adalah tidak mengenal relasi antar tabel dan tidak memerlukan struktur yang sama untuk setiap recordnya. Penggunaan NoSQL pada saat sekarang ini sudah mengalami peningkatan yang signifikan.



Gambar 2.5 **Tipe-tipe NoSQL**

Tipe-tipe NoSQL beserta penjelasannya adalah sebagai berikut:

3. Key-Value

Setiap *record* data yang disimpan hanya berupa *key* dan *value*. Tipe ini biasanya digunakan untuk *caching*. Kelebihannya adalah *read* dan *write* sangat cepat. Contoh yang paling populer adalah Redis dan Memcache.

4. Document

Setiap *record* data yang disimpan berupa dokumen-dokumen. Setiap dokumen satu dengan dokumen lainnya tidak

harus mempunyai struktur yang sama walaupun itu dalam satu kumpulan data (dalam terminologi SQL disebut *table*). Kelebihannya adalah tidak perlu menentukan skema, jadi setiap perubahan skema yang diperlukan aplikasi, tidak memerlukan perubahan semua data skema, tidak perlu *ALTER TABLE* seperti halnya SQL. Selain itu juga bagus untuk *horizontal scaling*. Setiap dokumen yang disimpan secara *default* mempunyai *key* unik. *Encoding* yang digunakan adalah XML, JSON, YAML dan BSON.

5. Graph

Database ini didesain untuk relasi yang direpresentasikan seperti *graph.* Terdiri dari *node* dan *edge*. Penggunaan yang paling populer adalah untuk social media, publik transportasi, map, topologi jaringan, dan lain-lain.

6. Column

Column database adalah penyimpanan data dengan model seperti *column*. *Database* ini menggunakan konsep dengan istilah *keyspace*. Sebuah *keyspace* seperti *schema* dalam *SQL*. Keyspace berisi *column families* (seperti tabel dalam SQL) terdiri dari *row*s, dan *column*. Kelebihannya adalah *compression data*, *aggregation queries* (SUM, COUNT, AVG, dll), *scalability*, *read* dan *write* sangat cepat. Contoh yang paling populer adalah Cassandra.

2.7 Room

Library persistensi Room yang merupakan bagian dari Android Jetpack menyediakan lapisan abstraksi di atas SQLite untuk memungkinkan akses database yang lancar sambil memanfaatkan kekuatan penuh SQLite. Secara khusus, Room memberikan keuntungan berikut:

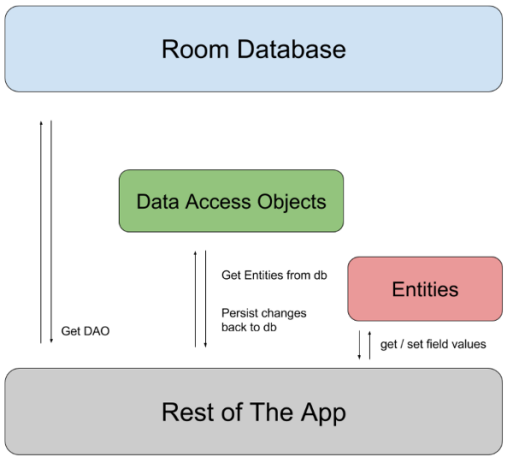
- Verifikasi waktu kompilasi untuk kueri SQL.
- Anotasi yang nyaman untuk digunakan untuk meminimalkan kode *boilerplate* berulang dan rawan kesalahan.
- Jalur migrasi database yang disederhanakan.

Kemudian, terdapat tiga komponen utama di Room:

- Kelas *database* yang menyimpan *database* dan berfungsi sebagai titik akses utama untuk koneksi yang mendasari ke data tersimpan aplikasi.
- Entitas data yang mewakili tabel di *database* aplikasi.
- *Data Access Object* (DAO) yang menyediakan metode yang bisa digunakan aplikasi untuk membuat kueri, memperbarui, menyisipkan, dan menghapus data dalam *database*.

Kelas *database* menyediakan aplikasi dengan contoh DAO yang terkait dengan *database* tersebut. Pada gilirannya, aplikasi dapat menggunakan DAO untuk mengambil data dari database sebagai contoh objek entitas data terkait. Aplikasi juga dapat menggunakan entitas data yang ditentukan untuk memperbarui baris dari tabel yang sesuai, atau untuk membuat baris baru untuk

penyisipan. Gambar 2.6 mengilustrasikan hubungan antara berbagai komponen Room.



Gambar 2.6 **Arsitektur *library* Room**

2.8 SQLite

SQLite adalah *in-process library* yang mengimplementasikan mesin database SQL transaksional, tanpa *server*, tanpa konfigurasi, dan mandiri. Kode untuk SQLite ada di domain publik dan dengan demikian gratis untuk digunakan untuk tujuan apa pun, komersial atau pribadi. SQLite adalah *database* yang banyak digunakan di dunia.

SQLite adalah mesin database SQL tertanam. Tidak seperti kebanyakan database SQL lainnya, SQLite tidak memiliki proses *server* terpisah. SQLite membaca dan menulis langsung ke file disk biasa. Database SQL lengkap dengan beberapa tabel, indeks, pemicu, dan tampilan, terdapat dalam satu file disk.

SQLite adalah *library* yang lengkap. Walau semua fitur diaktifkan, ukuran pustaka hanya kurang dari 600KiB, tergantung pada *platform target* dan

pengaturan pengoptimalan *compiler* (ukuran kode untuk versi 64-bit lebih besar serta beberapa pengoptimalan *compiler* seperti *inlining* fungsi yang agresif dan pengulangan *loop* dapat menyebabkan kode objek menjadi jauh lebih besar). Terdapat pertukaran antara penggunaan memori dan kecepatan. SQLite umumnya berjalan lebih cepat semakin banyak memori yang Anda berikan. Namun demikian, kinerja biasanya cukup baik bahkan di lingkungan dengan memori rendah. Bergantung pada cara penggunaannya, SQLite dapat lebih cepat daripada I / O sistem file langsung.

2.9 Agile Development

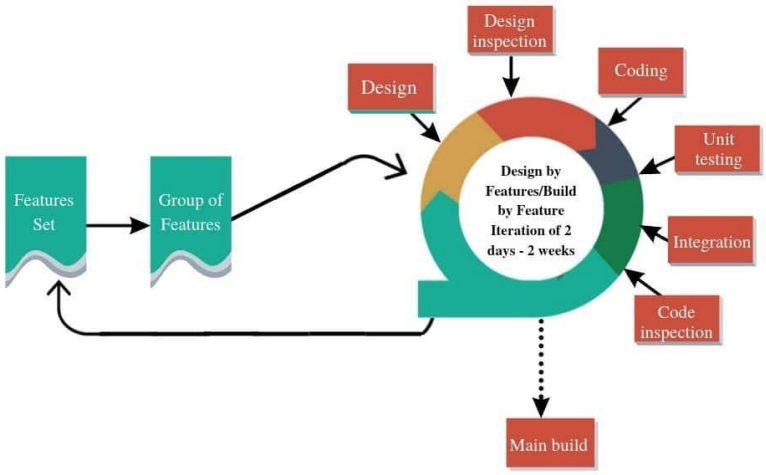
Menurut Pressman (2010), *agile development* adalah sebuah filosofi dan metode pengembangan program yang mengutamakan komunikasi antara *developer* dengan *customers*.

Feature Driven Development

Menurut Palmer & Felsing (2002), FDD adalah proses yang didesain dan dilaksanakan untuk menyajikan (*deliver*) hasil kerja secara berulang-ulang dalam waktu tertentu dan dapat diukur. FDD adalah pendekatan yang mengacu pembuatan sistem menggunakan metode yang mudah dimengerti dan mudah diimplentasikan; teknik *problem solving*; dan pelaporan yang mudah dimengerti dan dikontrol oleh *stakeholders*.

Pemrogram diberikan informasi yang cukup dan diberikan alat bantu yang baik untuk menyelesaikan aplikasi yang diberikan. *Team leader* dan manajer proyek diberikan informasi yang baik berdasar waktu

mengenai tim dan proyek yang berjalan untuk menghindari resiko yang mungkin terjadi. Pelaporan menjadi lebih mudah, tidak membebani, periodik dan akurat.



Gambar 2.7 **Diagram urutan proses pada *Feature Driven Development***

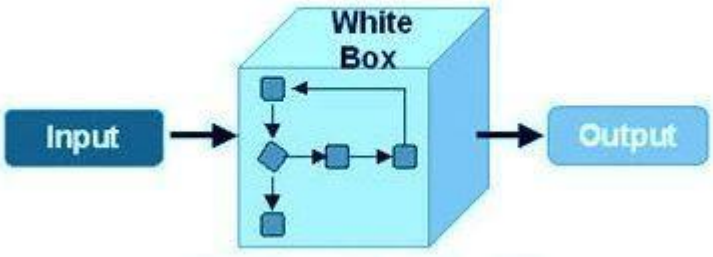
Pendekatan FDD mendefinisikan lima kegiatan kerangka kerja "berkolaborasi" (dalam FDD ini disebut "proses"), yaitu mengembangkan sebuah model keseluruhan (*develop an overall model*), membangun daftar fitur (*build a features list*), merencanakan dengan fitur (*plan by feature*), mendesain dengan fitur (*design by feature*), dan membangun dengan fitur (*build by feature*).

1. Mengembangkan sebuah model keseluruhan (*develop an overall model*): Proyek FDD dimulai dengan penelusuran tingkat tinggi dari ruang lingkup sistem dan konteksnya.
2. Membangun sebuah daftar fitur (*build a features list*): Seperangkat fitur digabungkan menjadi seperangkat area subjek

3. Merencanakan dengan fitur (*plan by feature*): Setelah daftar fitur selesai, langkah selanjutnya adalah membuat rencana pengembangan dan menetapkan kepemilikan fitur (atau kumpulan fitur) sebagai kelas untuk *programmer*.
4. Mendesain dengan fitur (*design by feature*): Menghasilkan paket desain (*design package*) untuk setiap fitur.
5. Membangun dengan fitur (*build by feature*): Setelah inspeksi desain yang berhasil untuk setiap kegiatan untuk menghasilkan fitur yang telah direncanakan. Setelah pengujian unit dan inspeksi kode yang berhasil, fitur yang lengkap (*complete feature*) dipromosikan *ke main build* lalu kembali lagi mengulangi iterasi ini pada fitur selanjutnya.

2.10 *Software Testing*

1. *White Box Testing*



Gambar 2.8 **Ilustrasi *White Box Testing***

Structural testing adalah pandangan di mana unit perangkat lunak dilihat sebagai "*white box*". Pilihan kasus uji didasarkan tentang implementasi entitas perangkat lunak. Rancang kasus uji yang menguji

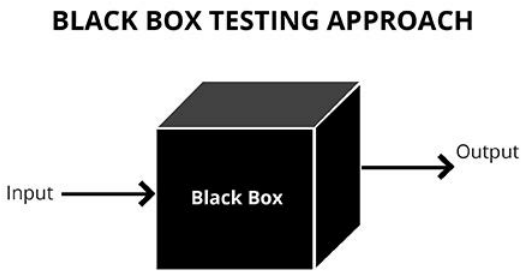
fungsi internal file perangkat lunak dari perspektif pengembang, pengujian kotak putih terutama berfokus pada logika internal dan struktur kode.

White box testing adalah pengujian yang dilakukan apabila programmer memiliki pengetahuan teknik yang lengkap tentang struktur program. Dengan teknik ini dimungkinkan untuk menguji setiap cabang dan keputusan dalam program. Ketika struktur internal diketahui, menarik untuk melihat kriteria cakupan yang berbeda. Salah satu yang paling penting adalah cakupan keputusan. Tes ini tepat hanya jika penguji mengenali programnya yang harus di lakukan. Penguji kemudian dapat melihat apakah program tersebut terpisah dari tujuan yang dimaksudkan.

White box testing terutama digunakan untuk mendeteksi kesalahan logis dalam kode program. Ini digunakan untuk *debugging* kode, menemukan kesalahan ketik, dan mengungkap pemrograman yang salah asumsi.

Pemilihan kasus uji didasarkan pada implementasi entitas perangkat lunak. Tujuan dari pemilihan kasus uji tersebut adalah untuk menyebabkan eksekusi tempat tertentu dalam entitas perangkat lunak, seperti pernyataan, cabang atau jalur program tertentu. Hasil yang diharapkan dievaluasi pada seperangkat kriteria cakupan. Contoh kriteria cakupan mencakup cakupan jalur, cakupan cabang, dan cakupan aliran data. Pengujian struktural menyoroti struktur internal entitas perangkat lunak (Nidhra & Dondeti, 2012).

2. *Black Box Testing*



Gambar 2.9 **Ilustrasi *Black Box Testing***

Functional testing adalah pandangan di mana program perangkat lunak atau sistem yang diuji dipandang sebagai "*black box*". *Black box testing* adalah pengujian berdasarkan spesifikasi persyaratan dan tidak perlu menguji kode dalam pengujian kotak hitam. Ini murni dilakukan berdasarkan sudut pandang pelanggan, hanya penguji yang tahu set input dan output yang dapat diprediksi. Pengujian black box dilakukan pada produk jadi (Nidhra & Dondeti, 2012).

Menurut Shalahuddin dan Sukamto (2011), *black box testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian black box testing harus dibuat dengan kasus benar dan kasus salah.

Menurut Pressman (2010), *black box testing* juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian black box memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performasi, kesalahan inisialisasi dan akhir program. Terdapat 3 macam *black box testing* yang umum digunakan, yaitu:

- *Functional testing* (Pengujian fungsional) - Jenis pengujian kotak hitam ini terkait dengan persyaratan fungsional suatu sistem; itu dilakukan oleh perangkat lunak penguji.
- *Non-functional testing* (Pengujian non-fungsional) - Jenis pengujian kotak hitam ini tidak terkait dengan pengujian fungsionalitas tertentu, tetapi persyaratan non-fungsional seperti kinerja, skalabilitas, kegunaan.
- *Regression testing* (Pengujian regresi) - Pengujian Regresi dilakukan setelah perbaikan kode, peningkatan atau pemeliharaan sistem lainnya untuk memeriksa kode baru tidak memengaruhi kode yang ada.

Untuk teknik pengujian, terdapat 3 teknik *black box testing* yang umum digunakan, yaitu:

- *Equivalence Class Testing* (Pengujian Kelas Ekuivalen) : Digunakan untuk meminimalkan jumlah kasus uji yang mungkin hingga tingkat optimal sambil mempertahankan cakupan uji yang wajar.
- *Boundary Value Testing* (Pengujian Nilai Batas) : Pengujian nilai batas difokuskan pada nilai pada batas. Teknik ini menentukan apakah rentang nilai tertentu dapat diterima oleh sistem atau tidak. Ini sangat berguna dalam mengurangi jumlah kasus uji. Ini paling cocok untuk sistem di mana input berada dalam rentang tertentu.
- *Decision Table Testing* (Pengujian Tabel Keputusan) : Tabel keputusan menempatkan penyebab dan efeknya dalam sebuah matriks. Ada kombinasi unik di setiap kolom.

3. *Automation Testing*

Menurut Choudary (2020), *Automation Testing* atau pengujian otomatisasi adalah proses menggunakan bantuan alat, skrip, dan perangkat lunak untuk melakukan kasus uji dengan mengulangi tindakan yang telah ditentukan sebelumnya. Pengujian otomatisasi berfokus pada penggantian aktivitas manusia manual dengan sistem atau perangkat yang meningkatkan efisiensi.

Pengujian sangat penting untuk keberhasilan produk perangkat lunak apa pun. Jika perangkat lunak yang telah dikembangkan tidak berfungsi dengan baik, kemungkinan sebagian besar orang bahkan tidak akan membeli atau menggunakan produk perangkat lunak yang telah dikembangkan, bahkan jika mereka menggunakannya pun tidak akan

bertahan lama. Tetapi pengujian untuk menemukan cacat atau bug secara manual memakan waktu, mahal, sering kali berulang, dan tunduk pada kesalahan manusia. Di sinilah otomatisasi masuk untuk membantu. Otomasi sangat penting bagi tim pengembangan perangkat lunak untuk mengimbangi meningkatnya permintaan perangkat lunak berkualitas lebih tinggi secepat kilat. Saat memulai pengujian, salah satu keputusan utama yang harus dibuat adalah apakah pengujian akan dilakukan secara manual atau menggunakan pengujian otomatis. Berikut adalah perbedaan antara pengujian manual dan pengujian otomatis.

Tabel 2.1 **Perbedaan pengujian manual dan pengujian otomatisasi**

Aspek	Pengujian Manual	Penguijan Otomatisasi
Akurasi & Keandalan	Rendah, karena tes manual lebih rentan terhadap kesalahan manusia	Tinggi, karena alat dan skrip digunakan
Waktu yang dibutuhkan	Tinggi	Relatif rendah
Biaya Investasi	Kecil, tetapi timbal balik investasinya rendah	Besar, tetapi timbal balik investasinya tinggi
Penggunaan	Cocok untuk penjelajahan kegunaan dan pengujian *ad hoc*	Cocok untuk pengujian regresi, pengujian kinerja, pengujian beban
Keterlibatan	Memperbolehkan manusia untuk mengobservasi	Tidak ada pengamatan

manusia	dalam menemukan kecacatan	manusia yang terlibat
Pengalaman pelanggan	Membantu dalam meningkatkan pengalaman pelanggan	Tidak ada jaminan pengalaman pelanggan yang positif

4. *User Acceptance Testing* (UAT)

Menurut Perry (2006), *User Acceptance Testing* atau disingkat UAT merupakan pengujian yang dilakukan oleh *end-user* yaitu pengguna yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya. Setelah dilakukan *system testing*, *acceptance testing* menyatakan bahwa sistem *software* memenuhi persyaratan. *Acceptance testing* merupakan pengujian yang dilakukan oleh pengguna yang menggunakan teknik pengujian black box untuk menguji sistem terhadap spesifikasinya. Pengguna akhir bertanggung jawab untuk memastikan semua fungsionalitas yang relevan telah diuji.