BAB II
LANDASAN TEORI

2.1 Pengertian Peta

Peta adalah gambaran dari permukaan bumi yang divisualisasikan pada bidang yang datar lalu diperkecil dengan skala yang ditentukan serta dilengkapi dengan simbol-simbol sebagai penjelas dalam gambar. [7]

2.1.1 Peta Datar

Peta datar adalah jenis peta yang digambar pada bidang datar, misalnya kertas, kain, papan, dan sebagainya. Peta ini sering ditemui dan digunakan, peta ini juga terdapat berbagai macam simbol yang berbeda bentuk dan warnanya.

Gambar 2.1 Peta Datar

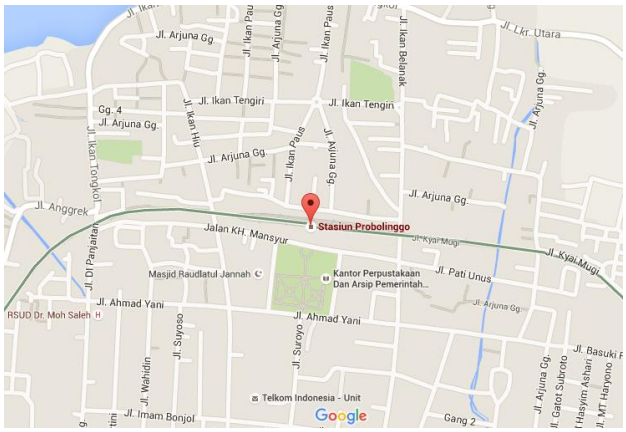
2.1.2 Peta Timbul

Peta timbur adalah jenis peta berbentuk 3 dimensi yang memiliki tujuan menyerupai dengan bentuk permukaan bumi yang sebenarnya, misalnya gunung dipeta ini terlihat agak menonjol dibandingkan permukaan yang lain, begitu pula dengan kedalaman laut yang lebih rendah dibandingkan permukaan tanah.

Gambar 2.2 Peta Timbul

2.1.3 Peta Digital

Peta digital merupakan jenis peta yang gambarkan melalui bantuan dari komputer secara digital. Peta ini hanya dapat dilihat melalui perangkat teknologi seperti laptop, komputer dan smartphone. Contoh peta digital adalah melalui aplikasi Google Maps.



Gambar 2.3 Peta Digital

Pada penelitian ini penulis bermaksud untuk menerapkan peta digital sebagai peta pada UNIKOM *Navigator* di gedung baru UNIKOM. Peta digital mempermudah pengguna untuk mengaksesnya karena pengguna hanya memerlukan smartphone dan membuka aplikasinya.

2.2 Pengertian Navigasi

Navigasi merupakan suatu kegiatan yang meliputi segala sesuatu yang berkaitan dengan sarana bantu untuk kepentingan navigasi saat berlayar. [8] Menurut Kamus besar bahasa Indonesia, navigasi diartikan :

1. Ilmu tentang cara menjalankan kapal laut atau kapal terbang.
2. Tindakan menetapkan haluan kapal atau arah terbang.
3. Pelayaran atau penerbangan.

Pada penelitian ini navigasi digunakan untuk menentukan arah dari lokasi pengguna ke lokasi tujuan.

2.3 Pengertian Beacon

Beacon adalah *Bluetooth Low Energy* (BLE) yang memancarkan sinyal radio *bluetooth* secara berkala yang menggunakan daya baterai koin, atau dengan daya lain seperti baterai AA dan USB. Dengan teknologi BLE yang memungkinkan Beacon beroperasi tanpa gangguan selama berbulan-bulan. Beacon berukuran kecil dan mudah dipasang, contoh beacon dapat dilihat pada Gambar 2.4.

Gambar 2.4 BLE Beacon

Beacon memancarkan sinyal sesuai dengan interval yang ditentukan, interval berkisar dari beberapa detik hingga 100ms atau kurang. Semakin kecil interval yang ditentukan, semakin sering beacon memancarkan sinyal.[3]

Beacon dapat dijadikan GPS (*Global Positioning System*) di dalam ruangan, karena GPS kurang akurat dalam menentukan posisi di dalam ruangan. Selain dapat menentukan posisi di dalam ruangan, dan biasanya di dalam ruangan atau gedung terdapat banyak lantai, Beacon dapat membedakaan *altitude* dari pengguna ketika smartphone dari pengguna terhubung dengan Beacon sedangkan GPS tidak dapat membedakan *altitude*. Beacon juga dapat menentukan suatu daerah tertentu, sebagai contoh di dalam tempat belanja buah atau sejenisnya yaitu

dengan menentukan daerah sayur-sayuran, buah-buahan, makanan ringan dan sebagainya, sehingga pengguna lebih mudah untuk mencari bahan makanan.

2.4 Android

Android merupakan suatu sistem operasi dengan basis linux yang khusus dirancang untuk perangkat layar sentuh seperti *smartphone*, *tablet* dan sejenisnya. Awalnya Android, inc mengembangkan sistem operasi Android dengan adanya dukungan finansial dari Google yang kemudian dibeli oleh Google itu sendiri pada tahun 2005. Pada tahun 2007 *Open Handset Allieance*, perusahaan-perusahaan software, hardware, dan telekomunikasi didirak bersamaan dengan rilisnya sistem operasi Android untuk memajukan standar terbuka perangkat seluler. Ponsel android pertama kali mulai dijual pada bulan oktober 2008.[9]

2.4.1 Sejarah Android

Android, Inc. Didirikan di Palo Alto, California, Pada bulan oktober 2003 oleh Andy Rubin (Pendiri Danger), Rich Milner (pendiri Wildfire Cimmunication, Inc.), Nick Sears (mantan Vp T-Mobile) dan Chris White (kepala desain dan pengembangan antarmuka WebTV) untuk mengembangkan perangkat smartphone yang lebih sadar akan referensi dan lokasi penggunanya. Tujuan dikembangkan sistem operasi Android adalah untuk pengembangan sebuah sistem operasi yang canggih untuk kamera digital, namun disadari bahwa pasar untuk perangkat tersebut tidak cukup besar dan pengembangan Android lalu dialihkan ke pasar smartphone untuk menyaingi Symbian dan Windows Mobile.

Pada tanggal 17 agustus 2005,Google mengakuisisi Android Inc dan menjadikannya sebagai anak perusahaan yang dimiliki oleh Google. Setelah diakuisisi oleh Google Miner, White dan Rubin selaku Pendiri Android Inc tetap bekerja di perusahaan. Setelah itu banyak anggapan yang menyatakan bahwa Google telah berencana untuk memasuki pasar telepon seluler dengan tindakannya ini.

Hingga bulan desember 2006 spekulasi tentang niat Google untuk memasuki pasar komunikasi seluler terus berkembang.

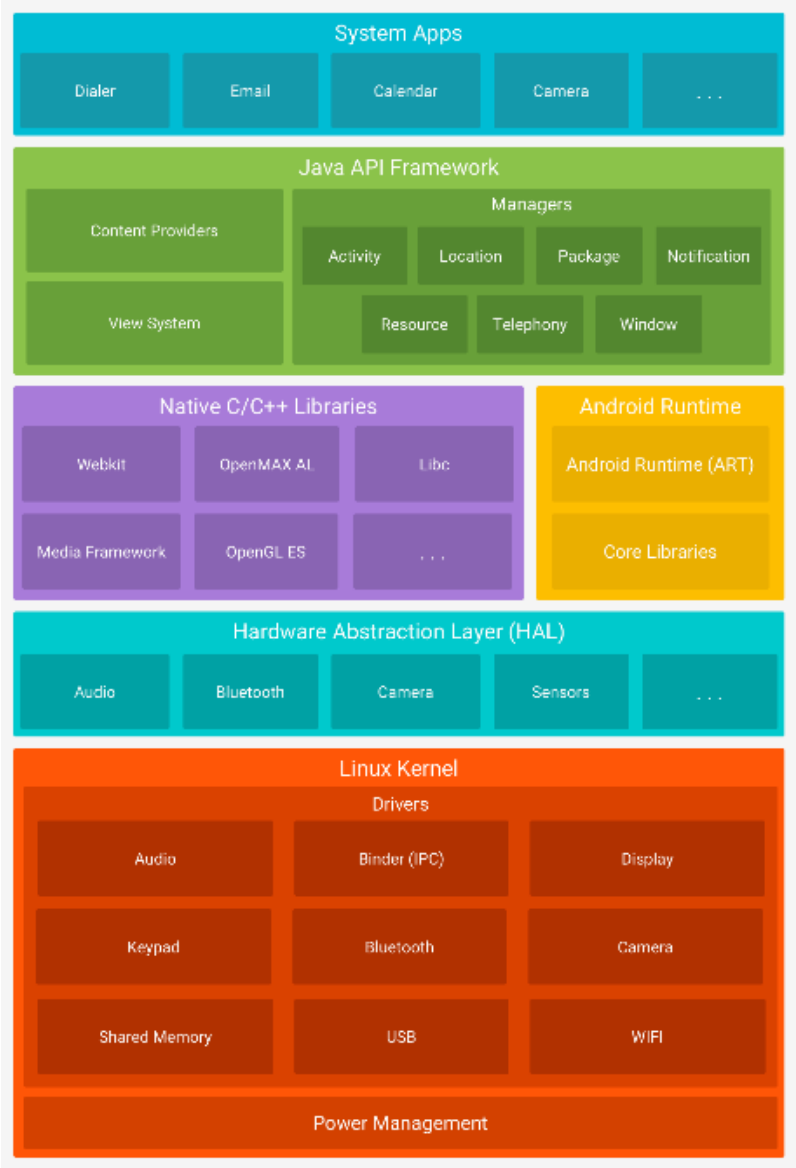
Open Handset Alliance (OHA) didirikan Pada tanggal 5 November 2007. OHA adalah konsorsium dari perusahaan-perusahaan teknologi seperti Google, produsen perangkat seluler seperti HTC, Sony dan Samsung, operator nirkabel seperti Sprint Nextel dan T-Mobile, serta produsen chipset seperti Qualcomm dan Texas Instruments. OHA sendiri bertujuan untuk mengembangkan standar terbuka bagi perangkat seluler Pada 22 Oktober 2008 yang juga telepon seluler komersial pertama yang menggunakan sistem operasi Android adalah HTC Dream.

Google merilis seri Nexus yaitu perangkat telepon pintar dan tablet dengan sistem operasi Android pada tahun 2010 yang diproduksi oleh mitra produsen telepon seluler seperti HTC, LG, dan Samsung. HTC bekerja sama dengan Google dalam merilis smartphone Nexus pertama, yaitu Nexus One.

Android telah melakukan sejumlah pembaruan untuk meningkatkan kinerja sistem operasi Android, menambahkan fitur baru, dan memperbaiki bug yang terdapat pada versi sebelumnya Sejak tahun 2008. Setiap versi utama yang dirilis dinamakan secara alfabetis berdasarkan nama-nama makanan pencuci mulut(desert) atau makanan ringan bergula; misalnya, versi 1.5 bernama Cupcake, yang kemudian diikuti oleh versi 1.6 Donut. Versi terbaru adalah 5.0 Lollipop, yang dirilis pada 15 Oktober 2014.[9]

2.4.2 Arsitektur Android

Android adalah tumpukan perangkat lunak berbasis Linux sumber terbuka yang dibuat untuk berbagai perangkat dan faktor bentuk. Diagram berikut menunjukkan komponen besar dari platform Android.



Gambar 2.5 Arsitektur Android

1. **Linux Kernel**

Fondasi platform Android adalah kernel Linux. Sebagai contoh, Android Runtime (ART) bergantung pada kernel Linux untuk fungsionalitas dasar seperti *threading* dan manajemen memori tingkat rendah. Menggunakan kernel Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

1. Hardware Abstraction Layer (Hal)

Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.

2. Android Runtime

Untuk perangkat yang menjalankan Android versi 5.0 (API level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap Android Runtime (ART). ART ditulis guna menjalankan beberapa mesin virtual pada perangkat bermemori rendah dengan mengeksekusi file DEX, format bytecode yang didesain khusus untuk Android yang dioptimalkan untuk footprint memori minimal. Buat rantai aplikasi, misalnya Jack, mengumpulkan sumber Java ke *bytecode* DEX, yang dapat berjalan pada platform Android. Beberapa fitur utama ART mencakup:

- Kompilasi mendahului waktu (AOT) dan tepat waktu (JIT)
- Pengumpulan sampah (GC) yang dioptimalkan
- Dukungan debug yang lebih baik, mencakup profiler sampling terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

  Sebelum ke Android versi 5.0 (API level 21), Dalvik adalah waktu proses Android. Jika aplikasi Anda berjalan baik pada ART, semestinya berfungsi baik juga pada Dalvik, tetapi mungkin tidak sebaliknya. Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8, yang digunakan kerangka kerja Java API.

3. Pustaka C/C++ Asli

Banyak komponen dan layanan sistem Android inti seperti ART dan HAL dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++. Platform Android memungkinkan kerangka kerja Java API mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses OpenGL ES melalui kerangka kerja Java OpenGL API Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda. Jika Anda mengembangkan aplikasi yang memerlukan kode C atau C++, Anda bisa menggunakan Android NDK untuk mengakses beberapa pustaka platform asli langsung dari kode asli.

4. Kerangka Kerja Java API

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui API yang ditulis dalam bahasa Java. API ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

- Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan.
- Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout.
- Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status.
- Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum.
- Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

  Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android.

5. Aplikasi Sistem

Android dilengkapi dengan serangkaian aplikasi inti untuk email, perpesanan SMS, kalender, menjelajahi internet, kontak, dll. Aplikasi yang disertakan bersama platform tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna. Jadi, aplikasi pihak ketiga dapat menjadi browser web utama, pengolah pesan SMS atau bahkan keyboard utama (beberapa pengecualian berlaku, seperti aplikasi Settings sistem). Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh developer dari aplikasi mereka sendiri. Misalnya, jika aplikasi Anda ingin mengirimkan pesan SMS, Anda tidak perlu membangun fungsionalitas tersebut sendiri—sebagai gantinya Anda bisa menjalankan aplikasi SMS mana saja yang telah dipasang guna mengirimkan pesan kepada penerima yang Anda tetapkan. (Android Studio and SDK Tool", n.d.)

2.4.3 Siklus Hidup Android

Saat pengguna menavigasi, keluar, dan kembali ke aplikasi, Aktivitas akan muncul dalam transisi di aplikasi melalui berbagai status dalam siklus hidupnya. Kelas *Activity* menyediakan sejumlah panggilan balik yang memungkinkan aktivitas mengetahui bahwa suatu keadaan telah berubah bahwa sistem membuat, menghentikan, atau melanjutkan suatu aktivitas, atau menghancurkan proses di mana aktivitas berada.

Dalam metode callback siklus hidup, dapat mendeklarasikan bagaimana aktivitas berperilaku ketika pengguna meninggalkan dan masuk kembali ke aktivitas. Misalnya, jika membuat pemutar video streaming, dapat menjeda video dan memutuskan koneksi jaringan ketika pengguna beralih ke aplikasi lain. Ketika pengguna kembali, dapat menyambung kembali ke jaringan dan memungkinkan pengguna untuk melanjutkan video dari tempat yang sama. Dengan kata lain, setiap panggilan balik memungkinkan melakukan pekerjaan tertentu yang sesuai dengan perubahan kondisi yang diberikan. Melakukan pekerjaan yang tepat di waktu yang tepat dan menangani transisi dengan benar membuat aplikasi lebih kuat dan berkinerja. Misalnya, penerapan panggilan balik siklus hidup yang baik

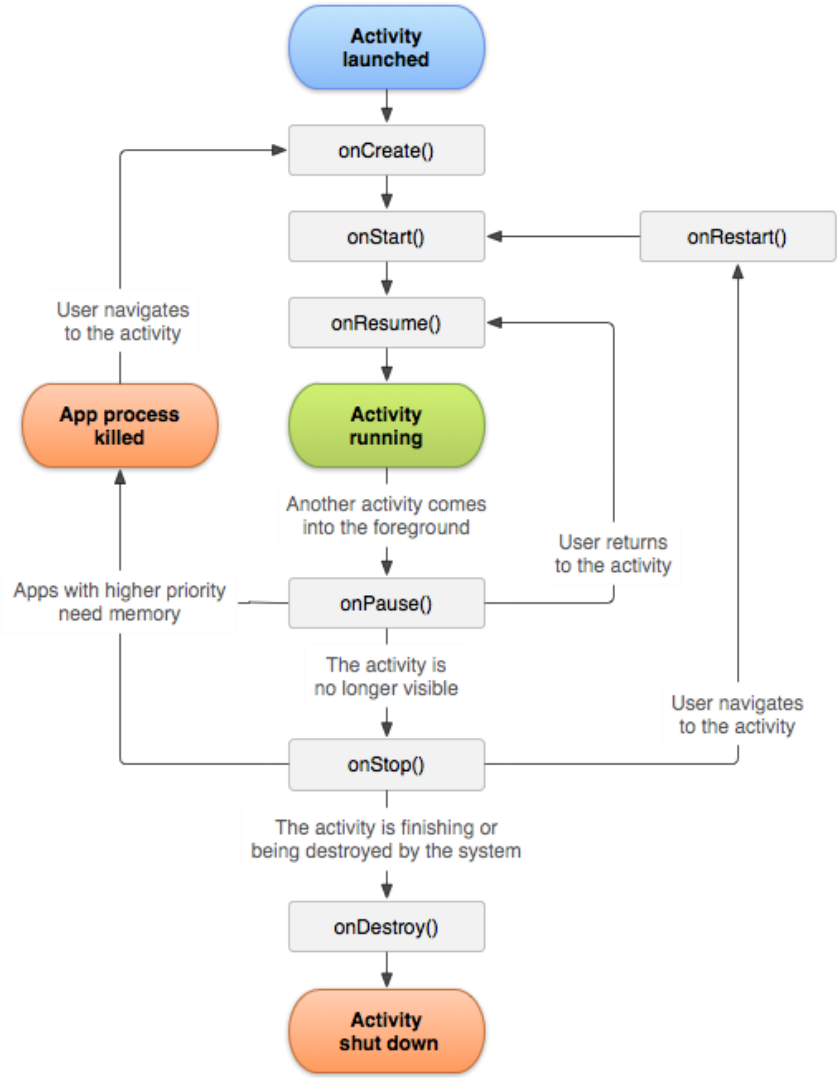
dapat membantu memastikan bahwa aplikasi menghindari: callback yang bagus dapat membantu memastikan aplikasi dihindari:

- *Crashing* jika pengguna menerima panggilan telepon atau beralih ke aplikasi lain saat menggunakan aplikasi.
- *Consuming* sumber daya sistem yang berharga saat pengguna tidak aktif menggunakannya
- *Losing* kemajuan pengguna jika mereka meninggalkan aplikasi dan kembali lagi nanti.
- *Crashing* atau kehilangan kemajuan pengguna saat layar berputar antara orientasi landscape dan portrait.

Dokumen ini menjelaskan siklus hidup aktivitas secara terperinci. Dokumen dimulai dengan menjelaskan paradigma siklus hidup. Selanjutnya, ini menjelaskan masing-masing panggilan balik: apa yang terjadi secara internal saat mereka mengeksekusi, dan apa yang harus Anda terapkan selama mereka. Ini kemudian secara singkat memperkenalkan hubungan antara keadaan aktivitas dan kerentanan suatu proses untuk dibunuh oleh sistem. Terakhir, ini membahas beberapa topik yang terkait dengan transisi antara status aktivitas. (Android Studio and SDK Tool", n.d.)

2.4.4 Konsep Siklus Hidup Aktivitas

Untuk menavigasi transisi antara tahap siklus hidup aktivitas, kelas Activity menyediakan set inti enam panggilan balik yaitu onCreate (), onStart (), onResume (), onPause (), onStop (), dan onDestroy (). Sistem memanggil masing-masing panggilan balik ini saat aktivitas memasuki keadaan baru.



Gambar 2.6 Ilustrasi sederhana dari siklus hidup aktivitas

Ketika pengguna mulai meninggalkan aktivitas, sistem memanggil metode untuk membongkar aktivitas. Dalam beberapa kasus, pembongkaran ini hanya sebagian; aktivitas masih berada dalam memori (seperti ketika pengguna beralih ke aplikasi lain), dan masih dapat kembali ke latar depan. Jika pengguna kembali ke aktivitas itu, aktivitas dilanjutkan dari tempat pengguna tadi pergi. Kemungkinan sistem untuk membunuh proses tertentu bersama dengan aktivitas di dalamnya tergantung pada keadaan aktivitas pada saat itu. Status aktivitas dan ejeksi dari memori memberikan lebih banyak informasi tentang hubungan antara status dan kerentanan terhadap ejeksi.

Bergantung pada kerumitan aktivitas, mungkin tidak perlu menerapkan semua metode siklus hidup. Namun, penting untuk memahami masing-masing dan menerapkannya yang memastikan aplikasi dapat berperilaku seperti yang diharapkan pengguna.

Bagian selanjutnya dari dokumen ini memberikan detail tentang panggilan balik yang Anda gunakan untuk menangani transisi antar negara.

2.4.5 Siklus Hidup Callback

Bagian ini memberikan informasi konseptual dan implementasi tentang metode panggilan balik yang digunakan selama siklus hidup aktivitas. Beberapa tindakan, seperti memanggil setContentView(), termasuk dalam metode siklus hidup aktivitas itu sendiri. Namun, kode yang menerapkan tindakan komponen dependen harus ditempatkan dalam komponen itu sendiri. Untuk mencapai ini, Anda harus membuat komponen yang sadar siklus hidup. Lihat Menangani Siklus Hidup dengan Komponen Siklus-Sadar untuk mempelajari cara membuat komponen dependen Anda sadar siklus.

onCreate()

Anda harus menerapkan panggilan balik ini, yang menyala saat sistem pertama kali membuat aktivitas. Pada pembuatan aktivitas, aktivitas memasuki kondisi Dibuat. Dalam metode onCreate(), Anda melakukan logika *startup* aplikasi dasar yang harus terjadi hanya sekali selama seumur hidup aktivitas. Misalnya, implementasi onCreate() mungkin mengikat data ke daftar, mengaitkan aktivitas dengan *ViewModel*, dan membuat *instance* beberapa variabel lingkup kelas. Metode ini menerima parameter *SavedInstanceState*, yang merupakan objek bundel yang berisi keadaan aktivitas yang disimpan sebelumnya. Jika aktivitas belum pernah ada sebelumnya, nilai objek Bundle adalah nol.

Contoh berikut dari metode onCreate() menunjukkan pengaturan mendasar untuk aktivitas, seperti mendeklarasikan antarmuka pengguna (didefinisikan dalam file tata letak XML), mendefinisikan variabel anggota, dan mengkonfigurasi beberapa UI. Dalam contoh ini, file tata letak XML ditentukan

dengan meneruskan ID sumber daya file R.layout.main_activity ke setContentView().

```
TextView textView;

// some transient state for the activity instance
String gameState;

@Override
public void onCreate(Bundle savedInstanceState) {
    // call the super class onCreate to complete the creation of activity like
    // the view hierarchy
    super.onCreate(savedInstanceState);

    // recovering the instance state
    if (savedInstanceState != null) {
        gameState = savedInstanceState.getString(GAME_STATE_KEY);
    }

    // set the user interface layout for this activity
    // the layout file is defined in the project res/layout/main_activity.xml file
    setContentView(R.layout.main_activity);

    // initialize member TextView so we can manipulate it later
    textView = (TextView) findViewById(R.id.text_view);
}

// This callback is called only when there is a saved instance that is previously saved by using
// onSaveInstanceState(). We restore some state in onCreate(), while we can optionally restore
// other state here, possibly usable after onStart() has completed.
// The savedInstanceState Bundle is same as the one used in onCreate().
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    textView.setText(savedInstanceState.getString(TEXT_VIEW_KEY));
}

// invoked when the activity may be temporarily destroyed, save the instance state here
@Override
public void onSaveInstanceState(Bundle outState) {
    outState.putString(GAME_STATE_KEY, gameState);
    outState.putString(TEXT_VIEW_KEY, textView.getText());

    // call superclass to save any view hierarchy
    super.onSaveInstanceState(outState);
}
```

Gambar 2.7 Metode onCreate()

Sebagai alternatif untuk mendefinisikan file XML dan meneruskannya ke setContentView (), Anda bisa membuat objek View baru dalam kode aktivitas Anda dan membangun hierarki tampilan dengan memasukkan Tampilan baru ke dalam ViewGroup. Anda kemudian menggunakan tata letak itu dengan melewatkan root ViewGroup ke setContentView(). Untuk informasi lebih lanjut tentang membuat antarmuka pengguna, lihat dokumentasi Antarmuka Pengguna. Aktivitas Anda tidak berada di negara Dibuat. Setelah metode onCreate() menyelesaikan eksekusi, aktivitas memasuki keadaan awal, dan sistem memanggil metode onStart() dan onResume() secara berurutan. Bagian selanjutnya menjelaskan panggilan balik onStart().

onStart()

Ketika aktivitas memasuki kondisi Mulai, sistem memanggil panggilan balik ini. Panggilan onStart() membuat aktivitas terlihat oleh pengguna, saat aplikasi mempersiapkan aktivitas untuk memasuki latar depan dan menjadi interaktif. Misalnya, metode ini adalah tempat aplikasi menginisialisasi kode yang mengelola UI. Metode onStart() selesai dengan sangat cepat dan, seperti pada kondisi Dibuat, aktivitas tidak tetap berada dalam status Mulai. Setelah panggilan balik ini selesai, aktivitas memasuki status Dilanjutkan, dan sistem memanggil metode onResume().

onResume()

Ketika aktivitas memasuki keadaan dilanjutkan, ia datang ke latar depan, dan kemudian sistem memanggil panggilan balik onResume(). Ini adalah keadaan di mana aplikasi berinteraksi dengan pengguna. Aplikasi tetap dalam kondisi ini sampai terjadi sesuatu untuk mengambil fokus dari aplikasi. Peristiwa semacam itu mungkin, misalnya, menerima panggilan telepon, pengguna bernavigasi ke aktivitas lain, atau layar perangkat mati. Saat aktivitas beralih ke status yang dilanjutkan, komponen sadar siklus apa pun yang terkait dengan siklus hidup aktivitas akan menerima acara ON_RESUME. Di sinilah komponen siklus hidup dapat mengaktifkan fungsionalitas apa pun yang perlu dijalankan saat komponen terlihat dan di latar depan, seperti memulai pratinjau kamera. Ketika suatu peristiwa interupsi terjadi, aktivitas memasuki keadaan dijeda, dan sistem memanggil panggilan balik onPause(). Jika aktivitas kembali ke status Lanjutkan dari kondisi Jeda, sistem sekali lagi memanggil metode onResume(). Karena alasan ini, Anda harus menerapkan onResume() untuk menginisialisasi komponen yang Anda lepaskan selama onPause(), dan melakukan inisialisasi lainnya yang harus terjadi setiap kali aktivitas memasuki keadaan dilanjutkan.:

```
public class CameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    public void initializeCamera() {
        if (camera == null) {
            getCamera();
        }
    }

    ...
}
```

Gambar 2.8 Metode onResume()

Kode di atas menempatkan kode inisialisasi kamera dalam komponen yang menyadari siklus hidup. Anda bisa menempatkan kode ini langsung ke *callback* daur hidup aktivitas seperti onStart() dan onStop() tetapi ini tidak dianjurkan. Menambahkan logika ini ke dalam komponen yang mandiri dan sadar siklus memungkinkan Anda untuk menggunakan kembali komponen tersebut di berbagai aktivitas tanpa harus menduplikasi kode. Lihat Menangani Siklus Hidup dengan Komponen Siklus-Sadar untuk mempelajari cara membuat komponen yang sadar siklus.

onPause()

Sistem memanggil metode ini sebagai indikasi pertama bahwa pengguna meninggalkan aktivitas Anda (meskipun itu tidak selalu berarti aktivitas sedang dihancurkan); itu menunjukkan bahwa aktivitas tidak lagi di latar depan (meskipun mungkin masih terlihat jika pengguna berada dalam mode multi-jendela). Gunakan metode onPause() untuk menjeda atau menyesuaikan operasi yang tidak boleh dilanjutkan (atau harus dilanjutkan dalam jumlah sedang saat Aktivitas dalam keadaan Jeda, dan Anda berharap untuk melanjutkan segera. Ada beberapa alasan mengapa suatu kegiatan dapat memasuki kondisi ini. Sebagai contoh:

- Beberapa acara mengganggu eksekusi aplikasi, seperti yang dijelaskan di bagian onResume (). Ini adalah kasus yang paling umum.

- Di Android 7.0 (API level 24) atau lebih tinggi, beberapa aplikasi berjalan dalam mode multi-jendela. Karena hanya satu aplikasi (windows) yang memiliki fokus kapan saja, sistem menjeda semua aplikasi lain.
- Aktivitas semi-transparan baru (seperti dialog) terbuka. Selama aktivitas masih terlihat sebagian tetapi tidak dalam fokus, itu tetap dijeda.

Anda juga dapat menggunakan metode onPause() untuk melepaskan sumber daya sistem, menangani sensor (seperti GPS), atau sumber daya apa pun yang dapat memengaruhi masa pakai baterai saat aktivitas Anda dijeda dan pengguna tidak membutuhkannya. Namun, seperti yang disebutkan di atas di bagian onResume(), aktivitas yang Dijeda mungkin masih sepenuhnya terlihat jika dalam mode multi-jendela. Karena itu, Anda harus mempertimbangkan menggunakan onStop() daripada onPause() untuk sepenuhnya melepaskan atau menyesuaikan sumber daya dan operasi terkait UI untuk lebih mendukung mode multi-jendela.

```
public class JavaCameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
    public void releaseCamera() {
        if (camera != null) {
            camera.release();
            camera = null;
        }
    }

    ...
}
```

Gambar 2.9 Metode onPause()

Penyelesaian metode onPause() tidak berarti bahwa aktivitas meninggalkan status Jeda. Sebaliknya, aktivitas tetap dalam kondisi ini sampai aktivitas dilanjutkan atau menjadi sama sekali tidak terlihat oleh pengguna. Jika aktivitas dilanjutkan, sistem sekali lagi memanggil panggilan balik onResume(). Jika aktivitas kembali dari status Jeda ke status Dilanjutkan, sistem menyimpan instance instance tetap dalam memori, mengingat instance tersebut ketika sistem memanggil onResume(). Dalam skenario ini, Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan balik yang mengarah ke status Lanjutkan. Jika aktivitas menjadi benar-benar tidak terlihat,

sistem memanggil onStop (). Bagian selanjutnya membahas panggilan balik onStop().

onStop()

Ketika aktivitas Anda tidak lagi terlihat oleh pengguna, itu telah memasuki status Berhenti, dan sistem memanggil panggilan balik onStop(). Ini dapat terjadi, misalnya, ketika aktivitas yang baru diluncurkan mencakup seluruh layar. Sistem juga dapat memanggil onStop() ketika aktivitas telah selesai berjalan, dan akan dihentikan.

Dalam metode onStop(), aplikasi harus melepaskan atau menyesuaikan sumber daya yang tidak diperlukan saat aplikasi tidak terlihat oleh pengguna. Misalnya, aplikasi Anda dapat menjeda animasi atau beralih dari pembaruan lokasi berbutir halus ke berbutir kasar. Menggunakan onStop() dan bukan onPause() memastikan bahwa pekerjaan terkait UI berlanjut, bahkan ketika pengguna melihat aktivitas Anda dalam mode multi-jendela.

Anda juga harus menggunakan onStop() untuk melakukan operasi shutdown yang relatif intensif CPU. Misalnya, jika Anda tidak dapat menemukan waktu yang lebih tepat untuk menyimpan informasi ke database, Anda mungkin melakukannya selama onStop(). Contoh berikut menunjukkan implementasi onStop() yang menyimpan konten catatan konsep ke penyimpanan persisten:

```
@Override
protected void onStop() {
    // call the superclass method first
    super.onStop();

    // save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    // do this update in background on an AsyncQueryHandler or equivalent
    asyncQueryHandler.startUpdate (
            mToken,  // int token to correlate calls
            null,    // cookie, not used here
            uri,    // The URI for the note to update.
            values,  // The map of column names and new values to apply to them.
            null,    // No SELECT criteria are used.
            null     // No WHERE columns are used.
    );
}
```

Gambar 2.10 Metode onStop()

Ketika aktivitas Anda memasuki status Berhenti, objek Aktivitas disimpan di dalam memori: Objek menyimpan semua informasi status dan anggota, tetapi tidak dilampirkan ke manajer jendela. Ketika aktivitas dilanjutkan, aktivitas mengingat informasi ini. Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan balik yang mengarah ke status Lanjutkan. Sistem juga melacak keadaan saat ini untuk setiap objek Lihat dalam tata letak, jadi jika pengguna memasukkan teks ke dalam widget EditText, konten tersebut dipertahankan sehingga Anda tidak perlu menyimpan dan mengembalikannya.

onDestroy()

onDestroy() dipanggil sebelum aktivitas dihancurkan. Sistem memanggil panggilan balik ini karena:

- aktivitas sedang selesai (karena pengguna benar-benar mengabaikan aktivitas atau karena selesai () dipanggil pada aktivitas).
- sistem sementara menghancurkan aktivitas karena perubahan konfigurasi (seperti rotasi perangkat atau mode multi-jendela).

Jika aktivitas selesai, onDestroy() adalah panggilan balik siklus hidup terakhir yang diterima aktivitas. Jika onDestroy() dipanggil sebagai hasil dari perubahan konfigurasi, sistem segera membuat instance aktivitas baru dan kemudian memanggil onCreate() pada instance baru dalam konfigurasi baru. Callback onDestroy() harus melepaskan semua sumber daya yang belum dirilis oleh callback sebelumnya seperti onStop(). (Android Studio and SDK Tool", n.d.).

2.4.6 Android Studio

Android Studio adalah IDE resmi untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain kemampuan yang Anda harapkan dari IntelliJ, Android Studio menawarkan:

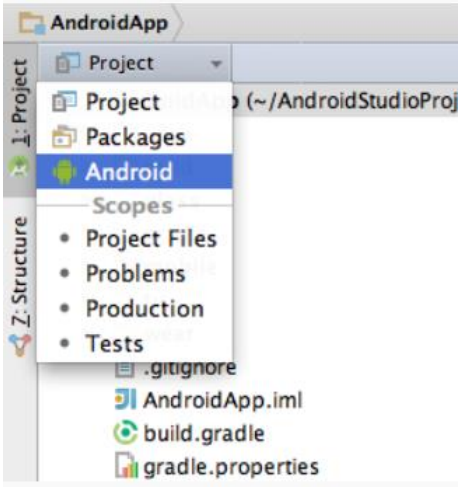
- Sistem pembangunan berbasis Gradle fleksibel.
- Buat varian dan beberapa pembuatan *file* apk.

- *Template* kode untuk membantu Anda membangun fitur aplikasi umum.
- Editor tata letak kaya dengan dukungan untuk mengedit tema *drag and drop*.
- Alat serat untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya.
- Kemampuan ProGuard dan penandatanganan aplikasi.
- Dukungan bawaan untuk Google Cloud Platform, membuatnya mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine.

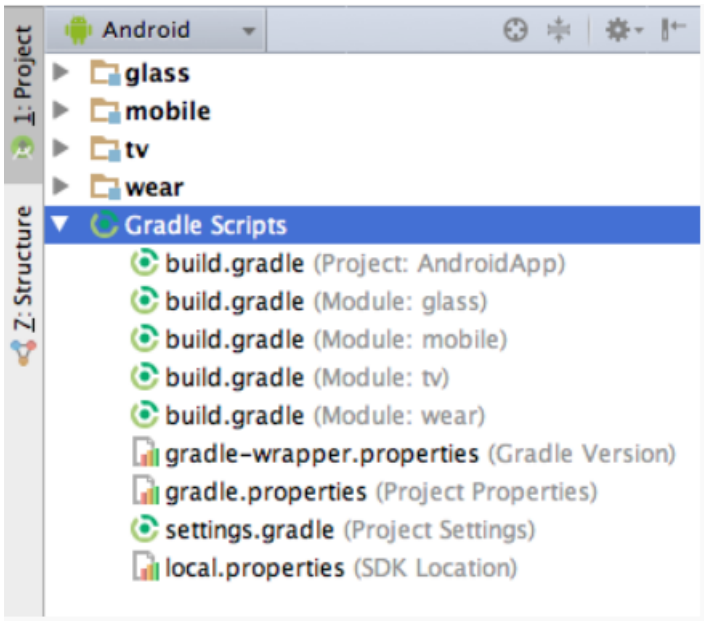
1. **Struktur Proyek**

Secara default, Android Studio menampilkan file profil Anda dalam tampilan proyek Android. Tampilan ini menunjukkan versi rata dari struktur proyek Anda yang menyediakan akses cepat ke file sumber utama proyek Android dan membantu Anda bekerja dengan sistem build berbasis Gradle yang baru. Tampilan proyek Android:

- Grup file build untuk semua modul di tingkat atas hirarki proyek.
- Memperlihatkan direktori sumber paling penting di tingkat atas hirarki modul.
- Grup semua file manifes untuk setiap modul.
- Memperlihatkan file sumber daya dari semua set sumber Gradle.
- Grup file sumber daya untuk berbagai lokal, orientasi, dan jenis layar dalam satu grup per jenis sumber daya



Gambar 2.11 File proyek di tampilan Android



Gambar 2.12 File pembuatan proyek Android

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

- Modul aplikasi Android
- Modul Pustaka
- Modul Google *App Engine*

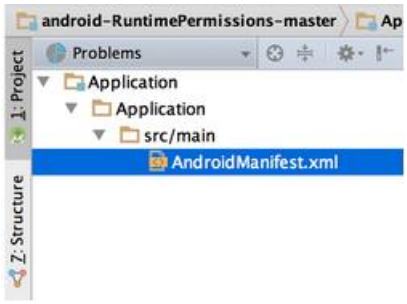
Secara default, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar 1. Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke file sumber utama proyek Anda.

Semua file versi terlihat di bagian atas di bawah **Gradle Scripts** dan masing-masing modul aplikasi berisi folder berikut:

- manifests: Berisi file *AndroidManifest.xml.*
- java: Berisi file kode sumber Java, termasuk kode pengujian JUnit.
- res: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar bitmap.

Sruktur proyek Android pada disk berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih **Project** dari menu tarik turun **Project** (dalam gambar 1, struktur ditampilkan sebagai **Android**).

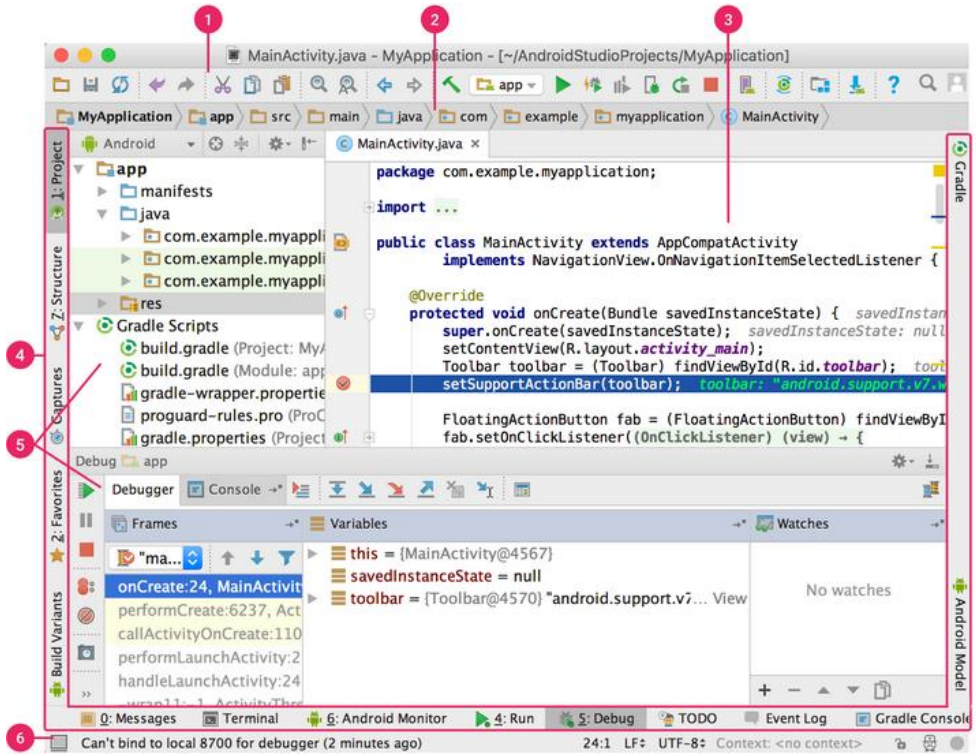
Anda juga bisa menyesuaikan tampilan file proyek untuk berfokus pada aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih tampilan **Problems** dari tampilan proyek Anda akan menampilkan tautan ke file sumber yang berisi kesalahan pengkodean dan sintaks yang dikenal, misalnya tag penutup elemen XML tidak ada dalam file tata letak.



Gambar 2.13 File proyek di tampilan Problem.

2. Antarmuka Pengguna

Jendela utama Android Studio terdiri dari beberapa bidang logika yang diidentifikasi dalam gambar berikut.



Gambar 2.14 Jendela utama Android Studio

1) Bilah alat memungkinkan Anda untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat Android.
2) Bilah navigasi membantu Anda bernavigasi di antara proyek dan membuka file untuk diedit. Bilah ini memberikan tampilan struktur yang terlihat lebih ringkas dalam jendela Project.
3) Jendela editor adalah tempat Anda membuat dan memodifikasi kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor menampilkan Layout Editor.
4) Bilah jendela alat muncul di luar jendela IDE dan berisi tombol yang memungkinkan Anda meluaskan atau menciutkan jendela alat individual.
5) Jendela alat memberi Anda akses ke tugas tertentu seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi. Anda bisa meluaskan dan juga menciutkannya.
6) Bilah status menampilkan status proyek Anda dan IDE itu sendiri, serta setiap peringatan atau pesan.

Anda bisa menata jendela utama untuk memberi Anda ruang layar yang lebih luas dengan menyembunyikan atau memindahkan bilah alat dan jendela alat. Anda juga bisa menggunakan pintasan keyboard untuk mengakses sebagian besar fitur IDE.

Anda dapat menelusuri seluruh kode sumber, basis data, tindakan, elemen antarmuka pengguna, dan seterusnya setiap saat dengan menekan tombol Shift dua kali, atau mengeklik kaca pembesar di sudut kanan atas dari jendela Android Studio. Ini akan sangat berguna misalnya saat Anda mencoba menemukan tindakan IDE tertentu yang Anda lupakan cara memicunya. (Android Studio Overview, n.d.)

Jendela Alat

Dari pada menggunakan perspektif yang sudah diatur sebelumnya, Android Studio mengikuti konteks Anda dan secara otomatis memunculkan jendela alat yang relevan saat Anda bekerja. Secara default, alat yang tersering dipakai akan disematkan ke bilah jendela alat di tepi jendela aplikasi.

- Untuk meluaskan atau menciutkan jendela alat, klik nama alat di bilah jendela alat. Anda juga bisa menyeret, menyematkan, melampirkan, dan melepaskan jendela alat.
- Untuk kembali ke tata letak jendela alat default saat ini, klik *Window* > *Restore Default Layout* atau sesuaikan tata letak default Anda dengan mengeklik *Window* > *Store* Current Layout as Default.
- Untuk kembali ke tata letak jendela alat default saat ini, klik **Window > Restore Default Layout** atau sesuaikan tata letak default Anda dengan mengeklik **Window > Store Current Layout as Default**.
- Untuk menampilkan atau menyembunyikan bilah jendela alat, klik ikon jendela di sudut kiri bawah jendela Android Studio
- Untuk menemukan jendela alat tertentu, arahkan ke atas ikon jendela dan pilih jendela alat tersebut dari menu.

Anda juga bisa menggunakan pintasan keyboard untuk membuka jendela alat. Tabel 2.1 mencantumkan pintasan jendela paling umum.

Tabel 2.1 Keyboard shortcut ke beberapa jendela alat yang penting

Jendela Alat	Windows dan Linux	Mac
Proyek	Alt+1	Command+1
Kontrol Versi	Alt+9	Command+9
Run	Shift+F10	Control+R
Debug	Shift+F9	Control+D
Android Monitor	Alt+6	Command+6
Kembali ke Editor	Esc	Esc
Menyembunyikan Semua Jendela Alat	Control+Shift+F12	Esc

Jika Anda ingin menyembunyikan semua bilah alat, jendela alat, dan tab editor, klik *View > Enter Distraction Free Mode*. Ini akan mengaktifkan *Distraction Free Mode*. Untuk keluar dari *Distraction Free Mode*, klik *View > Exit Distraction Free Mode*.

Anda bisa menggunakan *Speed Search* untuk menelusuri dan memfilter di dalam sebagian besar jendela alat dalam Android Studio. Untuk menggunakan *Speed Search*, pilih jendela alat lalu ketik kueri penelusuran Anda.

Pelengkapan Kode

Android Studio memiliki tiga jenis pelengkapan kode, yang bisa anda akses memakai pintasan keyboard.

Tabel 2.2 Keyboard shortcut untuk pelengkapan code

Tipe	Keterangan	Windows dan Linux	Mac
Pelengkapan Dasar	Menampilkan saran dasar untuk variabel, tipe, metode, ekspresi, dan seterusnya. Jika Anda memanggil pelengkapan dasar dua kali secara berturutturut, Anda melihat lebih banyak hasil, termasuk anggota pribadi dan anggota statis yang tidak diimpor.	Control+Space	Control+Space
Pelengkapan Cerdas	Menampilkan opsi relevan berdasarkan konteks. Pelengkapan cerdas mengetahui tipe yang diharapkan dan alur data. Jika Anda memanggil Pelengkapan Cerdas dua kali berturut-turut, Anda akan melihat lebih banyak hasil, termasuk rantai.	Control+Shift+Space	Control+Shift+Space
Pelengkapan Pernyataan	Membantu Anda melengkapi pernyataan saat ini, menambahkan tanda kurung, tanda kurung siku, tanda kurung kurawal, pemformatan, dsb.	Control+Shift+Enter	Shift+Command+Enter

Anda juga bisa melakukan perbaikan cepat dan menunjukkan tindakan maksud Anda dengan menekan Alt+Enter.

Navigasi

Berikut beberapa tip untuk membantu Anda menjelajahi di dalam Android Studio.

- Beralih antar file yang baru saja diakses menggunakan tindakan *Recent Files*. Tekan *Control+E* (*Command+E* pada Mac) untuk memunculkan tindakan *Recent Files*. Secara default, akses yang terakhir dipilih. Anda juga bisa mengakses jendela alat mana saja melalui kolom kiri dalam tindakan ini.
- Tampilkan struktur file saat ini menggunakan tindakan *File Structure*. Munculkan tindakan *File Structure* dengan menekan *Control+F12*

(*Command+F12* pada Mac). Menggunakan tindakan ini, Anda bisa menavigasi dengan cepat ke bagian mana pun dari file Anda saat ini.

- Telusuri dan masuk ke kelas tertentu di proyek menggunakan *tindakan Navigate to Class*. Munculkan tindakan dengan menekan *Control+N* (*Command+O* pada Mac). Navigasikan ke Kelas yang mendukung ekspresi canggih, termasuk *CamelHumps*, jalur, baris menavigasi ke, nama tengah pencocokan, dan banyak lagi. Jika Anda memanggilnya dua kali berturut-turut, hasil dari kelas proyek akan ditampilkan.
- Masuk ke file atau folder menggunakan tindakan "*Navigate to File**. Munculkan tindakan *Navigate to File* dengan menekan *Control+Shift+N* (*Command+Shift+O* pada Mac). Untuk menelusuri folder dan bukan file, tambahkan / di akhir ekspresi Anda.
- Masuk ke metode atau bidang menurut nama menggunakan tindakan *Navigate to Symbol*. Munculkan tindakan *Navigate to Symbol* dengan menekan *Control+Shift+Alt+N* (*Command+Shift+Alt+O* pada Mac).
- Temukan semua bagian kode yang merujuk kelas, metode, bidang, parameter, atau pernyataan di posisi kursor saat ini dengan menekan Alt+F7

Gaya dan Pemformatan

Saat Anda mengedit, Android Studio otomatis menerapkan pemformatan dan gaya seperti yang ditetapkan dalam setelan gaya kode. Anda dapat menyesuaikan setelan gaya kode dengan bahasa pemrograman, termasuk menetapkan konvensi untuk tab dan inden, spasi, pembungkusan dan tanda kurung kurawal, dan baris kosong. Untuk menyesuaikan setelan gaya kode, *klik File > Settings > Editor > Code Style (Android Studio > Preferences > Editor > Code Style* pada Mac.) Meski IDE otomatis menerapkan pemformatan saat Anda bekerja, Anda juga dapat secara eksplisit memanggil tindakan Reformat Code dengan menekan *Control+Alt+L* (*Opt+Command+L* pada Mac), atau inden otomatis semua baris dengan menekan *Control+Alt+I* (*Alt+Option+I* pada Mac).

Dasar-dasar Kontrol Versi

Android Studio mendukung berbagai versi sistem kontrol, termasuk *Git, GitHub, CVS, Mercurial, Subversion*, dan Penyimpanan *Google Cloud Source*.

Setelah mengimpor aplikasi Anda ke dalam Android Studio, gunakan opsi menu Android Studio VCS untuk mengaktifkan dukungan VCS bagi sistem kontrol versi yang diinginkan, membuat penyimpanan, mengimpor file baru ke dalam kontrol versi, dan melakukan pengoperasian kontrol versi lainnya:

- Dari menu Android Studio VCS, klik *Enable Version Control Integration*.
- Dari menu tarik-turun, pilih sistem kontrol versi yang terkait dengan akar proyek, lalu klik OK.
- Menu VCS sekarang menunjukkan sejumlah opsi kontrol versi berdasarkan sistem yang Anda pilih.

2.4.7 Sistem Versi Gradle

Android Studio menggunakan Gradle sebagai dasar sistem versi, dengan kemampuan khusus Android yang disediakan oleh Plugin Android untuk Gradle. Sistem ini bisa dijalankan sebagai alat terpadu dari menu Android Studio dan secara independen dari baris perintah. Anda bisa menggunakan fitur-fitur sistem versi untuk melakukan yang berikut:

- Menyesuaikan, mengonfigurasi, dan memperluas proses pembangunan.
- Membuat beberapa APK untuk aplikasi Android Anda, dengan aneka fitur menggunakan proyek dan modul yang sama. Menggunakan kembali kode dan sumber daya pada seluruh set sumber

Dengan menerapkan fleksibilitas Gradle, Anda dapat mencapai semua ini tanpa mengubah file sumber inti aplikasi. File versi Android Studio diberi nama build.gradle. File ini adalah teks biasa yang menggunakan Groovy mengonfigurasi versi dengan elemen yang disediakan oleh plugin Android untuk Gradle. Masing-masing proyek memiliki file versi level atas untuk seluruh proyek dan file versi level modul terpisah untuk setiap modul. Saat Anda mengimpor proyek saat ini, Android Studio otomatis menghasilkan file versi yang diperlukan.

Varian Versi

Sistem versi dapat membantu Anda membuat versi berbeda dari aplikasi yang sama dari satu proyek. Ini berguna ketika Anda sama-sama memiliki versi gratis dan versi berbayar dari aplikasi, atau jika Anda ingin mendistribusikan beberapa APK untuk perangkat berbeda di Google Play.

Pemisahan APK

Pemisahan APK memungkinkan Anda untuk membuat beberapa APK berdasarkan kepadatan layar atau ABI. Misalnya, pemisahan APK memungkinkan Anda membuat versi hdpi dan mdpi terpisah dari aplikasi sembari masih mempertimbangkannya sebagai satu varian dan memungkinkannya untuk berbagi setelan aplikasi pengujian, javac, dx, dan ProGuard.

Penyusutan Sumber Daya

Penyusutan sumber daya di Android Studio secara otomatis membuang sumber daya yang tidak terpakai dari aplikasi terkemas dan dependensi perpustakaan. Misalnya, jika aplikasi Anda menggunakan layanan Google Play untuk mengakses fungsi Google Drive, dan saat ini Anda tidak memakai Google Sign-In, maka penyusutan sumber daya dapat membuang berbagai aset yang dapat digambar untuk tombol SignIn Button.

Mengelola Dependensi

Dependensi untuk proyek Anda ditetapkan oleh nama dalam file build.gradle. Gradle menangani penemuan dependensi Anda dan menyediakannya di versi Anda. Anda bisa mendeklarasikan dependensi modul, dependensi biner jarak jauh, dan dependensi biner setempat dalam file build.gradle Anda. Android Studio mengonfigurasi proyek untuk menggunakan Penyimpanan Pusat Maven secara default. (Konfigurasi ini disertakan dalam file versi tingkat atas untuk proyek tersebut). (Android Studio Overview, n.d.)

2.5 Java

Java merupakan Bahasa pemrograman tingkat tinggi yang dipelopori oleh James Gosling yang merupaka enginer di Sun Microsystem. Java mulai dibangun pada tahun 1991. Versi alpha dan beta dari java dirilis pada tahun 1995, 4 tahun setelah projek Java diinisiasi. Pada tahun 2010, Sun Microsystem diakusisi oleg Oracle dan menjadikan Java dikembangkan di bawah kuasa Oracle.

Per Januari 2018, versi stabil terkahir dari Java yaitu versi Java SE 9. Sebelumnya Java SE 9, terdapat beberapa versi dari Java yang telah dirilis. Versi – versi dari java yang pernah dirilis yaitu :

- JDK Aplha dan Beta (1995)
- JDK 1.0 (23 Januari 1996)
- JDK 1.1 (19 Februari 1997)
- J2SE 1.2 (8 Desember 1998)
- J2SE 1.3 (8 Mei 200)
- J2SE 1.4 (6 Februari 2002)
- J2SE 5.0 (30 September 2004
- Java SE 6 (11 Desember 2006)
- Java SE 7 (18 Juli 2011)
- Java SE 8 (18 Maret 2014)
- Java SE 9 (21 September 2017)

Java merupakan salah satu Bahasa yang popular saat ini. Saking populernya, beberapa edisi untuk Java pu diciptakan, contohnya yaitu J2EE untuk Aplikasi Enterprise dan J2ME untuk Aplikasi Mobile. Salah satu alasan lain mengapa Java merupakan bahasa yang popular yaitu dikarenakan Java dapat berjalan di bebagai platfrom system operasi. Java pun dikenal dengan *Write Once, Run Anywhere* karena kompatibilitasnya tersebut. Menuru Sun yang merupakan pelopor dari Java, Java itu:

Simpel

Java merupakan Bahasa yang simple sehingga mudah dipahami. Bermodalkan pengetahuan programming dasar, fundamental dari pemrograman java akan cepat dipahamai

Berorientasi Objek

Java menggunakan konsep pemrograman berorientasi objek sehingga pembuat aplikasi bisa dilakukan lebih modular.

Kuat

Java didesain untuk membuat aplikasi yang memiliki reliabilitas tinggi. Salah satu cara Java untuk membuat program yang memiliki reliabilitas tinggi yaitu dengan mengeliminasi situasi error dengan memeriksanya pada compile time dan runtime.

Aman

Java didesain untuk digunakan pada lingkungan yang terdistribusi. Keamanan merupakan hal yang penting dalam lingkungan terdistribusi. Fitur – fitur keamanan yang dimiliki Java membuat perangkat lunak yang dibuat tidak bisa diserang dari luar atau disisipi virus.

Arsitektur Netral

Aplikasi yang dibuat menggunakan Java merupakan aplikasi yang platfrom independent. Aplikasi hanya perlu satu buah versi yang bisa dijalankan pada platfrom sistem operasi yang berbeda.

Portable

Java dengan karakteristik arsitektur netralnya membuat Java tidak bergantung pada mesin tertentu atau dengan kata lain Java sangatlah portable.

Perfoma Tinggi

Java sangat memperhatikan perfoma. Dengan pengenalan *Just in Time compilation*, proses kompilasi Java menjadi lebih cepat.

Multithreaded

Java memungkinkan pembuatan aplikasi yang bisa melakukan beberapa pekerjaan secara bersamaan.

Dinamis

Java lebih dinamis dibandingan C atau C++ karena didesain untuk dijalankan pada langkungan yang dinamis. (Adam & Firman, 2018)

2.5.1 Sejarah Java

Sejarah Java dimulai pada tahun 1991. Sebuah tim kecil yang berisikan engineer dari Sun Microsystem mempelopori proyek Java ini. Tim ini disebut Green Team dan dipimpin oleh James Gosling. Saat awal pengembangag,, bahasa yang dikembangkan oleh Green Team disebut nama “Greentalk”. Setelah itu, mereka mengganti Namanya menjadi “Oak” ketika pohon oak yang tiba-tiba muncul di luar kantornya. Pada tahun 1995, Green Team menggnati Namanya menjadi “Java” karena Oak sudah menjadi trademark dari Oak Technologies. Pada tahun itu, Sun Microsystem lalu merilis Java untuk pertama kali.Pada 13 November 2006, Sun Microsystem merilis Java dengan gratis dan open source dengan lisensi GNU General Public License (GPL). Tetapi pada tahun 2010, Sun Microsystem dibeli oleh Oracle dan menjadikan Java dikembangkan dan dipelihara di bawah kendali Oracle. (Adam & Firman, 2018)

2.5.2 Spesifikasi Java

Spesifikasi minimum untuk komputer menjalankan Java yaitu Pentium 2 266 MHz dengan RAM 128 MB. Selain itu, perlu juga menyiapkan software berikut: (Adam & Firman, 2018)

- Sistem operasi Windows XP/7/8/10 atau Ubuntu Linux minimal versi 12.04 atau Mac OS X minimal versi 10.8.3.
- Java SE 8
- IDE atau text editor seperti Intellij IDEA, Netbeans, Eclipse, Visual Studio Code, Sublime, Atom, dan sebagainya.

2.6 API

Application Programming Interface (API) adalah seperangkat prosedur, fungsi, protokol dan aturan untuk membangun sebuah peangkat lunak. Secara umum, API menentukan bagaimana komponen perangkat lunak atau paket yang berbeda harus berinteraksi satu sama lain meskipun dengan bahasa yang sangat berbeda.[18] API yang bagus memudahkan pengembangan program komputer dengan menyediakan semua blok bangunan, yang kemudian disatukan oleh pemrogram. API digunakan untuk sistem berbasis web, sistem operasi, sistem basis data, perangkat keras komputer atau perpustakaan perangkat lunak. Spesifikasi API dapat mengambil banyak bentuk, namun sering kali mencakup spesifikasi untuk rutinitas, struktur data, kelas objek, variabel atau panggilan jarak jauh. *POSIX*, *Microsoft Windows* API, *C ++ Standard Template Library dan Java* API adalah contoh dari berbagai bentuk API. Dokumentasi untuk API biasanya disediakan untuk memudahkan penggunaan.

2.6.1 GoIndoor API

GoIndoor API adalah *Web Service* yang menawarkan *real-time indoor navigation tools* secara gratis pada GoIndoor *software*, namun pada GoIndoor *hardware* ada batasan. Adapun jenis dari GoIndoor API, yaitu GoIndoor software dan GoIndoor Hardware.

2.6.1.1 GoIndoor Hardware

Hardware yang digunakan adalah Beacon. Beacon adalah *Bluetooth Low Energy* (BLE) yang memancarkan sinyal radio *bluetooth* secara berkala yang menggunakan daya baterai koin, atau dengan daya lain seperti baterai AA dan

USB. Pada GoIndoor API terdapat batasan untuk penggunaan Beacon, yaitu apabila Beacon yang dipasang lebih dari 5, maka penggunaan GoIndoor menjadi berbayar. Untuk lebih jelas bisa dilihat pada Gambar 2.15

NUMBER OF BEACONS	MONTHLY PRICE PER BEACON
0 - 5	Free - Try it now!
6 - 15	7.0 €
16 - 30	6.0 €
31 - 50	5.3 €
51 - 100	4.8 €
101 - 250	4.4 €
251 - 500	4.0 €
Above 500	Contact us for more details

The above prices are for licenses only. We provide beacons only with our Starter Pack – free starting license + 5 beacons for 99 €.

Gambar 2.15 Harga Beacon pada GoIndoor API

Beacon digunakan untuk mengganti GPS(Global Positioning System) pada ruangan, karena GPS tidak dapat digunakan atau kurang baik untuk menentukan POI's (Point of Interests) dari pengguna. Berbagai tipe dan merek beacon tersedia. Adapun merek beacon yang didukung oleh GoIndoor API, yaitu :

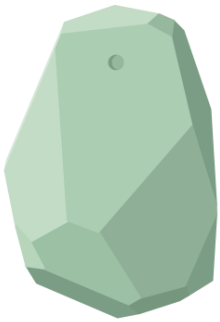
1. iBeacon

   iBeacon adalah nama untuk standar teknologi Apple, yang memungkinkan Aplikasi *smartphone* (berjalan pada perangkat iOS dan Android) untuk mendengarkan sinyal dari gelombang yang dikeluarkan iBeacon.[16]

Gambar 2.16 iBeacon

2. Eddystone Beacon

Eddystone Beacon adalah nama untuk standar teknologi Google, yang memungkinkan Aplikasi *smartphone* (berjalan pada perangkat iOS dan Android) untuk mendengarkan sinyal dari gelombang yang dikeluarkan Eddystone Beacon.[17]



Gambar 2.17 Eddystone Beacon

Bentuk dari kedua merek yang disebutkan sebelumnya memiliki cangkang atau bentuk yang serupa, yang membedakan hanya produksi beacon dan chipnya.

2.6.1.1 GoIndoor Software

GoIndoor API Menawarkan *real-time indoor navigation tools* dan membantu dalam pemetaan didalam ruangan serta memiliki fitur, antara lain:

- Keakuratan dalam menentukan posisi dalam ruangan
- Pembuatan rute yang mudah
- Transisi lantai
- Pengelolaan beacon
- Pemberian notifikasi

Cara kerja GoIndoor API yaitu harus adanya koneksi antara *bluetooth* dari *smartphone* dengan Beacon. Sinyal Beacon yang dipancarkan akan diterima melalui *bluetooth smartphone* kemudian GoIndoor API SDK akan melakukan pemrosesan untuk menghitung keakuratan lokasi pada smartphone. Informasi lokasi yang telah diperoleh dapat digunakan dalam berbagai aplikasi dengan

berbagai cara, misalnya untuk penelitian saat ini akan dijadikan *indoor navigation* di UNIKOM.

Pada saat pembangunan aplikasi menggunakan GoIndoor API, Beacon harus diletakkan di dalam gedung. Dipastikan pemasangan Beacon tidak terhalang oleh benda di sekitarnya, lebih baik pemasangan dilakukan di permukaan *ceiling*(langit-langit). Diperlukan juga denah gedung yang terlibat dengan pembangunan aplikasi ini. Kemudian denah tersebut diletakkan sesuai dengan lokasi bangunan sesungguhnya. Setelah denah tersebut sesuai dengan lokasi sesungguhnya, lalu menambahkan lantai, menentukan rute, menentukan rute untuk transisi tiap lantai, pemberian notifikasi. Setelah denah gedung atau ruangan sudah ada, yang perlu dilakukan adalah menempatkan Beacon pada GoIndoor API sesuai dengan penempatan Beacon yang aslinya.[12]

2.6.1.1 Penempatan Beacon - Bangunan

Penempatan Beacon adalah langkah yang sangat penting untuk memastikan akurasi yang diinginkan tercapai. Jumlah Beacon, orientasinya, dan ketinggian tempat penempatannya adalah beberapa faktor yang berkontribusi terhadap kinerja keseluruhan sistem penempatan dalam ruangan berbasis Beacon. Ingatlah faktor-faktor berikut saat sedang menempatkan Beacon.

1. **Orientasi Beacon**

   Cara beacon diorientasikan memiliki pengaruh besar terhadap kekuatan sinyal. Salah satu kesalahan umum adalah menguji Beacon dengan hanya meletakkannya di atas perabot misalnya. Jika Beacon diarahkan ke atas, sinyal sangat lemah. Orientasikan dengan benar dengan menempelkannya ke langit-langit atau di dinding untuk hasil terbaik.

Gambar 2.18 Orientasi Beacon di langit-langit

Gambar 2.19 Orientasi Beacon di dinding

2. Pengaturan Beacon

Daya tx dan tingkat pembaruan adalah dua parameter yang menentukan Beacon. Kekuatan Tx memiliki pengaruh pada jangkauan Beacon dan laju pembaruan memengaruhi stabilitas sinyal. Memiliki daya tx yang tinggi dapat menguras baterai Beacon dengan sangat cepat. Pada saat yang sama, sinyalnya tidak akan berguna untuk perangkat yang jauh karena sulit untuk menghitung jarak ke Beacon melebihi 20 meter. Tidak mendengar Beacon bisa menjadi informasi yang berharga tetapi terlalu banyak mendengar Beacon dapat mengurangi stabilitas lokasi. Untuk banyak bangunan,

menggunakan -56dBm dan 500ms untuk laju dan frekuensi tx. Dengan laju dan tx tersebut, baterai dari beacon kurang lebih dapat bertahan ±180 hari.

Tabel 2.3 Pengaturan *default* Beacon

Attributes	Factory settings
Name	51822：RDL51822
Default broadcast	1000ms
Default power setting	4dBm
Usage time	190 days

3. Tinggi terbaik

Untuk menghindari rintangan antara perangkat Anda dan Beacon, letakkan setidaknya setinggi 3 meter untuk mendapatkan garis pandang yang jelas. Pada ketinggian 5 meter dan lebih, sinyalnya melemah. Dalam hal ini, tambah daya tx dari pengaturan default (sekitar -56dBm).

Gambar 2.20 Tinggi pemasangan Beacon

4. Rintangan dan gangguan

Sinyal dapat diblokir atau diserap oleh rintangan. Logam dan air memiliki efek luar biasa. Bahkan suhu ekstrem dapat memengaruhi sinyal Beacon yang mengakibatkan sinyal yang di pancarkan kurang maksimal dan aplikasi tidak dapat menerima sinyal dengan sempurna serta dapat mengurangi

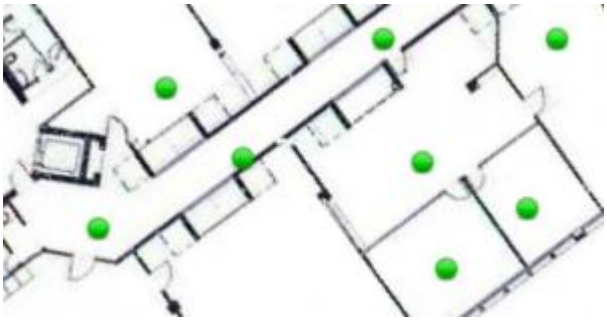
keakuratan POI's(*Point of Interests*) dari pengguna, maka dari itu letakkan beacon pada ruangan terbuka.

5. Jumlah Beacon

Terlalu sedikit Beacon dapat meninggalkan beberapa tempat di lokasi Anda terbuka yang menyebabkan ada lokasi yang tidak terdeteksi oleh beacon sehingga koneksi antara smartphone dengan Beacon terputus. Pada saat yang sama, terlalu banyak Beacon dapat memengaruhi keakuratan posisi yang dihitung dan menyebabkan sinyal yang diterima oleh *smartphone* tumpang tindinh antara sinyal beacon satu dengan yang lain. Sebaiknya penempatan beacon diberi jarak 10-15 meter seperti pada Gambar 2.21 dan Gambar 2.23.

2.6.1.1 Penempatan Beacon – Banyak Ruangan

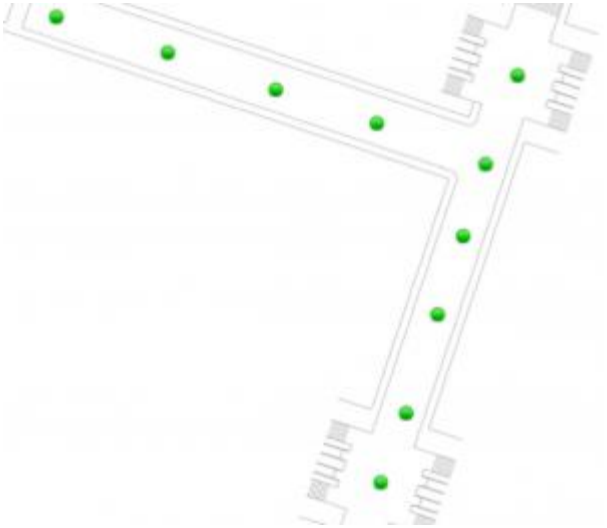
Bangunan biasanya terdiri dari ruangan dan koridor. Untuk ruangan , itu tergantung pada apa yang ingin Anda capai. Biasanya untuk sebuah ruangan, satu Beacon sudah cukup untuk dapat bernavigasi ke ruangan atau memicu pemberitahuan berdasarkan itu. Anda juga dapat memilih untuk meletakkan Beacon di koridor di luar ruangan jika hanya navigasi ke ruangan itu yang diperlukan.



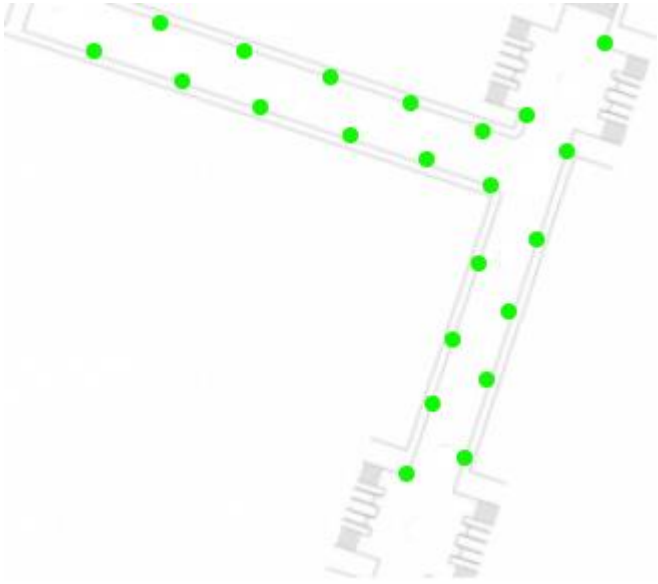
Gambar 2.21 Sampel - ruangan masing-masing dilengkapi dengan satu Beacon

2.6.1.1 Penempatan Beacon – Koridor

Untuk **koridor** , yang terbaik adalah meletakkan Beacon di langit-langit setiap 10-15 meter. Namun, jika Anda tidak bisa meletakkannya di langit-langit, letakkan di dinding. Letakkan satu Beacon di sebelah kiri, yang berikutnya di sebelah kanan, dan ulangi prosesnya. Ingatlah untuk selalu meletakkan satu Beacon di persimpangan jalan. Akurasi lebih baik ketika Anda dekat dengan Beacon, dan jenis lokasi ini penting untuk navigasi.



Gambar 2.22 Sampel koridor langit-langit



Gambar 2.23 Sampel koridor dinding

2.6.1.1 Penempatan Beacon - Ruang Terbuka

Dalam ruang seperti itu, di mana Anda harus menempatkan banyak Beacon untuk menutupi seluruh tempat secara efektif, faktor-faktor seperti gangguan masuk ke dalam gambar. Mainkan dengan kekuatan sinyal untuk mengurangi tumpang tindih. Praktik terbaik untuk tempat-tempat tersebut adalah membuat kotak Beacon simetris untuk menutupi seluruh tempat. Kepadatan kisi akan menentukan akurasi dalam penentuan lokasi.

Sebagai praktik terbaik, direkomendasikan bahwa jarak antara setiap Beacon harus antara 10 dan 20 meter. Ketika Beacon berjarak 10 meter, ia akan memberi Anda akurasi sekitar 2 meter. 20 meter akan memberikan akurasi 5 meter.

Gambar 2.24 Sampel-Ruang Terbuka

Dengan pengaturan seperti itu, area seluas 10.000 meter persegi dapat ditutup oleh 121 Beacon (11 x 11) atau 36 Beacon (6 x 6) tergantung pada keakuratan yang diperlukan.

Namun, jika lokasi terlalu besar dan membutuhkan ribuan Beacon untuk dipasang yang mungkin tidak efektif secara biaya, cari lokasi terbaik seperti POI, tempat masuk dan keluar.

2.6.1.1 Penempatan Beacon - Saran Umum

Jika area yang akan dicakup tidak simetris atau perlu diberi perhatian khusus seperti area resepsionis atau belokan koridor di mana lokasi buruk, Beacon tambahan dapat ditambahkan untuk meningkatkan jangkauan mereka.

Untuk area kurang dari 40 meter persegi, saran kami adalah menggunakan tidak lebih dari satu Beacon.

Untuk kamar kecil di mana Beacon diperlukan di dalam, letakkan sejauh mungkin dari pintu masuk sehingga tidak dapat didengar di luar ruangan. Anda juga dapat memilih untuk menurunkan kekuatan Tx dari Beacon tertentu.

Saat berganti level, disarankan satu Beacon ditempatkan di dekat awal tangga, eskalator, atau lift, dan satu lagi di dekat ujungnya. Ini akan memberi Anda akurasi terbaik dan membuat transisi lantai lebih cepat dalam aplikasi. Jika tangga terlalu panjang, mungkin perlu Beacon lagi di tengah.

2.6.1.1 Alat Pengaturan Cepat - Pendahuluan dan Instalasi

GoIndoor SDK membutuhkan informasi beacon yang ditempatkan di gedung Anda untuk memberikan Anda informasi penempatan di dalam ruangan yang bermanfaat. Ini dapat dilakukan dengan mudah dengan bantuan alat 'pengaturan cepat' gratis dan mudah kami gunakan untuk memasang dan mengelola Beacon di gedung Anda.

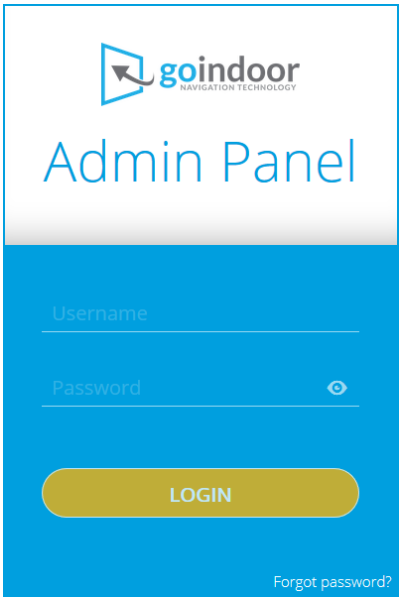
Quick Set Up Tool adalah aplikasi seluler untuk memudahkan pemasangan beacon di lokasi Anda. Ini tersedia untuk perangkat android saat ini. Ini mendukung protokol sue ibeacon dan eddystone.

'Alat Pengaturan Cepat' memerlukan izin berikut untuk menjalankan perangkat.

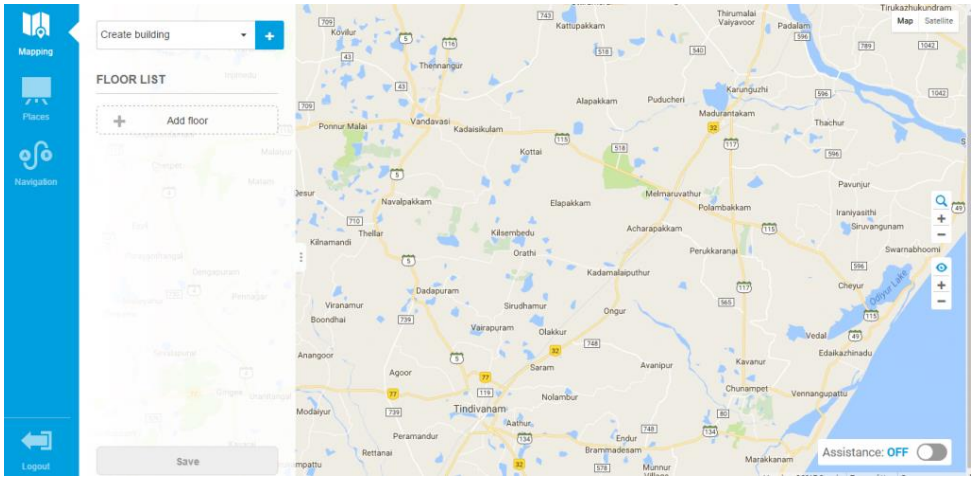
- Lokasi
- Penyimpanan
- Akses jaringan
- Akses Bluetooth

Cara kerja dan penggunaan GoIndoor API terbilang cukup mudah, IDE yang digunakan oleh peneliti adalah android studio, maka *activity* atau template dari GoIndoor API ini sudah tersedia dan dapat langsung digunakan. Untuk lebih jelasnya, berikut adalah tahapan penggunaan GoIndoor API :

1. Daftar atau buat akun terlebih dahulu pada website https://www.goindoor.co/
2. Setelah daftar, buka tab 'Developer' dan buka 'admin panel dan lakukan Log in. Seperti pada Gambar 2.24.

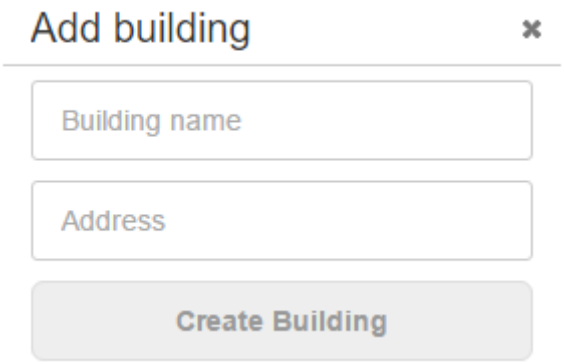


Gambar 2.25 Tampilan Login Admin Panel GoIndoor API



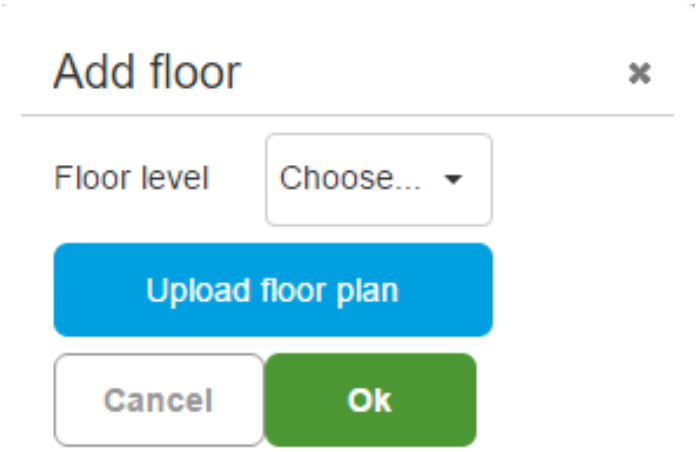
Gambar 2.26 Halaman utama Admin Panel GoIndoor API

3. Pada halaman utama Admin Panel. Lakukan Create Building untuk pembuatan gedung dan juga alamat gedung.



Gambar 2.27 Tampilan pembuatan gedung dan alamat.

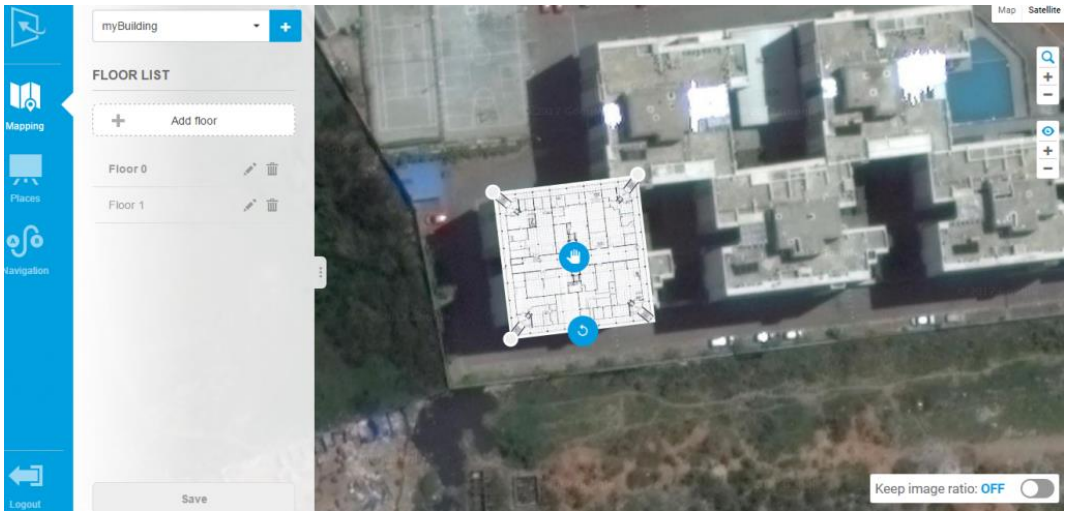
4. Pilih menu Mapping. Pilih gedung yang sudah dibuat sebelumnya, lalu upload denah dari gedung tersebut dan identifikasi ada dilantai berapa denah tersebut.



Gambar 2.28 Tampilan penambahan gambar denah dan identifikasi lantai

Gambar 2.29 Tampilan denah gambar dan identifikasi lantai sudah diisi

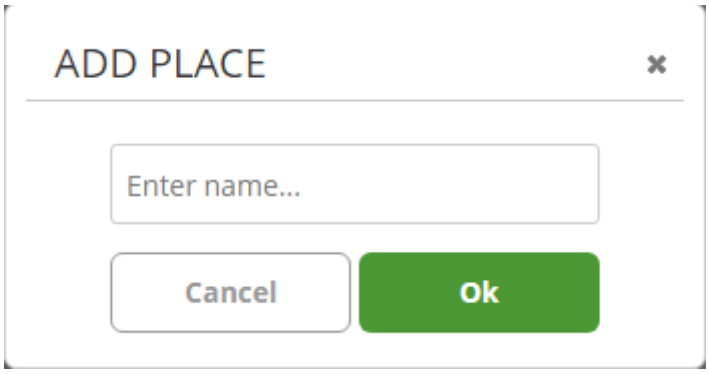
5. Sesuaikan denah pada lokasi gedung yang asli.



Gambar 2.30 Peletakan denah pada lokasi gedung yang asli

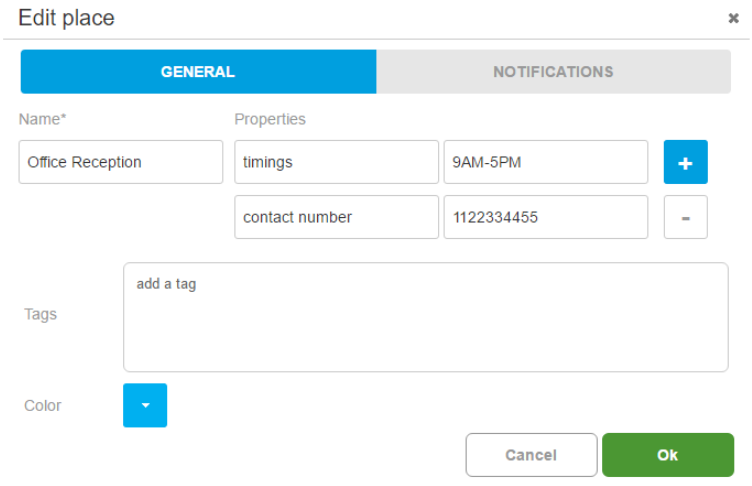
6. Lakukan peletakan tempat atau ruangan pada denah, sesuaikan dengan ruangan pada gedung yang asli. Terdapat 3 bentuk simbol untuk menentukan peletakan ruangan pada denah, yaitu :
   a.  POI ,yaitu untuk menentukan 1 titik pada denah.
   b.  Circular, yaitu untuk menentukan kewilayahan.
   c.  Polygon, yaitu untuk menggambar ruangan sesuai dengan bentuk denah.

   Tentukan nama ruangan yang sudah ditentukan menggunakan simbol. Seperti pada Gambar 2.30.



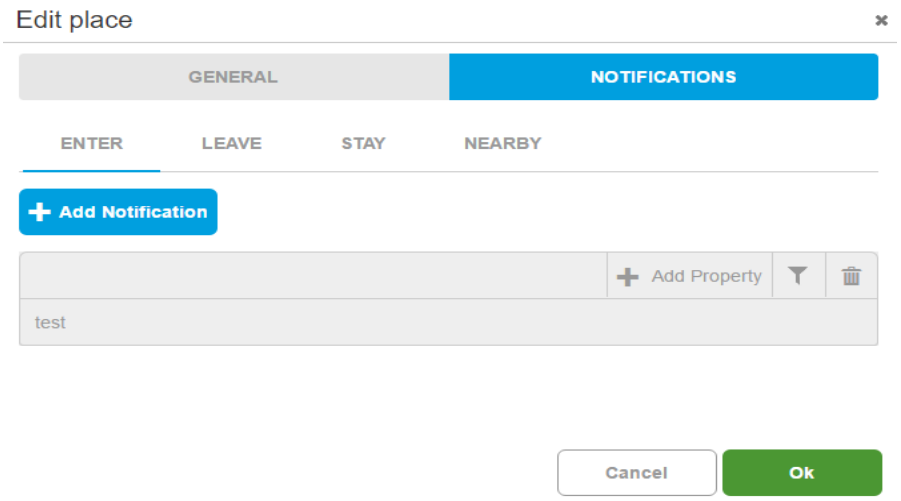
Gambar 3.31 Penentuan nama ruangan atau tempat

7. Penambahan *properties* pada suatu lokasi, misalnya office Reception hanya buka pada pukul 9am-5pm dan juga menentukan nomor kontak.



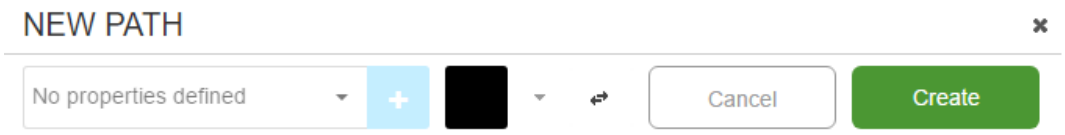
Gambar 3.32 Penentuan *properties*

8. Penambahan *Properties* berguna juga untuk memberi kustom notifikasi kepada pegguna, seperti ketika masuk atau keluar ruangan, berada di dalam ruangan, atau sudah dekat dengan ruangan yang dikunjungi. Seperti pada Gambar 2.32.

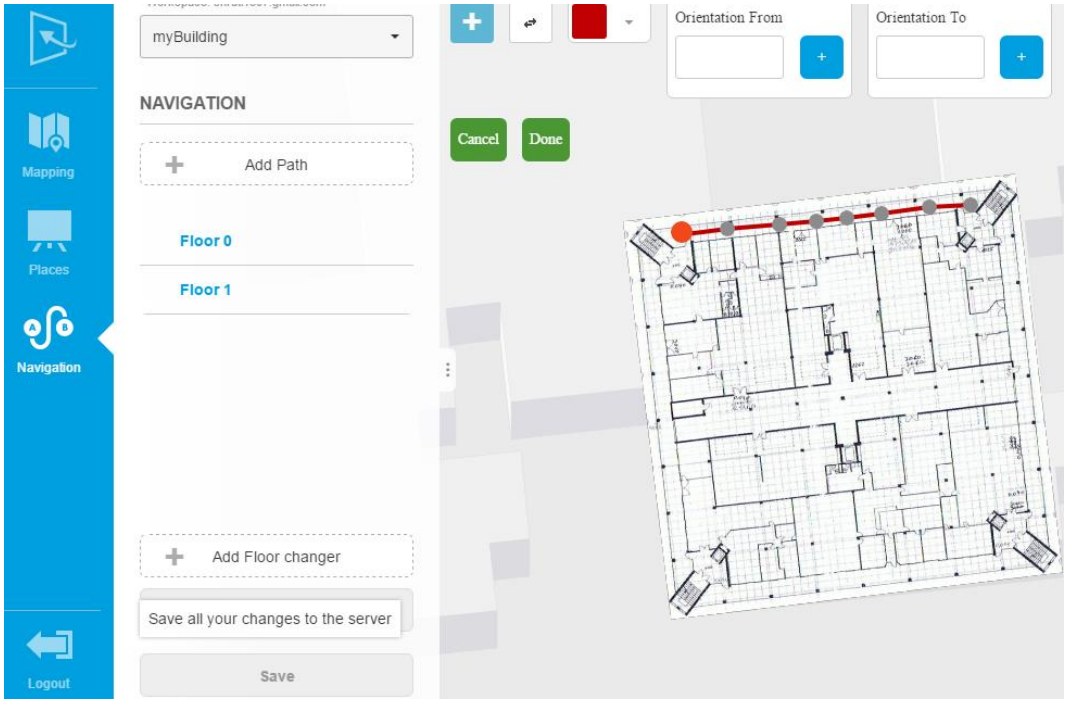


Gambar 2.33 Pemberian notifikasi

9. Pemberian navigasi, pilih menu Navigation, lalu pilih denah pada lantai yang ingin diberikan navigasi, tentukan warna untuk navigasi dan tentukan apakah navigasi hanya 1 arah atau 2 arah dengan menekan simbol ⇄ .

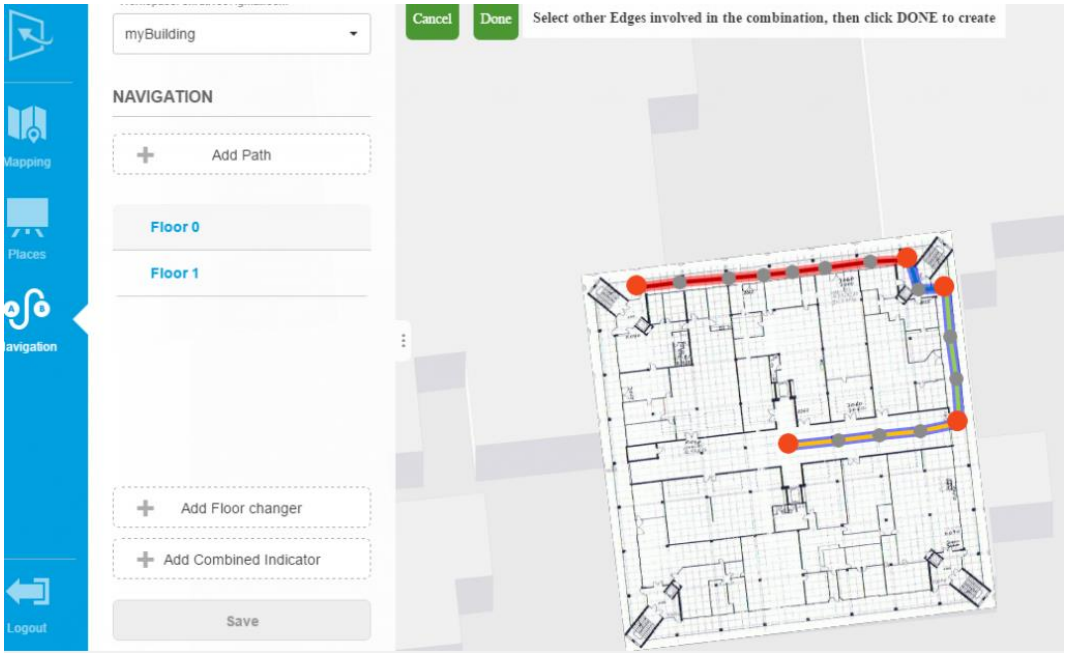


Gambar 2.34 Penentuan warna dan arah navigasi



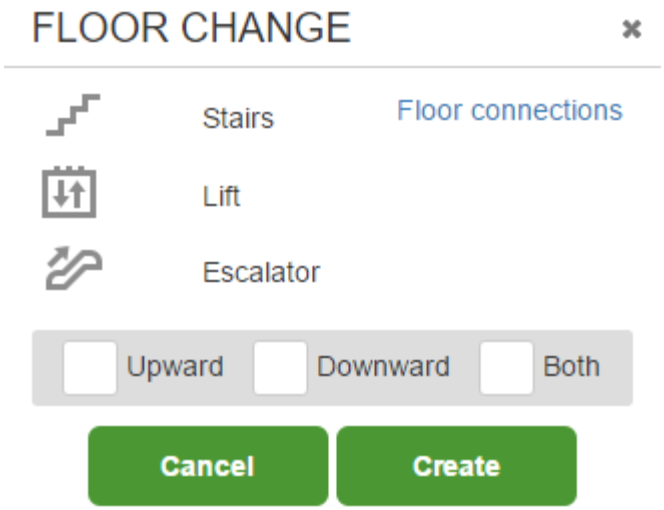
Gambar 2.35 Denah yang sudah diberi navigasi

Pemberian navigasi bisa lebih dari satu dan di dapat kombinasikan dengan navigasi yang lain dilantai yang sama.



Gambar 2.36 Kombinasi navigasi pada denah

10. Penentuan perpindahan lantai, pada menu Navigation terdapat menu ‘Add Floor Changer’ untuk menentukan perpindahan lantai dan menentukan arah hanya ke atas, kebawah atau keduanya. Terdapat tiga bentuk perpindahan lantai, yaitu tangga, *lift*, dan *escalator*.



Gambar 2.37 Penentuan bentuk perpindahan lantai

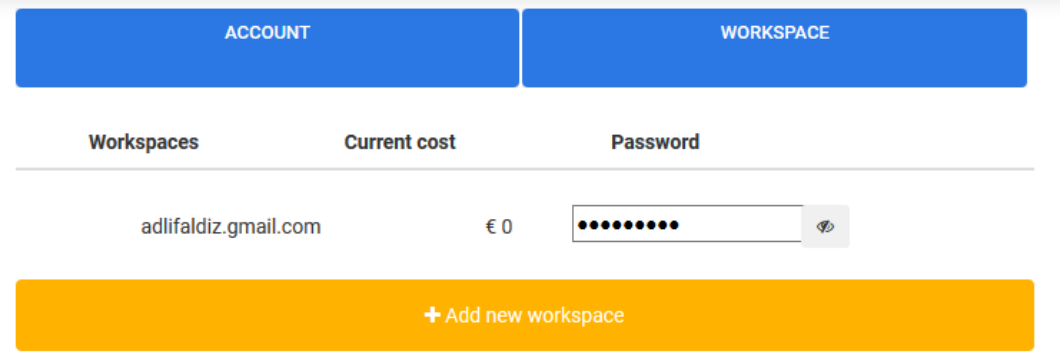
Letakkan simbol pada perpindahan lantai pada denah. Lalu kombinasikan dengan navigasi yang mengarah pada simbol perpindahan. Lakukan pada denah pada lantai yang berbeda.

11. Penempatan Beacon pada gedung yang asli, penempatan beacon harus ditempat tidak terhalang oleh objek agar sinyal yg dipancarkan dapat terjangkau oleh smartphone. Penempatan yang bagus yaitu dipasang di ceiling gedung.

12. Penempatan Beacon pada denah yang ditentukan, penempatan Beacon pada denah harus sesuai dengan penempatan beacon pada gedung asli menggunakan aplikasi yang disediakan oleh GoIndoor.

Gambar 2.38 Peletakkan beacon pada denah

13. Memanggil GoIndoor API, yaitu menggunakan android veri 2.+ atau lebih tinggi dengan android SDK Platform versi 4.4 (Kitkat) API level 19 atau lebih tinggi.
14. Dapatkan akses id dan *password* pada website GoIndoor pada akun *workspace.*



Gambar 2.39 Akses id dan *password*

15. Menambahkan library GoIndoor pada build.gradle

Tabel 2.4 Menambahkan library GoIndoor

```
dependencies{
    compile 'com.oym.indoor:goindoor:2.4.1'
}
```

16. Unduh Goindoor Goindoor-2.4.1.aar. Import file ke library project dan tambahkan library pada build.gradle.

Tabel 2.5 Menambahkan file Goindoor-2.4.1.aar.

```
dependencies {
    compile(name:'goindoor-2.4.1', ext:'aar')
}
```

17. Menambahkan akses fitur bluetooth, internet, akses wifi, lokasi, dan getar pada smartphone.

Tabel 2.6 Menambahkan akses fitur pada smartphone

```
<!-- Bluetooth -->
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<!-- Wifi -->
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"
/>
<!-- Maps -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
<!-- Notification -->
<uses-permission android:name="android.permission.VIBRATE" />
```

18. Menyambungkan aplikasi dengan server GoIndoor.

Tabel 2.7 Menyambungkan dengan server GoIndoor

```
/*
Create a GoIndoor object .
*/
go = new GoIndoor.Builder()
        .setContext(context)
        .setAccount(account)
        .setPassword(password)
        .setConnectCallback(callbackConnect )
        .build();
```

19. Sinkronisasi dengan server.

Tabel 2.8 Sinkronisasi dengan server GoIndoor

```
private ConnectCallback callbackConnect = new
ConnectCallback() {
   @Override
   //when the connection to the server has been correctly
established and the library is initialized
   public void onConnected() {
      Log.i(TAG, "Connected");
   }

   @Override
   //when an exception is thrown when trying to connect to
the server
   public void onConnectFailure() {
      Log.i(TAG, "Connection Failed");
   }
```

20. Sinkronisasi lokasi servis.

Tabel 2.9 Sinkronisasi lokasi servis

```
private LocationBroadcast broadcast = new
LocationBroadcast() {
    @Override
    public void onLocation(LocationResult location) {
    //Process location updates
    }

    @Override
    public void onNotification(NotificationResult
notification) {
    // Process notifications
    }
};
```

21. POI posisi pengguna terkini.

Tabel 3.10 Posisi pengguna terkini

```
private void drawUserPosition(LatLng point, double
accuracy) {
            if (isMapUpdated && markerBitmap != null) {
                  isMapUpdated = false;
                  // Draw user point and circle
                  drawAnimatedMarker(point, accuracy);
                  // Update camera
                  if (isCameraUpdated) {
                        animateCamera(point,
MAP_TIME_MARKER, map.getCameraPosition().zoom);
                  }
            }
      }
```

22. Notifikasi pada pengguna

Tabel 3.11 Notifikasi Pengguna

```
private LocationBroadcast indoorLocation = new
LocationBroadcast() {
      @Override
      public void onLocation(LocationResult location) {

       }

      @Override
      public void onNotification(NotificationResult nr) {

            switch (nr.notification.action) {
              case ENTER:
                    showMessage("Welcome to "+
nr.place.getName());
                     break;
              case STAY:
                    showMessage("Would you like to know
more about "+ nr.place.getName());
                     break;
              case LEAVE:
                    showMessage("Thank you for visiting" +
nr.place.getName());
                     break;

               case NEARBY:
                     showMessage("We are waiting for you
at" + nr.place.getName());
                     break;

               default:
                     return;

             }
        }
};
```

23. Navigasi dari titik awal ke titik akhir.

Tabel 2.12 Navigasi

```
@Override
protected Boolean doInBackground(Void... params) {
      try {
            synchronized (mutexPosition) {
                  if (currentPosition.type !=
LocationResult.TYPE_BEACON) {
      synchronized (mutex) {
            gs.addLocationCallback(this);
                              mutex.wait();
                        }
                  }
RoutePoint rp = new RoutePoint(currentPosition.longitude,
```

```
currentPosition.latitude, currentPosition.floorNumber,
currentPosition.building);
Route r = gs.getGoIndoor().computeRoute(rp, point);
if (r != null
&& !r.getProjection(currentPosition).isRecomputeRequired)
{
      gs.setRoute(r);
      gs.getGoIndoor().getLogger().logRoute(r);
                                    return true;
                  }
            }
      } catch (Exception e) {
            e.printStackTrace();
      }
return false;
}
```

24. Menampilkan tempat atau ruangan

Tabel 2.13 Menampilkan tempat atau ruangan

```
// Get building
            if (loc.type == LocationResult.TYPE_BEACON
                        && (building == null || (building
!= null && !building.getId().equals(loc.building) &&
isBuildingReady))) {
                  building =
gs.getGoIndoor().getBuildings(loc.building);
                  if (building != null) {
                        isBuildingReady = true;

      fubLocal.setText(building.getName());

                        // Check areas
                        drawerAreas = new ArrayList<>();
                        for (Place place :
gs.getGoIndoor().getPlaces(building.getId())) {
                              // FIXME
                              CustomSingleItem csi = new
CustomSingleItem(place.getName(),
                                          new
RoutePoint(place.getX(), place.getY(),
place.getFloorNumber(), place.getBuilding()));
                              drawerAreas.add(csi);
                        }
                        populateDrawer();
                  }
            }
```

25. *Tracking* lokasi pengguna

Tabel 2.14 *Tracking* lokasi pengguna

```
public void onLocation(LocationResult location) {
       go.getLogger().logPosition(loc);
}
```

2.6.2 Google Maps API

Google Maps API adalah salah satu *Application Programming Interface* (API) yang dimiliki Google. API ini mempunyai fitur untuk melakukan aktivitas-aktivias yang berkaitan dengan Google Maps, antara lain menampilkan peta, mencari rute terdekat antara dua tempat, dan lain sebagainya. Google Maps API tersedia untuk platform *android, iOS, web*, dan *juga web service*. Google maps API juga menyediakan layanan seperti direction, geocoding, distance matrix API, dan elevation API.

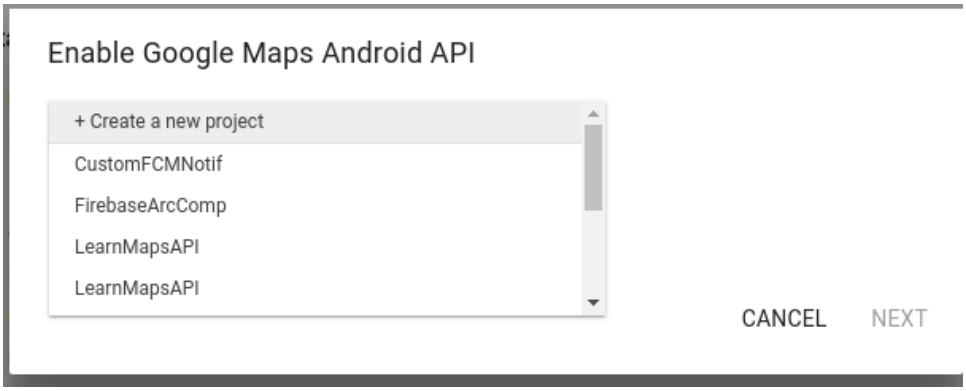
Google maps adalah sebuah layanan gratis yang diberikan oleh Google yang menampilkan suatu peta dunia yang dapat digunakan untuk melihat suatu daerah dengan menggunakan suatu web browser. Kita dapat menambahkan fitur google maps di dalam suatu sistem atau aplikasi yang kita sedang kita bangun. Google maps API adalah suatu library dengan bentuk javascript yang dapat diubah maupun menambahkan variabel-variabel sesuai dengan keinginan kita. *Java Interpreter* akan melakukan konversi berkas dari *bytecode* menjadi bahasa mesin yang sesuai dengan jenis dan platform yang digunakan pada aplikasi.[14]

Dalam perkembangannya google maps API diberikan kemampuan untuk mengambil peta statis. Melakukan geocoding dan memberikan penuntun arah. Kekurangan pada google maps API yaitu jika ingin melakukan akses harus terdapat layanan internet pada perangkat yang digunakan, sedangkan kelebihan yang dimiliki yaitu:

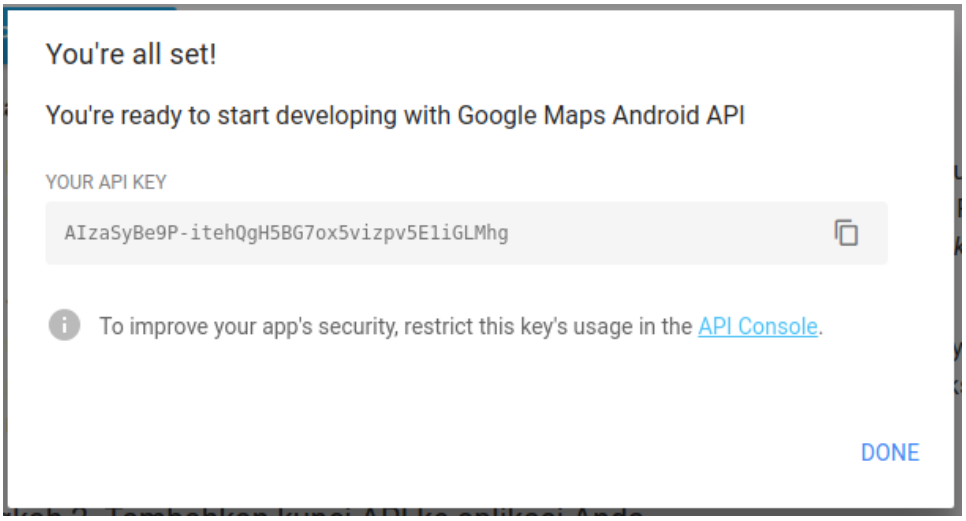
1. Dukungan penuh yang dilakukan google sehingga terjamin dan fitur yang bervariasi pada google maps API.
2. Banyak pengembang yang menggunakan google maps API sehingga mudah dalam mencari referensi dalam pengembangan aplikasi.

Cara kerja dan penggunaan Google Maps API pada aplikasi yang akan dibangun untuk menampilkan peta agar pengguna mengetahui keberadaan atau lokasi pengguna terkini:

1. Dapatkan Google API key pada Google Cloud Platform Console https://developers.google.com/maps/documentation/android-api/signup?hl=ID.
2. Pilih "Get API key" lalu pilih "+ Create a new project", beri nama project yang akan dibangun.

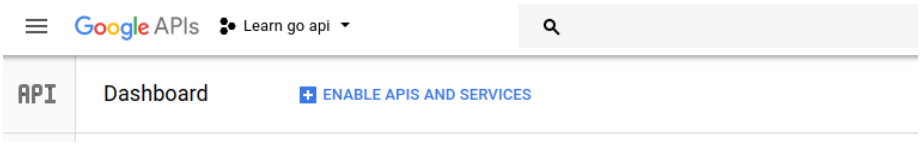


Gambar 3.40 Tampilan pembuatan project

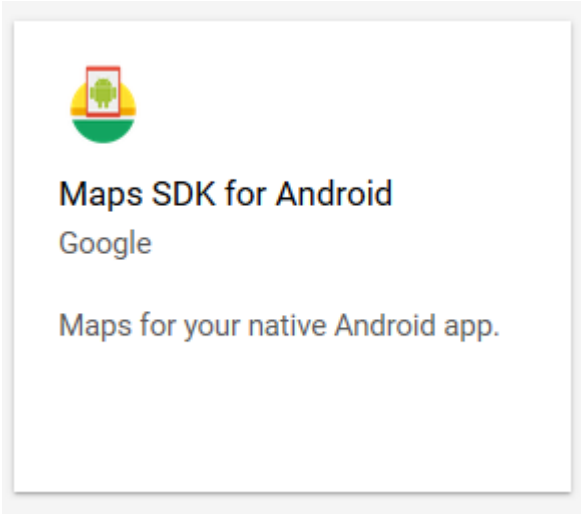


Gambar 3.41 API key yang sudah didapatkan

3. Masuk pada Google Cloud Platform https://console.developers.google.com, pilih "enable APIS and services" lalu "view all" dan aktifkan layanan "Maps SDK for Android"



Gambar 3.42 Mengaktifkan layanan API



Gambar 3.43 Layanan Google Maps untuk Android

4. Tambahkan permission pada AndroidManifest.xml untuk dapat mengakses fitur lokasi pada smartphone.

Tabel 3.15 Izin lokasi

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

5. Tambahkan meta tag untuk Google Maps API key yang sudah didapatkan.

Tabel 3.16 Penambahan meta tag untuk API key

```
<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="YOU_KEY"/>
```

6. Panggil Google Maps API pada aplikasi.

Tabel 3.17 Memanggil Google Maps API

```
@Override
public void onMapReady(GoogleMap googleMap) {
            map = googleMap;
            if (map != null) {
```

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                map.setIndoorEnabled(false);
                map.setBuildingsEnabled(false);
                map.getUiSettings().setZoomControlsEnabled(false);
```

2.7 Objek Oriented Analysis Design

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem.[13]

2.8 Unified Modelling Language

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

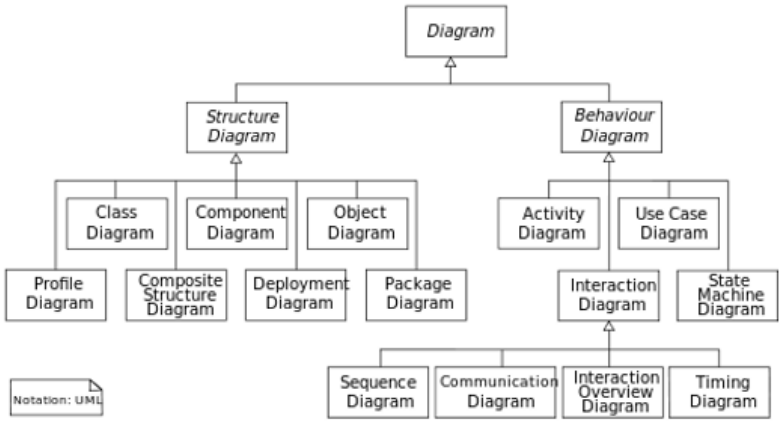
Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh

OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.

2.8.1 Diagram UML

Berikut penjelasan singkat dari pembagian kategori tersebut adalah:



Gambar 2.44 Diagram UML

- *Structure* Diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari suatu sistem yang dimodelkan
- Behavior Diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
- Interaction diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interkasi antar subsistem pada suatu sistem.

2.8.2 *Class Diagram*

Diagram kelas atau *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas - kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas sebagai berikut:

- Kelas main yaitu kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
- Kelas yang menangani tampilan sistem (*view*) yaitu kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
- Controller yaitu kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasa disebut kelas proses.
- Model yaitu kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Berikut adalah simbol-simbol yang ada pada diagram kelas :

Tabel 2.18 Simbol *Class Diagram*

Simbol	Deksripsi
Kelas **nama_kelas** +atribut +operasi()	Kelas pada struktur sistem.
Antarmuka / *Interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
Asosiasi / *Association* 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *Directed association* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum – khusus)
Kebergantungan / *Dependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas.
Agregasi / *Aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

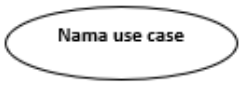
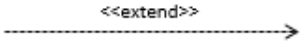
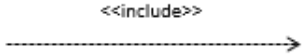
2.8.3 *Use Case Diagram*

Diagram *Use Case* merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *Use Case* digunakan untuk mengetahui fingsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Ada dua hal utama pada *Use Case* yaitu pendefinisian apa yang disebut aktor dan *Use Case*.[13]

- Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu orang.
- *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

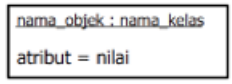
Tabel 2.19 Simbol *Use Case Diagram*

Simbol	Deksripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor, biasanya dinyatakan dengan menggunakan kata kerja di awal di frase nama usecase
Aktor / Actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase aktor.
Asosiasi / Association 	Komunikasi antar aktor dan use case yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi / Extend 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
Generalisasi / Generalization 	Hubungan generalisasi dan spesialisasi (umum- khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan / Include / Uses 	Relasi *use case* tamabahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

2.8.4 Objek Diagram

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Berikut adalah simbol-simbol yang ada pada diagram objek:[13]

Tabel 2.20 Simbol Diagram Objek

Simbol	Deskripsi
Objek 	objek dari kelas yang berjalan saat sistem dijalankan
Link 	relasi antar objek

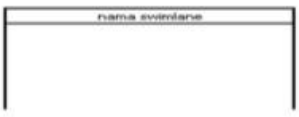
2.8.5 *Activity Diagram*

Diagram aktivitas asatu *Activity Diagram* menggambarkan workflow (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktifitas menggambarkan aktifitas bukan apa yang dilakukan aktor.[13] Diagram aktifitas juga banyak digunakan untuk mendefinisikan hal-hak berikut:

- Rancangan proses bisnis dimana setiap urutan aktifitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
- Urutan atau pengelompokan tampilan dari sistem / user interface dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan
- Rancangan pengujian dimana setiap aktifitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
- Rancangan menu yang ditampilkan pada diagram aktifitas

Berikut adalah simbol-simbol yang ada pada diagram aktifitas:

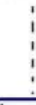
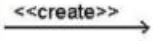
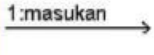
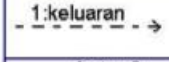
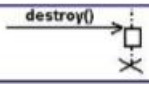
Tabel 2.21 Simbol *Activity Diagram*

Simbol	Deskripsi
status awal 	status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
aktivitas 	aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
percabangan / *decision* 	asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
penggabungan / *join*	asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
status akhir 	status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
swimlane  atau 	memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

2.8.6 *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *Use Case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirim dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objekobjek yang terlibat dalam sebuah *Use Case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.[13] Berikut adalah simbol-simbol yang ada pada *Sequence Diagram*:

Tabel 2.22 Simbol *Sequence Diagram*

aktor  	• orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari system. • Berpartisipasi secara berurutan dengan mengirimkan dan / atau menerima pesan. • Ditempatkan di bagian atas diagram.
objek  	**Sebuah objek:** • **Berpartisipasi secara berurutan dengan mengirimkan dan / atau menerima pesan.** • **Ditempatkan di bagian atas diagram.**
Garis hidup objek 	• Menandakan kehidupan obyek selama urutan. • diakhiri tanda X pada titik di mana kelas tidak lagi berinteraksi.
Objek sedang aktif berinteraksi 	**Fokus kontrol:** • **Adalah persegi panjang yang sempit panjang ditempatkan di atas sebuah garis hidup.** • **Menandakan ketika suatu objek mengirim atau menerima pesan.**
pesan  	objek mengirim satu pesan ke objek lainya
 	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
 	menyatakan bahwa suatu objek mengirimkan masukan ke objek lainnya arah panah mengarah pada objek yang dikirimi
 	**objek/metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian**
 	**menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy**