# BAB II

# LANDASAN TEORI

## 2.1. *Tenant* Apartemen

*Tenant* artinya penyewa atau pihak yang menyewa suatu bagian dari *property* kepada suatu lembaga, institusi, organiasasi atau kepemilikan. Kata *tenant* awal mulanya berasal dari istilah "*tenure*" di Inggris kuno. *Tenure* adalah "*landlord*" atau tuan tanah yang menguasai dan mengelola lahan-lahan untuk keperluan-keperluan pribadinya ataupun keperluan bisnisnya. Jadi *tenant* apartemen yaitu pihak yang meyewa properti (dalam hal ini *property* apartemen) untuk kebutuhan sehari-hari nya.

## 2.2. *Building Management*

*Building Management* adalah bentuk pemeliharaan gedung agar gedung dapat berfungsi dengan baik dan memiliki performansi penuh untuk pengguna gedung (*tenant*). Merupakan poin penting bagi *Building Management* untuk memastikan semua yang berkaitan dengan gedung, mulai dari fasilitas, kebersihan, keamanan dan *tenant relation* berjalan dengan baik dan benar sesuai prosedur.

## 2.3. Android

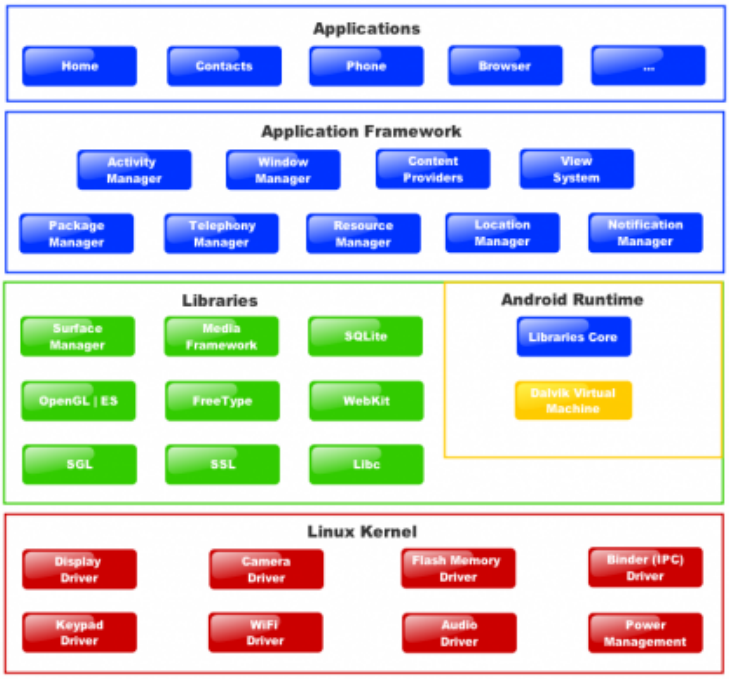
Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GSM) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD). (Efmi Maiyana, 2018)


**Gambar 2.1. Logo Android**

Pada dasarnya saat ini kebanyakan vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis Android. Hal ini karena android itu adalah sistem operasi yang *open source* sehingga bebas di distribusikan dan dipakai oleh *vendor* manapun. Pesatnya pertumbuhan android selain faktor yang disebutkan sebelumnya adalah karena android itu sendiri adalah *platform* yang sangat lengkap baik sistem operasinya, aplikasi dan *Tool* Pengembangan, *Market* aplikasi android serta dukungan yang sangat tinggi dari komunitas *open source* dunia. (Efmi Maiyana, 2018)

## 2.4. Struktur Arsitektur Android

Android memungkinkan penggunanya untuk memasang aplikasi pihak ketiga, baik yang diperoleh dari toko aplikasi seperti Google Play, Amazon App store, ataupun dengan mengunduh dan memasang berkas APK dari situs pihak ketiga. Di Google Play, pengguna bisa menjelajah, mengunduh, dan memperbarui aplikasi yang diterbitkan oleh Google dan pengembang pihak ketiga, sesuai dengan persyaratan kompatibilitas Google. Google Play akan menyaring daftar aplikasi yang tersedia berdasarkan kompatibilitasnya dengan perangkat pengguna.




**Gambar 2.2. Struktur Arsitektur Android**

Android dibangun dengan menggunakan asal *object oriented*, dimana elemen-elemen penyusun sistem operasinya berupa objek yang dapat kita gunakan kembali/*reusable*. Agar bisa membuat aplikasi dengan baik, tentunya kita harus mengetahui arsitektur OS Android beserta elemennya. (Julio Anthony, 2015)

### 2.4.1. *Layer Application*

Aplikasi berada pada lapisan terluar dari Arsitektur Android. Pengguna awam Android pasti akan berinteraksi dengan lapisan ini untuk fungsi umum seperti menelepon, mengakses website, dan lain-lain. Lapisan di bawah dari lapisan aplikasi ini diakses kebanyakan oleh *Developer, Programmer* atau sejenisnya. Beberapa aplikasi standar yang pasti ada pada setiap perangkat, seperti:

- SMS (*Short Message Service*)
- *Web Browser*
- *Contact Manager*

### 2.4.2. *Layer Application Framework*

Lapisan ini berinteraksi langsung dengan aplikasi kita. Program-program di atas memanajemen fungsi dasar dari perangkat seperti manajemen *Resource*, Manajemen Panggilan, Manajemen *Window* dan lain-lain. Sebagai seorang *developer*, kita dapat melihat lapisan ini sebagai alat dasar yang dapat digunakan untuk mengembangkan aplikasi. Beberapa program penting pada *Application Framework* antara lain:

- *Activity Manager*
- *Content Providers*
- *Resource Manager*
- *Notification Manager*

### 2.4.3. *Android Runtime*

Terletak pada *level* yang sama dengan lapisan *Library* juga terdapat Lapisan Android Runtime dan juga sekumpulan *Library* Java yang dikhususkan untuk Android. *Programmer* Aplikasi Android membuat aplikasinya menggunakan bahasa pemrograman Java. Dalam lapisan *Android Runtime* juga terdapat Dalvik VM (*Virtual Machine*). (Julio Anthony, 2015)

### 2.4.4. *Android Libraries*

Kategori ini menyangkut *Library* berbasis Java yang berfungsi khusus untuk pengembangan Android. Contoh dari *Library* yang termasuk dalam kategori ini adalah Library yang memfasilitasi pembangunan *User Interface*, Penggambaran Grafik dan akses *Database*, juga *library* yang terdapat pada *Application Framework*. Beberapa contoh library android yang tersedia yang dapat digunakan oleh *developer* yaitu:

- Android.app
- Android.content
- Android.database
- Android.widget

### 2.4.5. *Library*

*Library* membawa sekumpulan instruksi untuk mengarahkan perangkat Android kita dalam menangani berbagai tipe data. Contohnya,perekam dari berbagai macam format Video dan Audio ditangani oleh *Media Framework Library*. Berikut adalah beberapa kegunaan *Library*:

- *Surface Manager*: Mengolah tampilan Windows Pada Layar
- SGL: Grafik 2 Dimensi
- *Media Framework*: Menunjang perekaman dari berbagai macam format audio, video, dan gambar
- *Free Type*: Penerjemah Font

### 2.4.6. *Linux Kernel*

Di lapisan terbawah Arsitektur Android terdapat Linux Kernel. Lapisan ini tidak benar benar berinteraksi dengan pengguna maupun *developer*, tapi lapisan ini merupakan jantung dari seluruh sistem di Android karena lapisan inilah yang memberikan fungsi-fungsi berikut pada sistem Android:

- Abstraksi *Hardware*
- Program Manajemen *Memory*
- Pengaturan Sekuritas
- Manajemen Energi *Software* ( Baterai )

## 2.5. SDLC (*Systems Development Life Cycle*)

*System Development Life Cycle* (SDLC) adalah keseluruhan proses dalam membangun sistem melalui beberapa langkah. Ada beberapa model SDLC. Model yang cukup populer dan banyak digunakan adalah *waterfall*. Beberapa model lain SDLC misalnya *fountain*, *spiral*, *rapid application development*, prototyping, *incremental*, *build & fix*, dan *synchronize & stabilize*. Dengan siklus SDLC, proses membangun sistem dibagi menjadi beberapa siklus SDLC, prosesnya adalah sebagai berikut :

- Analisis sistem
- Spesifikasi kebutuhan sistem
- Perancangan sistem
- Pengembangan sistem,
- Pengujian sistem
- Implementasi dan pemeliharaan sistem

SDLC sendiri terbagi menjadi beberapa model dan salah satunya adalah RAD (*Rapid Application Development*) yang nantinya akan membantu dalam perancangan aplikasi mobile agar dapat di mengerti oleh peneliti lain untuk dapat di kembangkan lebih lanjut. (Isnardi, 2016)

### 2.5.1. RAD (*Rapid Application Development*)

*Rapid Aplication Development* (RAD) adalah sebuah proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan dalam waktu yang singkat. RAD menggunakan metode iteratif (berulang) dalam mengembangkan sistem dimana *working* model (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan (*requirement*) pengguna dan selanjutnya disingkirkan. Dalam pengembangan sistem informasi normal, memerlukan waktu minimal 180 hari, namun dengan menggunakan metode RAD, sistem dapat diselesaikan dalam waktu 30-90 hari. (Safrian Aswati, 2016)


**Gambar 2.3. *Rapid Application Development***

Model RAD memiliki 3 tahapan sebagai berikut.

1. Rencana Kebutuhan (*Requirement Planning*): *User* dan *analyst* melakukan pertemuan untuk mengidentifikasi tujuan dari sistem dan kebutuhan

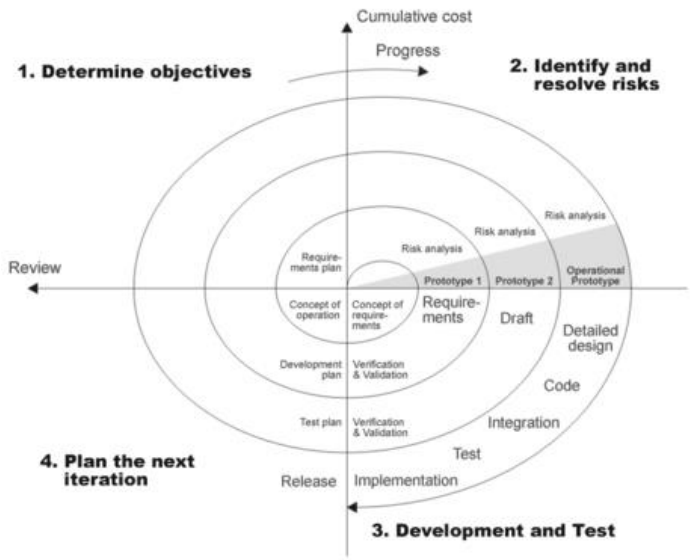
informasi untuk mencapai tujuan. Pada tahap ini merupakan hal terpenting yaitu adanya keterlibatan dari kedua belah pihak.

2. Proses Desain Sistem (*Design System*): Pada tahap ini keaktifan *user* yang terlibat menentukan untuk mencapai tujuan karena pada proses ini melakukan proses desain dan melakukan perbaikan-perbaikan apabila masih terdapat ketidaksesuaian desain antara *user* dan analyst. Seorang *user* dapat langsung memberikan komentar apabila terdapat ketidaksesuaian pada desain, merancang sistem dengan mengacu pada dokumentasi kebutuhan *user* yang dibuat pada tahap sebelumnya. Keluaran dari tahapan ini adalah spesifikasi *software* yang meliputi organisasi sistem secara umum, struktur data dan yang lain.

3. Implementasi (*Implementation*): Tahapan ini adalahan tahapan programmer yang mengembangkan desain suatu program yang telah disetujui oleh *user* dan analyst. Sebelum diaplikasikan pada suatu organisasi terlebih dahulu dilakukan proses pengujian terhadap program tersebut apakah ada kesalahan atau tidak. Pada tahap ini *user* biasa memberikan tanggapan akan sistem yang sudah dibuat serta mendapat persetujuan mengenai sistem tersebut.

2.5.2. *Spiral*

Model *Spiral* adalah salah satu metode yang dapat digunakan dalam pengembangan perangkat lunak. Model *spiral* merupakan penggabungan dari model *prototyping* dan model *waterfall*. Model *prototyping* yang fokus pada penyajian atau presentasi kepada *user* dengan format input dan output kemudian perangkat lunak akan dievaluasi. Model *waterfall* yang fokus kepada proses pengembangan perangkat lunak yang sistematis atau berurutan. Model *spiral* menekankan pada Analisa resiko setiap tahapannya. Fungsi model *spiral* adalah untuk melakukan perubahan, penambahan dan pengembangan perangkat lunak dengan memaksimalkan aspek kecepatan dan ketepatan berdasarkan keinginan dan kebutuhan penggunanya. (Herlinda Kusmiati, 2015)




**Gambar 2.4. *Spiral***

### 2.5.3. *Prototyping*

Metode *Prototype* merupakan satu metode dalam pengembangan perangkat lunak, metode ini merupakan suatu paradigma baru dalam pembuatan atau pengembangan perangkat lunak. Metode ini adalah evolusi dalam dunia pengembangan atau pembuatan perangkat lunak, metode ini juga merevolusi metode pengembangan atau pembuatan perangkat lunak yang lama, yaitu sistem sekuensial yang biasa dikenal dengan nama metode *waterfall*. Dalam metode *prototype/prototyping*, perangkat lunak yang dihasilkan kemudian dipresentikan kepada klien, dan klien tersebut diberikan kesempatan untuk memberikan masukan dan kritikan, sehingga *software* yang dihasilkan sesuai dengan kebutuhan dan keinginan pelanggan. Perubahan perangkat lunak dapat dilakukan berkali-kali hingga dicapai kesepakatan bentuk dari *software* yang akan dikembangkan. (Dwi Purnomo, 2017)

## 2.6. MySQL

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengolahan datanya.

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pembangun aplikasi *web* yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis *web*, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman script PHP. (Astria Firman, 2016)

## 2.7. PHP

PHP adalah bahasa pemrogram *script* yang paling banyak dipakai saat ini. PHP banyak dipakai untuk memrogram situs *web* dinamis, walaupun tidak tertutup kemungkinan untuk digunakan untuk pemakaian lain. PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP bernama FI (F*orm Interpreted*). Pada saat tersebut PHP adalah sekumpulan *script* yang digunakan untuk mengolah data form dari *web*. Perkembangan selanjutnya adalah Rasmus melepaskan kode sumber tersebut dan menamakannya PHP/ FI, pada saat tersebut kepanjangan dari PHP/ FI adalah *Personal Home Page/Form Interpreter*. PHP Hypertext preposessor adalah merupakan bahasa berbentuk skrip yang ditempatkan dalam server. Hasilnya adalah yang dikirim ke klien, tempat pemakai menggunakan browser. Secara khusus, PHP dirancang untuk membentuk *web* dinamis. Artinya semua sintak yang diberikan akan sepenuhnya dijalankan pada server. Sedangkan yang dikirim ke browser hanya hasilnya saja. (Astria Firman, 2016)

Kode PHP juga bisa berkomunikasi dengan *database* dan melakukan perhitungan-perhitungan yang kompleks. Pada saat ini, PHP cukup popular sebagai peranti pemrograman *web*, terutama di lingkungan *linux*. Walaupun demikian, PHP sebenarnya juga dapat berfungsi pada server-server yang berbasis UNIX, Windows NT, dan Macintosh. PHP bersifat bebas pakai tidak perlu membayar apapun untuk menggunakan perangkat lunak ini. Salah satu kelebihan dari PHP adalah mampu berkomunikasi dengan berbagai *database* yang terkenal. Dengan demikian, menampilkan data yang bersifat dinamis yang diambil dari *database* merupakan hal yang mudah untuk mengimplementasikan. Itulah sebabnya sering dikatakan bahwa

PHP sangat cocok untuk membangun halaman-halaman *web* dinamis. Penemu bahasa pemrograman ini adalah Rasmus Lerdorf yang bermula dari keinginan sederhana ahli tersebut untuk mempunyai alamat batu (*tool*) dalam memonitor pengunjung yang melihat situs *web* pribadinya. (Astria Firman, 2016)

## 2.8. JSON

JSON — singkatan untuk JavaScript Object Notation — adalah sebuah format untuk berbagi data. Seperti dapat kita lihat dari namanya, JSON diturunkan dari bahasa pemrograman javaScript, akan tetapi format ini tersedia bagi banyak bahasa lain termasuk Python, Ruby, PHP, dan Java. JSON biasanya dilafalkan seperti nama "Jason."JSON menggunakan ekstensi .json saat ia berdiri sendiri. Saat didefinisikan di dalam format file lain (seperti di dalam .html), ia dapat tampil didalam tanda petik sebagai JSON string, atau ia dapat dimasukkan kedalam sebuah variabel. Format ini sangat mudah untuk ditransfer antar *server web* dengan klien atau *browser*. (Bhakti Destian, 2016)

Karena sangat mudah dibaca dan ringan, JSON memberikan alternatif lebih baik dari XML dan membutuhkan formatting yang tidak banyak.

Penggunaan JSON:

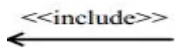
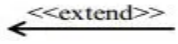
- JSON digunakan saat menulis aplikasi dengan Javascript mencakup esktensi *browser* dan *website*.
- Format JSON digunakan untuk serialisasi dan transmisi data terstruktur melalui jaringan komputer.

- Pada umumnya digunakan untuk transmisi data antara server dan aplikasi web.
- *Web service* dan API menggunakan format JSON untuk menyediakan data untuk publik.

## 2.9. *Use Case Diagram*

*Use Case Diagram* adalah sesuatu atau proses merepresentasikan hal-hal yang dapat dilakukan oleh aktor dalam menyelesaikan sebuah pekerjaan. 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 fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. (Ade Hendini, 2016)

**Tabel 2.1. Komponen pada *Use Case Diagram***

| Simbol | Keterangan |
|---|---|
|  | Aktor: Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case* |
|  | *Use case*: abstraksi dan interaksi antara sistem dan aktor |
|  | *Association*: abstraksi dan penghubung antara aktor dan use case |
|  | *Generalisasi*: menunjukan spesialisasi aktor untuk dapat berpartisipasi dengan *use case* |
|  | *Include*: Menunjukan bahwa suatu *use case* seluruhnya merupkan fungsionalitas dari use case lainnya |
|  | *Extend*: Menunjukan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi |

**Sumber: (Ade Hendini, 2016)**

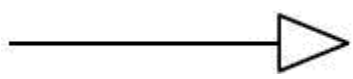
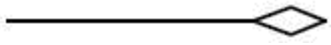
## 2.10. *Class Diagram*

*Class diagram* adalah model statis yang menggambarkan struktur dan deskripsi *class* serta hubungannya antara *class*. *Class* diagram mirip ERD pada perancangan *database*, bedanya pada *class diagram* tidak terdapat operasi/metode tapi hanya atribut. *Class* terdiri dari nama kelas, atribut dan operasi/methode. Berikut ini merupakan Elemen *Class Diagram*

- *Class* (Nama)
- *Attributes*
- *Operations*
- *Relationships*

Dalam *class diagram* terdapat penghubung antara satu *class* dengan *class* lainnya, berikut adalah tabel dari simbol hubungan antara *class* yang digunakan pada *class diagram*. (Ade Hendini, 2016)

**Tabel 2.2. Simbol *Class Diagram***

| Simbol | Keterangan |
|---|---|
|  | *Association*: Relasi antar kelas dengan makna yang umum |
|  | *Direct Association*: relasi antar kelas dengan makna kelas yang satu digunakan dengan kelas yang lain |
|  | *Generalisasi*: relasi antar kelas dengan makna generalisasi-spesialisasi |
|  | *Dependency*: relasi antar kelas dengan makna keberuntungan antar kelas |
|  | *Aggregation*: relasi antar kelas dengan makna semua bagian |

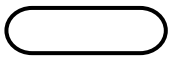
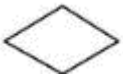
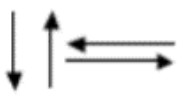
**Sumber: (Ade Hendini, 2016)**

## 2.11. *Activity Diagram*

*Activity Diagram* menggambarkan work flow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas dapat dilakukan oleh sistem. (Ade Hendini, 2016)

*Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan state diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-trigger oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. (Ade Hendini, 2016)

**Tabel 2.3. Komponen Pada Activity Diagram**

| Simbol | Keterangan |
|---|---|
|  | *Action*: aktivitas atau proses yang dilakukan dalam aliran kerja. |
|  | *Initial State*: awal dimulainya aliran kerja pada activity diagram dan hanya ada satu dalam sebuah alur. |
|  | *Final State*: adalah bagian akhir dari alur activity diagram dan bisa ada lebih dari satu dalam sebuah alur |
|  | *Decision*: proses yang digunakan apabila dalam sebuah alur memiliki percabangan |
|  | *Line connector*: digunakan untuk menghubungkan satu simbol dengan simbol yang lainnya dalam sebuah alur |

**Sumber: (Ade Hendini, 2016)**

## 2.12. Penelitian Terdahulu

Berikut ini beberapa penelitian terdahulu yang akan dijadikan landasan dan tolak ukur pada penelitian ini, yaitu:

**Tabel 2.4. Penelitian Terdahulu**

| No. | Judul | Permasalahan | Hasil | Metode |
|---|---|---|---|---|
| 1. | Aplikasi Peminjaman Ruangan Rapat Kantor Gubernur Provinsi Kepulauan Bangka Belitung Berbasis Android<br><br>**Nama Penulis:** Rendy Rian Chrisna Putra & Eza Budi Perkasa<br>**Tahun Penulisan:** 2019<br>**Jurnal**: Jurnal SISFOKOM, Vol 08 No 02. | Peminjaman ruangan rapat di Kantor Gubernur Provinsi Kepulauan Bangka Belitung yang dilakukan saat ini masih melalui proses manual yaitu peminjaman harus menemui langsung pihak pengelola yang ada di kantor Gubernur untuk mendapatkan persetujuan peminjaman ruangan yang dibutuhkan. Di sisi lain saat ingin meminjam ruangan harus menemui pengelola terkait untuk peminjaman ruangan harus menemui langsung pihak-pihak terkait yang terkadang berhalangan hadir atau sedang tidak ada di tempat. | Merancang dan Membangun aplikasi Peminjaman Ruangan Rapat Kantor Gubernur Provinsi Kepulauan Bangka Belitung agar lebih mudah dalam melakukan peminjaman ruangan rapat. | RAD |
| 2. | Aplikasi Sistem Informasi Penyewaan Fasilitas Di Universitas Lancang Kuning Berbasis Online | Proses yang panjang dalam peminjaman fasilitas di universitas tersebut dimana peminjam di haruskan mengajukan surat peminjaman dan mengisi formulir peminjaman secara manual untuk kemudian meminta persetujuan dari Wakil Rektor II. Setelah itu barulah peminjam bisa | Sebuah sistem informasi layanan komplain (Helpdesk) mahasiswa terhadap dosen berbasis android yang dapat diakses melalui *smartphone* dan mahasiswa dapat melakukan komplain (Helpdesk) dari | RAD |

| No. | Judul | Permasalahan | Hasil | Metode |
|---|---|---|---|---|
| | **Nama Penulis:** Nurliana Nasutio **Tahun Penulisan:** 2017 **Jurnal:** Inovtek Polbeng – Seri Informatika Vol 2 No 2. | memesan fasilitas yang dibutuhkan selama fasilitas tersebut tersedia. Disisi lain | *smarphone* mereka masing-masing | |
| 3. | Sistem Aplikasi Peminjaman Fasilitas Universitas Widyatama **Nama Penulis:** Iwan Rijayana **Tahun Penulisan:** 2016 **Jurnal:** Jurnal Khatulistiwa Informatika, Vol 04 No 02. | Bagian Fasilitas Universitas Widyatama, ingin mengembangkan software pengelolaan data peminjaman Fasilitas, dimana pengelolaan data peminjaman Fasilitas masih dilakukan secara manual. Hal ini tentu dapat menjadi hambatan dalam proses pendataan peminjamn Fasilitas, karena staf Bagian Fasilitas harus dapat memproses data peminjaman Fasilitas dengan cepat, tepat dan akurat. | Hasil dari penelitian ini yaitu berupa aplikasi yang dapat digunakan dalam proses pendataan peminjaman Fasilitas. | RAD |

Tabel 2.4. Menjelaskan bahwa penelitian terdahulu menjadi bahan acuan untuk dapat melakukan pembuatan aplikasi Apartemen Victoria Square sehingga aplikasi yang di buat tidak salah dalam perancangan maupun implementasiannya serta menjadi landasan untuk memilih metode perancangan yang tepat bagi aplikasi yang akan dibuat.