PEMBANGUNAN *ANDROID CLASS LIBRARY* DENGAN PENDEKATAN *DOMAIN ENGINEERING* PADA APLIKASI INFORMASI PENJUALAN DAN PENYEWAAN PROPERTI

Mohammad Abdul Iman Syah[1], Alif Finandhita[2]

Teknik Informatika - Universitas Komputer Indonesia
Jalan Dipatiukur 112-114 Bandung
E-mail : imancha_266@ymail.com[1], alif.finandhita@email.unikom.ac.id[2]

ABSTRAK

Tren pencarian properti melalui portal properti *online* kian meningkat menjadikan berkembangnya pembangunan aplikasi jenis informasi penjualan dan penyewaan properti. Pembangunan fitur untuk aplikasi jenis informasi penjualan dan penyewaan properti harus dilakukan secara berulang-ulang sehingga tidak dapat meminimalkan waktu dan biaya yang dibutuhkan dalam pengerjaan aplikasi. Belum tersedianya kode sumber yang dapat diguna ulang merupakan penyebab dari masalah tersebut. Disamping itu, *unmaintainable code* yang dihasilkan akibat dari kurangnya analisis dibandingkan implementasinya dalam pembangunan aplikasi sehingga menyulitkan pengembang untuk melakukan perbaikan atau pengembangan terhadap aplikasi yang dibangun. Pembangunan *class library* adalah solusi dari permasalahan tersebut. Dengan menggunakan metode FODA (*Feature Oriented Domain Analysis*), fitur-fitur pada aplikasi dalam domain yang dapat diguna ulang dianalisis melalui tiga buah *sample* aplikasi yang kemudian dibangun menjadi sebuah *class library.* Berdasarkan hasil dari *unit testing, integration testin*g dan *user acceptance testing* terhadap *class library* yang dibangun, dapat disimpulkan bahwa dengan adanya *class library* waktu dan biaya dalam pembangunan aplikasi dapat diminimalkan serta dapat memudahkan pengembang aplikasi dalam memperbaiki atau mengembangkan aplikasi dengan jenis informasi penjualan dan penyewaan properti.

Kata kunci : *Class Library, Reusable Code, Domain Engineering*

1. PENDAHULUAN

Kebutuhan yang tinggi terhadap hunian dan pergeseran tren masyarakat ke era digital menjadikan portal properti *online* kian populer bagi pemburu rumah [1]. Hal ini memicu berkembangnya pembangunan aplikasi informasi penjualan dan penyewaan properti. Pembangunan aplikasi dengan jenis tersebut masih kurang efektif dan efisien dikarenakan telah banyak aplikasi dengan jenis yng sama namun tidak ada kode sumber yang dapat diguna ulang.

Belum tersedianya kode sumber yang dapat diguna ulang menyebabkan dalam setiap pengerjaan untuk fitur yang sama harus dilakukan secara berulang-ulang sehingga waktu dan biaya yang dibutuhkan dalam pengerjaan aplikasi tidak dapat diminimalkan. Disamping itu juga, proses analisis yang dianggap masih kurang dibandingkan dengan implementasinya dalam setiap pembangunan aplikasi menyebabkan kode sumber yang dihasilkan menjadi *unmantainable code* sehingga pengembang aplikasi merasa kesulitan ketika akan melakukan perbaikan atau pengembangan terhadap aplikasi yang telah dibangun. Berdasarkan masalah tersebut, maka akan dibangun sebuah *class library* untuk domain aplikasi informasi penjualan dan penyewaan properti pada *platform* android.

1.1 Penjualan dan Penyewaan Properti

Tren pencarian properti melalui portal properti *online* kian meningkat seiring penetrasi internet yang semakin meluas saat ini. Penetrasi internet yang kini menjangkau 34,9% penduduk Indonesia berimpak pada beragam sektor, termasuk *real estate*. Bila dulunya media cetak menjadi andalan bagi para pemburu properti untuk menemukan properti impian, kini pelan-pelan portal properti *online* mulai mendominasi [1].

Survei dari Lamudi mengungkapkan, 63% penduduk Indonesia berencana untuk mencari hunian atau berpindah ke rumah baru. Uniknya, mereka melihat internet sebagai media yang tepat untuk menemukan properti yang memenuhi kebutuhan mereka. Pasar *real estate online* kenyataannya sudah menjadi alat favorit bagi pencari properti. Sebanyak 84,9% responden yang disurvei Lamudi mengungkapkan, mereka menggunakan metode *online* untuk menemukan properti ideal. Hanya 6,7% yang masih mengandalkan metode *offline* seperti koran, majalah dan *flyer*, sementara dengan metode lainnya sekitar 8,4% [1].

1.2 *Class Library*

Class library adalah kumpulan *class* yang dibuat dengan tujuan untuk digunakan kembali secara berulang-ulang, baik di dalam satu aplikasi ataupun beberapa aplikasi yang berbeda. Pemakaian kembali ini merupakan prinsip dasar dalam merancang kode program sebagai *class library*, lebih khusus lagi bila *class library* ini bersifat *open source* yang artinya

kode program juga akan dirilis atau disajikan ke konsumen sehingga konsumen *class library* tidak hanya bermaksud memakai *class-class* ini di dalam aplikasi yang dibuatnya, tetapi mereka juga akan membaca *source code*-nya, mempelajarinya, dan bahkan memperbaiki dan mengembangkannya [2].

Class library adalah salah satu contoh keuntungan yang akan diperoleh dari mengimplementasikan teknik *Object Oriented Programming*. *Class library* akan mempercepat pembangunan sebuah aplikasi, sebab tidak ada waktu yang terbuang percuma untuk membangun atau mengerjakan sesuatu yang telah dibuat oleh orang lain dengan bagus (*reinvent the wheel*) [2]. Pengujian yang dapat dilakukan untuk menguji sebuah *class library* adalah dengan melakukan *unit testing*, *integration testing*, dan *user acceptance testing*.

Unit testing umumnya digunakan untuk menjalankan fungsi dari unit kode terkecil (yang bisa berupa metode, kelas atau komponen) dengan cara yang berulang. *Unit testing* diperlukan untuk memverifikasi logika kode secara spesifik pada aplikasi. Misalnya, pada *unit testing* sebuah kelas untuk memastikan bahwa kelas dalam keadaan benar. Biasanya, unit kode diuji secara terpisah karena pengujian hanya mempengaruhi dan meninjau perubahan pada unit itu saja [3].

Integration testing merupakan pengujian yang dilakukan dengan meletakkan komponen-komponen yang berhubungan untuk memastikan modul-modul dan antarmuka dalam suatu aplikasi berinteraksi satu-sama lain dengan baik dan benar. Pengujian integrasi dilakukan dengan mengintegrasikan suatu unit dengan unit yang lain yang saling berinteraksi sehingga menyusun suatu sistem. Pengujian integrasi digunakan untuk mengetahui kesalahan dan masalah yang terdapat pada perangkat lunak [4].

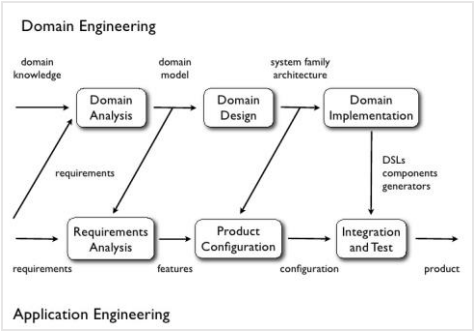
User acceptance testing merupakan pengujian formal yang berhubungan dengan kebutuhan pengguna, persyaratan, dan proses bisnis yang dilakukan untuk menentukan apakah suatu sistem memenuhi kriteria penerimaan atau tidak, dan pengguna atau badan lain yang berwenang sangat mungkin untuk menentukan menerima sistem atau tidak [5].

1.3 *Domain Engineering*

Domain adalah bidang pengetahuan yang dibatasi untuk memaksimumkan kepuasan pihak-pihak yang berkepentingan, memasukkan sekumpulan konsep dan terminologi yang dipahami oleh praktisi-praktisi di bidang itu, dan memasukkan pengetahuan bagaimana membangun sistem-sistem perangkat lunak (atau bagian-bagian dari sistem-sistem perangkat lunak) di bidang itu [6].

Domain perangkat lunak merupakan kategori dari setiap jenis perangkat lunak yang ada. Terdapat tujuh kategori mengenai jenis domain perangkat lunak yaitu s*ystem software, application software, engineering/scientific software, embedded software, product-line software* w*eb application* dan *artificial intelligence software* [7].

Domain Engineering adalah aktivitas mengumpulkan, mengorganisasikan, dan menyimpan pengalaman dalam membangun sistem-sistem atau bagian-bagian dari sistem-sistem pada domain tertentu dalam bentuk aset-aset yang dapat digunakan ulang, juga menyediakan sarana-sarana yang memadai untuk menggunaulang aset-aset ini (seperti kualifikasi, penyebaran, adaptasi, perakitan, dan sebagainya) ketika membangun sistem-sistem baru. *Domain Engineering* terdiri dari tiga tahap [6]:



Gambar 1. *Domain Engineering*

1. *Domain analysis*, mendefinisikan sekumpulan kebutuhan yang dapat digunaulang untuk sistem dalam domain.
2. *Domain design*, membangun satu arsitektur yang umum untuk sistem dalam domain.
3. *Domain implementation*, mengimplementasi aset-aset yang dapat digunaulang.

1.4 *Feature-Oriented Domain Analysis* (FODA)

FODA adalah bagian dari *Model-Based Software Engineering* (MBSE), yaitu pendekatan komprehensif untuk rekayasa perangkat lunak berdasarkan *Domain Engineering* yang dikembangkan oleh *Software Engineering Institute* (SEI) di mana FODA termasuk kedalam Analisis Domain [6].

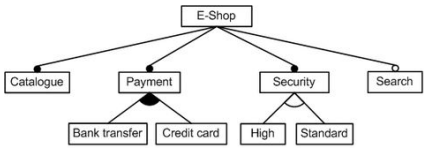
SEI memperkenalkan FODA pada tahun 1990. Metodologi tersebut telah lebih jauh berkembang dengan penggunaan setelahnya. Metodologi FODA terdiri dari tiga langkah yaitu analisis konteks, domain *modeling* dan arsitektur *modeling* [8].

Analisis konteks mendefinisikan ruang lingkup domain, hubungannya dengan domain yang lain, *input*, *output* dari domain dan kebutuhan data yang tersimpan. Hal ini dinyatakan sebagai model konteks dan dapat dilakukan dengan menggunakan diagram struktur dan diagram konteks [8].

Domain *modeling* berkaitan dengan mengidentifikasi persamaan dan perbedaan berbagai aplikasi dalam domain. Analisis dilakukan untuk mendapatkan fitur dari domain dan kosakata standar yang digunakan untuk domain. Entitas dan kebutuhan perangkat lunak dipahami menggunakan teknik spesifikasi khusus seperti aliran kontrol dan aliran data. Domain *modeling* menggunakan tiga jenis teknik dan menghasilkan *output* sebagai

berikut [8]:

1. *Information model*, memberikan informasi dalam sistem dan hubungan antara entitas.
2. *Feature model*, menjelaskan fitur dari sistem berdasarkan perspektif pengguna akhir dan berguna untuk komunikasi ke pengguna dan mendapatkan tampilan yang lengkap dan konsisten dari domain. *Feature model* biasanya direpresentasikan sebagai diagram fitur.
3. *Operational model*, struktur fungsi dan urutan yang membentuk dasar bagi pengembang untuk melihat bagaimana menyediakan fitur yang dibutuhkan oleh pengguna.



Gambar 2. Diagram Fitur

Awalnya, FODA memiliki fase ketiga yang disebut *Architecture Modeling*. Fase ini tidak lagi merupakan bagian dari FODA, melainkan diubah menjadi fase *domain design*, yang mengikuti keseluruhan *framework* dari *Model-Based Software Engineering* [6].

2. ISI PENELITIAN

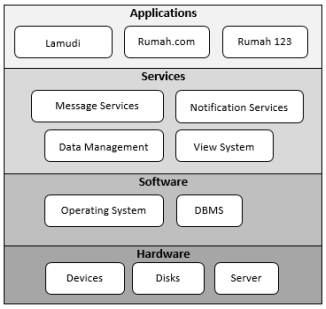
Class library yang akan dibangun merupakan *class library* untuk domain aplikasi informasi penjualan dan penyewaan properti. Pembangunan *class library* berdasarkan metode *domain engineering* terdiri dari tiga tahapan yaitu *domain analysis*, *domain design* dan *domain implementation.*

2.1. *Domain Analysis*

Domain analysis merupakan aktivitas untuk mendapatkan persamaan dan perbedaan fitur yang ada dalam domain sehingga menghasilkan kumpulan dari model domain yang dapat diguna ulang. Aplikasi yang digunakan untuk mendapatkan kandidat komponen yang dapat diguna ulang adalah Lamudi, Rumah.com dan Rumah123. *Analysis domain* berdasarkan metode FODA terdiri dari dua tahap, yaitu *context analysis* dan *domain modeling* [6].

2.1.1. *Context Analysis*

Context analysis merupakan tahapan untuk menentukan ruang lingkup dari domain yang mungkin untuk menghasilkan produk domain. Dalam tahap ini, hubungan antara domain dan unsur-unsur eksternal atau entitas dari domain dianalisis untuk mengetahui perbedaannya serta menggunakan hasil dari *context analysis* tersebut sebagai batasan area dari domain. Hasil dari *context analysis* berupa *context model* yang terdiri dari *structure diagram* dan *context diagram* [6].



Gambar 3. *Structure Diagram*

Gambar diatas menunjukkan *structure diagram* dari aplikasi dalam domain. Berdasarkan gambar tersebut, terdapat empat *layer* yang ada pada aplikasi dalam domain yaitu *layer Applications*, *Services*, *Software* dan *Hardware*.

Layer applications merupakan *layer* yang berisi kumpulan dari aplikasi-aplikasi dengan jenis informasi penjualan dan penyewaan properti yang digunakan sebagai *sample* untuk mendapatkan *library* yang dapat diguna ulang. Terdapat tiga buah *sample* aplikasi pada *layer applications* yang digunakan, yaitu terdiri dari aplikasi Lamudi, Rumah.com dan Rumah 123.

Layer services merupakan *layer* yang berisi kumpulan dari layanan yang disediakan oleh aplikasi-aplikasi yang ada pada *layer applications* yang terdiri dari empat layanan:

1. *Data Management*, layanan yang digunakan untuk pengelolaan data.
2. *Message Services*, layanan yang digunakan untuk pengiriman pesan berupa pesan teks, email dan telepon.
3. *Notification Services*, layanan yang digunakan untuk pengiriman notifikasi kepada pengguna.
4. *View System*, layanan yang digunakan untuk menampilkan antarmuka-antarmuka yang ada pada masing-masing aplikasi.

Layer software merupakan *layer* pendukung dalam bentuk perangkat lunak yang digunakan oleh aplikasi dalam domain yang terdiri dari dua perangkat yaitu:

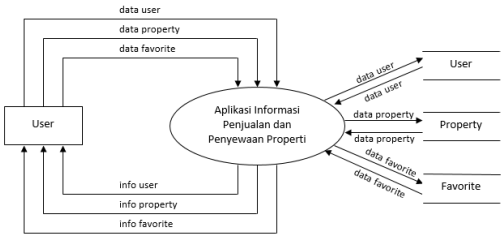
1. *Operating System*, perangkat lunak yang dibutuhkan oleh aplikasi dalam domain untuk menjalankan layanan-layanan yang disediakan.
2. DBMS (*Database Management System*), perangkat lunak yang dibutuhkan untuk menyimpan data.

Layer hardware merupakan *layer* pendukung dalam bentuk perangkat keras yang digunakan oleh aplikasi dalam domain yang terdiri dari tiga perangkat yaitu:

1. *Devices*, perangkat keras yang dibutuhkan oleh aplikasi dalam domain untuk menampilkan layanan-layanan yang disediakan.
2. *Disks*, perangkat keras yang dibutuhkan

untuk menyimpan data-data yang digunakan.

3. *Server*, perangkat keras yang dibutuhkan oleh aplikasi sehingga data-data yang tersimpan dapat diakses melalui suatu *network.*



Gambar 4. *Context Diagram*

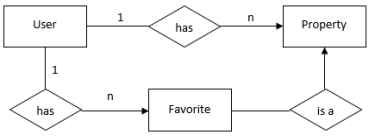
Gambar diatas menunjukkan *context diagram* dari aplikasi dalam domain. Berdasarkan gambar tersebut, terdapat satu entitas yang berinteraksi langsung dengan aplikasi dalam domain yaitu *User.* Disamping itu, terdapat tiga *data store* yang ada pada aplikasi dalam domain yaitu *User*, *Property* dan *Favorite.*

2.1.2. *Domain Modeling*

Domain modeling merupakan tahapan untuk mengidentifikasi dan memodelkan persamaan dan perbedaan fitur-fitur yang ada pada masing-masing aplikasi dalam domain berdasarkan hasil dari *context analysis* sebagai batasannya. *Domain modeling* terdiri dari tiga tahap, yaitu *information analysis*, *feature analysis* dan *operational analysis* [6].

2.1.2.1. *Information Analysis*

Information analysis bertujuan untuk menggambarkan pengetahuan domain dalam bentuk entitas dan hubungan antar entitasnya pada domain. Hasil dari *information analysis* yaitu *information model* yang dapat berupa jaringan semantik, *entity-relationship model* atau pemodelan berorientasi obyek [6].

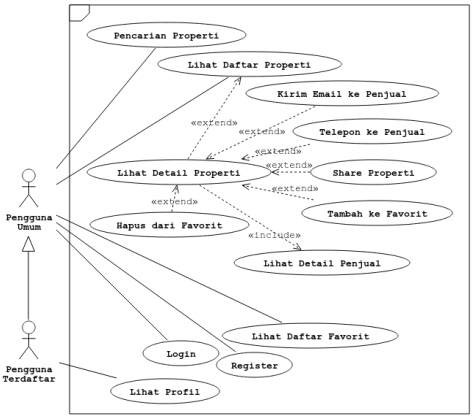


Gambar 5. *Entity-relationship Model*

Gambar diatas menunjukkan *entity-relationship* dari aplikasi dalam domain. Berdasarkan gambar tersebut, terdapat tiga entitas yang ada pada aplikasi dalam domain yaitu *User*, *Property* dan *Favorite*.

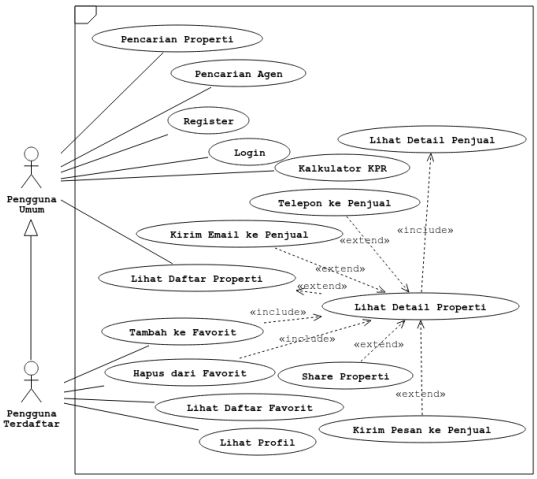
2.1.2.2. *Feature Analysis*

Feature analysis merupakan tahapan untuk mengetahui fitur-fitur yang disediakan pada setiap aplikasi dalam domain dari perspektif pengguna akhir. Persamaan dan perbedaan fitur dari masing-masing aplikasi juga dianalisis. Hasil dari *feature analysis* berupa *feature model* [6].



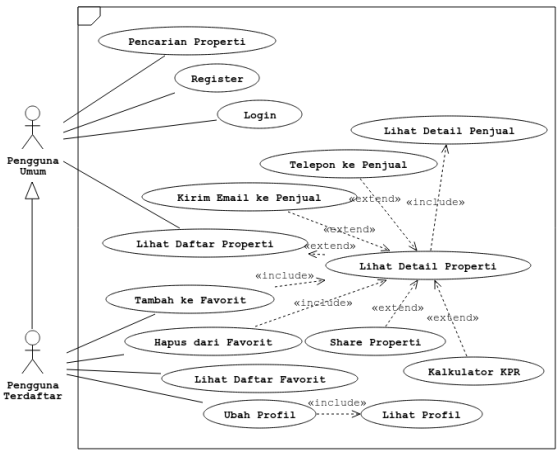
Gambar 6. *Use Case* Lamudi

Gambar diatas menunjukkan *use case* dari aplikasi Lamudi. Berdasarkan gambar tersebut, terdapat dua level pengguna yaitu pengguna umum dan pengguna terdaftar serta terdapat tiga belas *use case* yang ada pada aplikasi Lamudi.



Gambar 7. *Use Case* Rumah.com

Gambar diatas menunjukkan *use case* dari aplikasi Rumah.com. Berdasarkan gambar tersebut, terdapat dua level pengguna yaitu pengguna umum dan pengguna terdaftar serta terdapat enam belas *use case* yang ada pada aplikasi Rumah.com.



Gambar 8. *Use Case* Rumah123

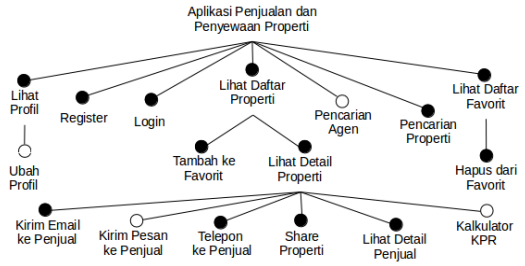
Gambar diatas menunjukkan *use case* dari aplikasi Rumah123. Berdasarkan gambar tersebut, terdapat dua level pengguna yaitu pengguna umum dan pengguna terdaftar serta terdapat lima belas *use case* yang ada pada aplikasi Rumah123.

Berdasarkan analisis fitur dari masing-masing aplikasi diatas, selanjutnya dilakukan analisis terhadap persamaan dan perbedaan fitur-fitur yang ada pada masing-masing aplikasi dalam domain.

Tabel 1. Persamaan dan Perbedaan Fitur

Fitur	Lamudi	Rumah .com	Rumah 123
Pencarian Properti	√	√	√
Pencarian Agen	-	√	-
Lihat Daftar Properti	√	√	√
Lihat Detail Properti	√	√	√
Kirim Email ke Penjual	√	√	√
Kirim Pesan ke Penjual	-	√	-
Telepon ke Penjual	√	√	√
Share Properti	√	√	√
Tambah ke Favorit	√	√	√
Hapus dari Favorit	√	√	√
Lihat Detail Penjual	√	√	√
Kalkulator KPR	-	√	√
Lihat Daftar Favorit	√	√	√
Register	√	√	√
Login	√	√	√
Lihat Profil	√	√	√
Ubah Profil	-	-	√

Tabel diatas menunjukkan persamaan dan perbedaan fitur pada masing-masing aplikasi. Berdasarkan tabel diatas, terdapat empat fitur yang hanya dimiliki oleh salah satu aplikasi, terdapat satu fitur yang tidak dimiliki oleh salah satu aplikasi dan terdapat lima belas fitur yang dimiliki oleh setiap aplikasi.



Gambar 9. *Feature Diagram*

Gambar diatar menunjukkan *feature diagram* dari aplikasi dalam domain. Berdasarkan *feature diagram* diatas, terdapat dua jenis fitur yang ada pada aplikasi dalam domain yaitu fitur *mandatory* dan fitur *optional.* Fitur-fitur yang akan dijadikan sebagai *class library* adalah seluruh fitur-fitur *mandatory* serta fitur-fitur *optional* yang mungkin dibutuhkan dalam pembangunan aplikasi dalam domain.

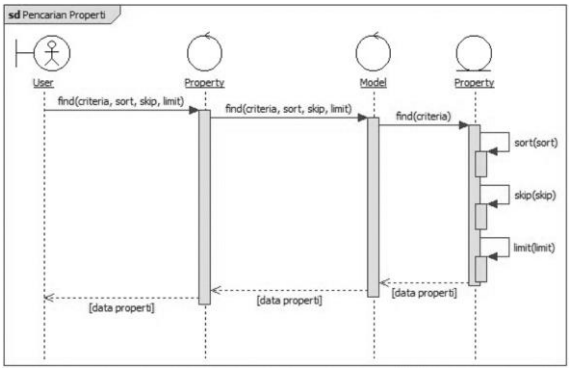
Tabel 2. Fitur *Class Library*

Fitur
Pencarian Properti
Lihat Daftar Properti
Lihat Detail Properti
Kirim Email ke Penjual
Telepon ke Penjual
Share Properti
Tambah ke Favorit
Hapus dari Favorit
Lihat Detail Penjual
Kalkulator KPR
Lihat Daftar Favorit
Register
Login
Lihat Profil

Tabel diatas menunjukkan fitur-fitur yang akan dijadikan *class library*. Berdasarkan tabel tersebut, terdapat empat belas fitur yang akan dijadikan *class library.*

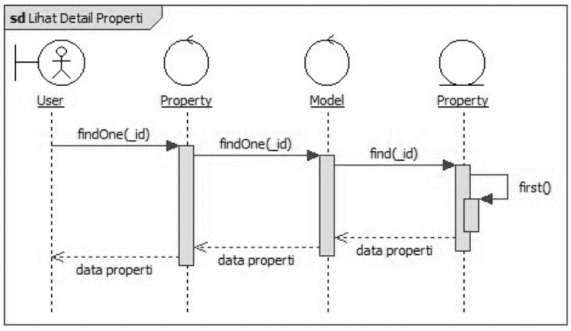
2.1.2.3. *Operational Analysis*

Operational analysis merupakan tahapan untuk mendeskripsikan kontrol dan *data-flow* pada aplikasi dalam domain berdasarkan *context analysis*, *information model* dan *feature model* yang telah dilakukan. Hasil dari *operational analysis* berupa *operational model* [6].



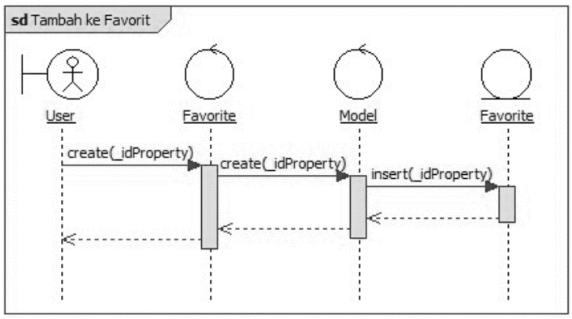
Gambar 10. *Sequence Diagram* Pencarian Properti

Gambar diatas menunjukkan *operational model* dari fitur pencarian properti. Berdasarkan gambar tersebut terdapat dua *class* yaitu *Property* dan *Model* serta satu *entity* yaitu *Property* yang digunakan untuk fitur pencarian properti.



Gambar 11. *Sequence Diagram* Lihat Detail Properti

Gambar diatas menunjukkan *operational model* dari fitur lihat detail properti. Berdasarkan gambar tersebut terdapat dua *class* yaitu *Property* dan *Model* serta satu *entity* yaitu *Property* yang digunakan untuk fitur lihat detail properti.



Gambar 12. *Sequence Diagram* Tambah ke Favorit

Gambar diatas menunjukkan *operational model* dari fitur tambah ke favorit. Berdasarkan gambar tersebut terdapat dua *class* yaitu *Favorite* dan *Model* serta satu *entity* yaitu *Favorite* yang digunakan untuk fitur tambah ke favorit.

2.2. *Domain Design*

Domain design merupakan tahapan perancangan terhadap *class library* yang akan dibangun yang terdiri dari perancangan *class* dan *package*.

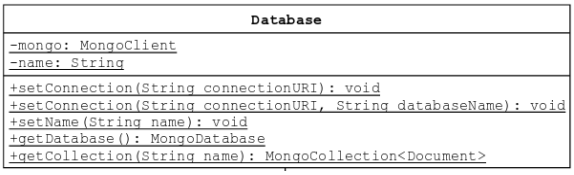
2.2.1. Perancangan *Class*

Berdasarkan hasil dari analisis-analisis yang telah dilakukan, maka fitur-fitur yang akan dijadikan *class library* kemudian dikategorikan dan atau diklasifikasikan berdasarkan konteks, keterkaitan fungsi dan data yang diolahnya untuk menentukan nama *class* dan *method*-nya.

Tabel 3. Pengklasifikasian Fitur

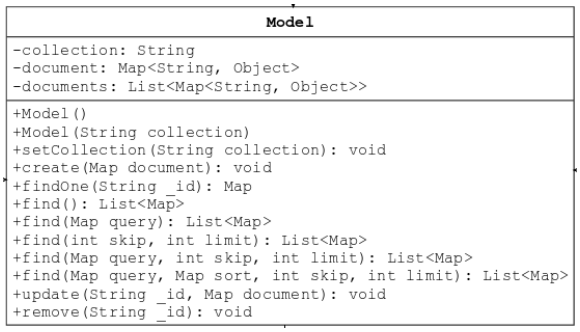
Fitur	Kelas
Tambah ke Favorit	*Favorite*
Hapus dari Favorit	
Lihat Daftar Favorit	
Pencarian Properti	*Property*
Lihat Daftar Properti	
Lihat Detail Properti	
Share Properti	
Kalkulator KPR	
Lihat Detail Penjual	*User*
Kirim Email ke Penjual	
Telepon ke Penjual	
Lihat Profil	
Register	
Login	
-	*Model*
-	*Database*

Tabel diatas menujukkan pengkategorian fitur-fitur untuk *class library*. Berdasarkan tabel tersebut, terdapat lima *class* yang akan dibangun yaitu *class Favorite*, *Property*, *User*, *Model* dan *Database*.



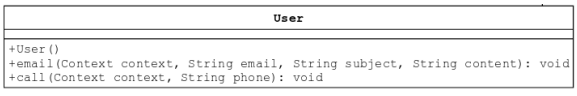
Gambar 13. *Class Database*

Gambar diatas menujukkan struktur *class Database* pada *class library*. Berdasarkan gambar tersebut terdapat dua *attribute* yaitu *mongo* dan *name* serta lima *method* yaitu *setConnection*, *setConnection*, *setName*, *getDatabase* dan *getCollection* yang ada pada *class Database.*



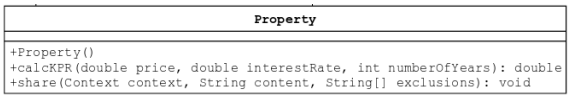
Gambar 14. *Class Model*

Gambar diatas menujukkan struktur *class Model* pada *class library*. Berdasarkan gambar tersebut terdapat tiga *attribute* yaitu *collection*, *document* dan *documents* serta sepuluh *method* yaitu *setCollection*, *create*, *findOne*, *find*, *find*, *find*, *find*, *find*, *update* dan *remove* yang ada pada *class Model.*



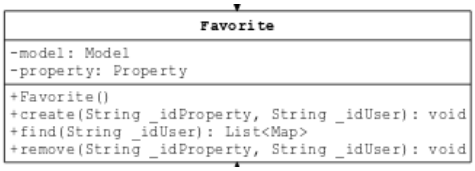
Gambar 15. *Class User*

Gambar diatas menujukkan struktur *class User* pada *class library*. Berdasarkan gambar tersebut terdapat dua *method* yaitu *email* dan *call* yang ada pada *class User*.



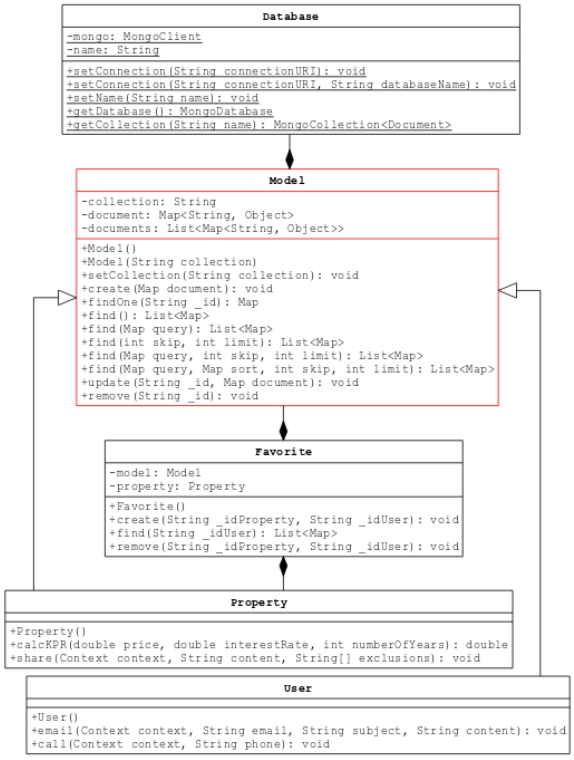
Gambar 16. *Class Property*

Gambar diatas menujukkan struktur *class Property* pada *class library*. Berdasarkan gambar tersebut terdapat dua *method* yaitu *calcKPR* dan *share* yang ada pada *class Property*.



Gambar 17. *Class Favorite*

Gambar diatas menujukkan struktur *class Favorite* pada *class library*. Berdasarkan gambar tersebut terdapat dua *attribute* yaitu *model* dan *property* serta tiga *method* yaitu *create*, *find* dan *remove* yang ada pada *class Favorite*.

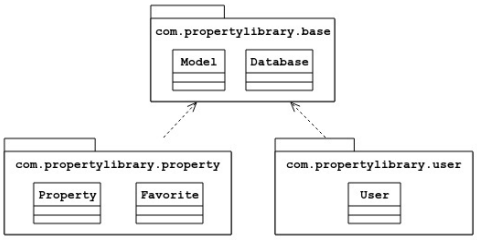


Gambar 18. *Class Diagram*

Gambar diatas menunjukkan hubungan atau relasi antar *class* yang ada pada *class library* yang dibangun. *Class Model* memiliki relasi *composition* dengan *class Database*. *Class Favorite* memiliki relasi *composition* dengan *class Model* dan *Property*. *Class Property* dan *class User* memiliki relasi *generalization* dengan *class Model*. *Class Model* merupakan *class* yang tidak boleh diakses secara langsung dalam penggunaannya, melainkan harus melalui *class* turunannya.

2.2.2. Perancangan *Package*

Perancangan *package* merupakan tahapan untuk merancang *package* berdasarkan dari perancangan *class-class* yang telah dilakukan. Perancangan *package* bertujuan untuk memudahkan dalam memahami dan memanggil dari struktur kelas yang dibangun.



Gambar 19. Perancangan *Package*

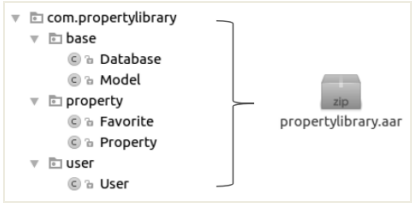
Berdasarkan gambar tersebut, terdapat tiga *package* yang ada pada *class library* yang dibangun yaitu *com.propertylibrary.base*, *com.propertylibrary.property* dan *com.propertylibrary.user*. Pada *package com.propertylibrary.base* terdapat *class Model* dan *Database*. Pada *package com.propertylibrary.property* terdapat *class Property* dan *Favorite*. Sedangkan pada *package com.propertylibrary.user* terdapat *class User*. *Package com.propertylibrary,property* dan *com.propertylibrary.user* memiliki *depencency* terhadap *package com.propertylibrary.base*.

2.3. *Domain Implementation*

Domain implementation merupakan tahapan pembangunan terhadap *class library* yang akan dibangun dengan membuat *package* dan *class-class* sesuai dengan perancangan yang telah dilakukan. Kemudian akan dilakukan integrasi terhadap *package* dan *class-class* yang telah dibuat sehingga menjadi sebuah *class library* serta melakukan implementasi dan pengujian terhadap *class library* tersebut dengan menggunakan *unit testing*, *integration testing* dan *user acceptance testing.*

2.3.1. Implementasi *Class Library*

Implementasi *class library* merupakan tahapan untuk mengimplementasikan dari *class library* yang dibangun. *Class library* yang telah dibangun dikompilasi menjadi bentuk *file android archive (.aar)* untuk memudahkan dalam penggunaannya.

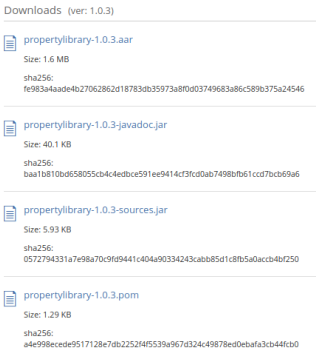


Gambar 20. Kompilasi *Class Library*

Gambar diatas merupakan hasil dari kompilasi dari *class library* yang dibangun. Implementasi terhadap *class library* yaitu dengan menggunakan

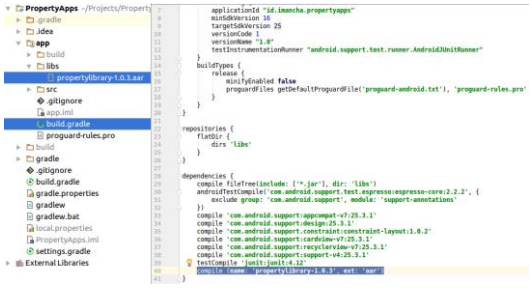
class library tersebut dalam pembangunan aplikasi informasi penjualan dan penyewaan properti. Adapun langkah-langkahnya yaitu sebagai berikut:

1. Mengunduh *class library* melalui *maven repository* seperti pada Gambar berikut [9]:



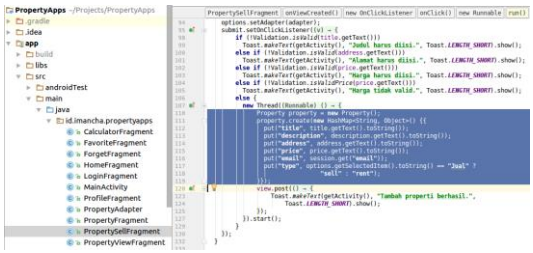
Gambar 21. Mengunduh *Class Library*

2. Menambahkan *class library* sebagai *dependency* dari *project.*



Gambar 22. Menambahkan *Class Library*

3. Menggunakan *class* dan *method* sesuai kebutuhan.



Gambar 23. Menggunakan *Class Library*

2.3.2. Pengujian *Class Library*

Pengujian *class library* merupakan tahapan untuk menguji *class library* yang telah dibangun untuk memastikan bahwa *class library* yang dibangun tersebut telah sesuai dengan tujuan dan kebutuhan. Pengujian *class library* terdiri dari tiga jenis yaitu *unit testing*, *integration testing* dan *user acceptance testing*.

2.3.3.1. *Unit Testing*

Unit testing merupakan tahapan untuk menguji setiap unit berupa metode-metode yang ada pada *class library* yang telah dibangun. *Unit testing* bertujuan untuk memastikan bahwa semua *method* yang telah dibuat sesuai dengan fungsinya.

Tabel 4. *Unit Testing*

Komponen Uji (*Class*)	Butir Uji (*Method*)	Jenis Uji	Hasil
Database	*setConnection (String connectionURI)*	*Unit Testing*	*Pass*
	setConnection (String connectionURI, String databaseName)	*Unit Testing*	*Pass*
	setName(String name)	*Unit Testing*	*Pass*
	getDatabase()	*Unit Testing*	*Pass*
	getCollection(String name)	*Unit Testing*	*Pass*
Model	*setCollection(String collection)*	*Unit Testing*	*Pass*
	create(Map document)	*Unit Testing*	*Pass*
	findOne(String _id)	*Unit Testing*	*Pass*
	find()	*Unit Testing*	*Pass*
	find(Map query)	*Unit Testing*	*Pass*
	find(int skip, int limit)	*Unit Testing*	*Pass*
	find(Map query, int skip, int limit)	*Unit Testing*	*Pass*
	find(Map query,Map sort, int skip, int limit)	*Unit Testing*	*Pass*
	update(String _id, Map document)	*Unit Testing*	*Pass*
	remove(String _id)	*Unit Testing*	*Pass*
Property	*calcKPR(double price, doubleinterestRate, int numberOfYears)*	*Unit Testing*	*Pass*
	share(Context context, String content, String[] exclusions)	*Unit Testing*	*Pass*
Favorite	*create(String _idProperty, String _idUser)*	*Unit Testing*	*Pass*

Komponen Uji (*Class*)	Butir Uji (*Method*)	Jenis Uji	Hasil
	find(String _idUser)	*Unit Testing*	*Pass*
	remove(String _idProperty, String _idUser)	*Unit Testing*	*Pass*
User	*email(Context context, String email, String subject, String content)*	*Unit Testing*	*Pass*
	call(Context context, String phone)	*Unit Testing*	*Pass*

Tabel diatas merupakan *scenario* dan hasil dari *unit testing* pada *class library*. Berdasarkan tabel tersebut dapat disimpulkan bahwa setiap unit berupa *method* pada *class library* yang dibangun sudah berjalan sesuai dengan yang diharapkan baik itu dari segi validasi maupun proses penanganan kesalahan serta menghasilkan *output* yang diinginkan.

2.3.3.2. *Integration Testing*

Integration testing merupakan tahapan pengujian dengan menggabungkan beberapa unit yang ada pada *class library*. *Integration testing* bertujuan untuk memastikan bahwa setiap unit *class* yang telah dibangun dapat diintegrasikan atau digunakan dengan *unit* yang lain seperti pada antarmuka halaman. *Integration testing* dilakukan dengan cara membangun sebuah aplikasi dengan menggunakan *class library* yang telah dibangun.

Tabel 5. *Integration Testing*

Komponen Uji (*Class*)	Butir Uji (Fitur)	Jenis Uji	Hasil
Property	Tambah Properti	*Integration Testing*	*Pass*
	Pencarian Properti	*Integration Testing*	*Pass*
	Lihat Daftar Properti	*Integration Testing*	*Pass*
	Lihat Detail Properti	*Integration Testing*	*Pass*
	Share Properti	*Integration Testing*	*Pass*
	Kalkulator KPR	*Integration Testing*	*Pass*
Favorite	Tambah ke Favorit	*Integration Testing*	*Pass*
	Hapus dari Favorit	*Integration Testing*	*Pass*
	Lihat Daftar Favorit	*Integration Testing*	*Pass*
User	Lihat Detail Penjual	*Integration Testing*	*Pass*
	Kirim Email ke Penjual	*Integration Testing*	*Pass*
	Telepon ke Penjual	*Integration Testing*	*Pass*
	Lihat Profil	*Integration Testing*	*Pass*
	Register	*Integration Testing*	*Pass*
	Login	*Integration Testing*	*Pass*

Tabel diatas merupakan *scenario* dan hasil dari *integration testing* pada *class library*. Berdasarkan tabel tersebut dapat disimpulkan bahwa setiap unit *class library* yang dibangun dapat berintegrasi dengan unit lainnya menjadi sebuah komponen dan berfungsi sesuai dengan fitur yang diinginkan.

2.3.3.3. *User Acceptance Testing*

User acceptance testing merupakan pengujian yang dilakukan di lingkungan pengguna dimana pengguna melakukan penilaian terhadap *class library* yang dibangun dengan menggunakan media wawancara.

Tabel 6. *User Acceptance Testing*

Pertanyaan	Jawaban
Apakah *class library* yang dibangun dapat membantu dalam membangun jenis aplikasi informasi penjualan dan penyewaan properti pada *platform* android?	Sangat membantu karena kalau tidak ada *class library* setiap mau bikin aplikasi informasi penjualan dan penyewaan properti pada perangkat android harus dimulai dari awal.
Apakah *class library* yang dibangun telah memenuhi fungsionalitas umum pada jenis aplikasi informasi penjualan dan penyewaan properti pada *platform* android?	Secara keseluruhan sudah memenuhi untuk mencapai *minimum viable product*.
Apakah *class library* yang dibangun mudah untuk digunakan?	Ya, tapi mungkin butuh waktu untuk menyesuaikan dengan skema yang ada pada *class library* tersebut.
Apakah *class library* yang dibangun dapat membuat kode sumber lebih terstruktur?	Dengan adanya *class library*, kode sumber dapat dibuat menjadi lebih terstruktur karena pada *class library* yang dibangun sudah dibuat

Pertanyaan	Jawaban
	class-class sesuai dengan fiturnya.
Apakah *class library* yang dibangun dapat meminimalkan waktu dan biaya pengerjaan aplikasi?	Kalau dibandingkan untuk pengerjaan fitur dengan menggunakan *class library* dan tanpa menggunakan *class library* maka dapat dibilang kalau pengerjaaan fitur dengan menggunakan *class library* lebih cepat serta dapat menghemat waktu dan biaya.
Apa kekurangan yang ada pada *class library* yang dibangun?	Kekurangan pada *class library* yang dibangun yaitu masih minimnya validasi dan penanganan kesalahan pada *method*nya, walaupun sudah ada beberapa. Begitu juga untuk dokumentasinya mungkin bisa lebih di perjelas lagi.

Tabel diatas merupakan *scenario* dan hasil dari *user acceptance testing.* Berdasarkan tabel tersebut dapat disimpulkan bahwa *class library* yang dibangun telah sesuai dengan tujuan, yaitu memudahkan pengembang aplikasi dalam membangun atau mengembangkan jenis aplikasi informasi penjualan dan penyewaan properti pada *platform* android.

3. PENUTUP

Berdasarkan pembahasan yang telah disampaikan dari hasil analisa, implementasi dan pengujian pada *class library* yang dibangun dapat disimpulkan bahwa:

1. *Class library* dapat meminimalkan waktu dan biaya dalam pembangunan aplikasi dengan domain informasi penjualan dan penyewaan properti.
2. *Class library* dapat memudahkan pengembang aplikasi dalam memperbaiki atau mengembangkan aplikasi dengan domain informasi penjualan dan penyewaan properti yang dibangun.

Saran untuk pengembangan *class library* selanjutnya adalah:

1. Menambahkan fungsionalitas yang belum ada pada *class library* yang dibangun seperti fungsionalitas untuk transaksi.
2. Melengkapi dokumentasi penggunaan *class library* sehingga pengembang aplikasi tidak mengalami kesulitan dalam menggunakan *class library*.
3. Menambahkan validasi dan penanganan kesalahan yang lebih lengkap pada setiap *method* pada *class library* sehingga memudahkan pengembang aplikasi untuk mengetahui kesalahannya.

DAFTAR PUSTAKA

[1] Properti, *Tren Pemasaran Properti Dalam Jaringan Kian Populer & Dominan*, 2016. [*Online*]. *Available*: http://properti.bisnis.com/read/20160210/107/517761/tren-pemasaran-properti-dalam-jaringan-kian-populer-dominan. [*Accessed*: 14- Feb- 2016].

[2] M. Azis, *Object Oriented Programming dengan Php 5*. Jakarta: Elex Media Komputindo, 2005.

[3] G. Testing, *Getting Started with Testing | Android Developers*, Developer.android.com, 2017. [*Online*]. *Available*: https://developer.android.com/training/testing/start/index.html. [*Accessed*: 14- Feb- 2016].

[4] J. Hermawan, Analisa Desain & Pemrograman Berorientasi Obyek dengan UML dan Visual Basic.NET. Yogyakarta: Andi, 2004.

[5] M. Huda and B. Komputer, *Membuat Aplikasi Database dengan Java, MySQL, dan NetBeans*. Jakarta: PT Elex Media Komputindo, 2010.

[6] Czarnceki, Krzysztof, Ulrich W. Eisenecker, *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2010.

[7] R. Pressman, *Software engineering: a practitioner's approach, 7th ed*., McGraw-Hill, 2010.

[8] Alana E., Rodriguez A., GMV Domain Engineering Methodology Survey. CORDET, 2007.

[9] propertylibrary, Bintray.com, 2017. [Online]. Available: https://bintray.com/imancha/maven/propertylibrary. [Accessed: 10-Aug-2017].