Pembangunan *Class Library* untuk Domain *Product Management* di Aplikasi *M-Commerce* pada Android

Adam Mukharil Bachtiar[1], Arifin Bardansyah[2]

***Abstract*- Criteo reports that the use of M-Commerce in 2015 has increased about 15% - 26%, compared to 2014. In 2015, users access M-Commerce using mobile phone more often than other devices. Around 54% of those users use mobile app while 46% use mobile browser. In addition, there is an increase in conversion value of the number of users switching from mobile browser to mobile app by 120%. In all these activities, 286% of users focus on product viewing activities. This phenomenon requires software developers to develop M-Commerce in a short time. On the other hand, the developers need time to develop M-Commerce. One of the time-consuming activities is the development of functions for product management. Based on the software development, there is one concept that can be used, which is class library. A class library is a collection of classes and methods that provide functionality as basis for building a system within a case domain. The first step in this research is to do domain analysis to acquire frozenspot, hotspot, and function mechanism that will be provided in class library. The result of the hotspot will be analyzed to obtain the nine basic functions in the domain of product catalog case in M-Commerce. These functions will be mapped into ten classes in a class library which will be tested using unit testing and integration testing that can be used by M-Commerce developers to accelerate the development of product management on M-Commerce.**

***Intisari*- Criteo melaporkan bahwa penggunaan *M-Commerce* pada tahun 2015 mengalami peningkatan sekitar 15% - 26%, dibandingkan pada tahun 2014. Pengguna *M-Commerce* di tahun 2015 lebih banyak mengakses *M-Commerce* menggunakan telepon seluler. Penggunaan ini terbagi menjadi 54% menggunakan aplikasi *mobile* dan 46% menggunakan *mobile browser*. Selain itu, terdapat fakta bahwa ada peningkatan nilai konversi jumlah pengguna yang beralih dari *mobile browser* ke aplikasi *mobile* sebesar 120%. Dari seluruh kegiatan tersebut, 286% pengguna terpusat pada kegiatan melihat produk. Hal ini memaksa banyak pengembang perangkat lunak untuk melakukan pembangunan *M-Commerce* dalam waktu yang singkat. Permasalahan yang timbul adalah waktu yang dibutuhkan untuk pembangunan *M-Commerce* sangat panjang apabila dilakukan dari awal dan salah satu kegiatan yang memakan waktu adalah pembangunan fungsi untuk manajemen produk. Berdasarkan keilmuan pembangunan perangkat lunak, ada satu konsep yang bisa dijadikan solusi untuk masalah ini, yaitu *class library*. *Class library* merupakan kumpulan *class* dan *method* yang menyediakan fungsionalitas sebagai dasar pembangunan sebuah sistem di dalam sebuah domain kasus. Tahap awal pada penelitian ini adalah melakukan analisis domain terhadap tiga aplikasi *M-Commerce* untuk mendapatkan *frozenspot, hotspot,* dan mekanisme fungsi yang akan disediakan pada *class library*. Hasil dari *hotspot* kemudian dianalisis sehingga didapatkan sembilan fungsi dasar pada domain kasus katalog produk di *M-Commerce*. Fungsi-fungsi ini kemudian dipetakan menjadi sepuluh *class* di dalam sebuah *class library* yang kemudian diuji secara unit dan uji integrasi agar bisa digunakan oleh para pengembang *M-Commerce* untuk mempercepat pembangunan manajemen produk pada *M-Commerce*.**

***Kata Kunci— Class Library*, *M-Commerce*, Manajemen Produk, Aplikasi *Mobile*.**

I. PENDAHULUAN

Penggunaan *M-Commerce* sebagai media perdagangan dan transaksi tumbuh dan berkembang pesat hingga tahun 2015. *Smartphone* yang relatif lebih murah dibandingkan dengan PC/Laptop menjadi penyebab utama berkembangnya *M-Commerce* [1]. Criteo melaporkan penggunaan *M-Commerce* pada tahun 2015 rata-rata sebesar 30% dan mencapai puncak pada kuartil keempat, sebesar 44%. Hal tersebut menunjukkan peningkatan rata-rata sebesar 15% - 26% dibandingkan tahun 2014 [2]. Penggunaan *M-Commerce* dapat dilakukan melalui aplikasi *mobile* dan *mobile browser*. Criteo mendokumentasikan penggunaan *M-Commerce* pada tahun 2015 lebih banyak melalui aplikasi *mobile*, yaitu sebesar 54% dan sisanya *mobile browser* sebesar 46% [2]. Peningkatan konversi *mobile browser* ke aplikasi *mobile* sebesar 120%, yang meliputi produk yang dilihat oleh pengguna sebesar 286%, transaksi pembelian sebesar 85%, dan transaksi pembayaran sebesar 23% [2]. Berdasarkan fakta tersebut, diketahui bahwa perkembangan *M-Commerce* cukup pesat dalam 1 tahun dan dimungkinkan akan terus berkembang. Hal ini berimbas kepada *programmer* sebagai pembangun *M-Commerce* yang harus dapat membangun serta mengembangkan *M-Commerce* sesuai kecepatan kebutuhan pasar. Perbedaan *E-Commerce* dan *M-Commerce* yang mendasar adalah perangkat yang digunakan, *M-Commerce* dikhususkan menggunakan perangkat *mobile*. Makalah ini membahas domain *M-Commerce* berdasarkan fakta *M-Commerce* yang akan terus berkembang.

Pembangunan aplikasi *mobile* domain *M-Commerce* seharusnya dapat lebih mudah karena berdasarkan observasi dari tiga buah aplikasi sejenis diketahui bahwa terdapat beberapa kesamaan fungsionalitas dasar, terutama apabila diperhatikan lebih jauh bahwa fungsi manajemen produk akan ada di setiap *M-Commerce* sebagai fungsi inti. Kesamaan tersebut seharusnya dapat memudahkan pembangunan aplikasi baru karena *programmer* tidak perlu memulai

[1] *Dosen Tetap Teknik Informatika UNIKOM, Universitas Komputer Indonesia, Jalan Dipatiukur Nomor 112-116 Bandung 40132 INDONESIA (tlp: 081318920636; email: adam@email.unikom.ac.id)*

[2] *Alumni Teknik Informatika UNIKOM, Universitas Komputer Indonesia, Jalan Dipatiukur Nomor 112-116 Bandung 40132 INDONESIA (tlp: 081321938232; email: arifinbardansyah@gmail.com)*

pekerjaan dari awal dengan menggunakan ulang fungsionalitas dasar tersebut. Pembangunan aplikasi dari awal dan tidak mengetahui fungsional dalam pembangunan *M-Commerce* menyebabkan lamanya pengerjaan. Ketika aplikasi terlambat dilepas di pasaran, maka akan ada kerugian. *Class library* dapat menjadi solusi atas permasalahan pembangunan *M-Commerce* tersebut dengan menyediakan fungsi yang dapat digunakan kembali.

Dalam makalah ini disampaikan tentang cara membangun *class library* menggunakan analisis *frozenspot* dan *hotspot* yang ada pada fungsi manajemen produk di *M-Commerce*. Selanjutnya, dijelaskan perancangan *class library* dengan alat pemodelan menggunakan UML. Hasil perancangan diimplementasikan dan diuji dengan pengujian unit dan pengujian integrasi, sehingga didapatkan bahwa *class library* yang dibangun dapat berjalan dan digunakan untuk membantu dalam pembangunan aplikasi *M-Commerce* dalam domain kasus manajemen produk.

II. METODOLOGI

Penelitian dalam makalah ini terbagi menjadi enam tahap dalam pembangunan *class library*, yaitu analisis domain aplikasi, analisis *frozenspot*, analisis *hotspot*, perancangan *class library*, pengujian unit, dan pengujian integrasi.

A. Analisis Domain Aplikasi

Pada tahap analisis domain aplikasi, dilakukan analisis terhadap kumpulan fungsi yang ada pada beberapa aplikasi yang memiliki domain kasus yang sama dengan domain kasus yang diteliti. Pemilihan aplikasi yang dianalisis merupakan hal penting karena kesamaan fungsi-fungsi akan menentukan *frozenspot* dan *hotspot* yang dibentuk.

B. Analisis Frozenspot

Setelah melakukan analisis mendalam terhadap fungsi-fungsi yang ada pada tiga *M-Commerce* dipilihlah fungsi-fungsi yang merupakan irisan dari fungsi-fungsi yang ada pada tiga *M-Commerce* tersebut. Tentunya pada makalah ini fokus analisis adalah domain kasus manajemen produk.

C. Analisis Hotspot

Dari *frozenspot* yang terpilih, dibentuklah *hotspot* yang merupakan titik fleksibilitas yang ada di setiap aplikasi *M-Commerce* pada tahap analisis domain aplikasi. Selain itu, dilakukan pembuatan kartu *hotspot* untuk memberikan gambaran yang lebih jelas tentang fungsi-fungsi terpilih akan diimplementasikan di dalam *class library*.

D. Perancangan Class Library

Setelah memilih *frozenspot* dan *hotspot*, langkah berikutnya adalah melakukan perancangan *class library*. Perancangan ini menggunakan *class diagram* dan *package diagram*. Selain itu, juga alur kerja antar *method* yang ada pada *class-class* yang ada juga dirancang di dalam tahap ini.

E. Pengujian Unit

Pada tahapan ini *method-method* yang sudah dibuat kemudian diuji secara terpisah agar terlihat kesesuaian nilai atau tampilan keluaran yang diharapkan. Pengujian ini dilakukan menggunakan *compiler* yang ada pada IDE Android Studio.

F. Pengujian Integrasi

Sebagai tahapan terakhir, *method-method* yang telah dibuat kemudian dibentuk menjadi sebuah aplikasi untuk menguji kinerja mereka bersama di dalam sebuah bentuk prototipe aplikasi manajemen produk *M-Commerce*. Aplikasi yang dibangun dijalankan pada *smartphone* Android.

III. HASIL DAN PEMBAHASAN

Di bagian ini dibahas hasil dan pembahasan sesuai dengan metodologi yang telah dijelaskan sebelumnya.

A. Analisis Domain Aplikasi

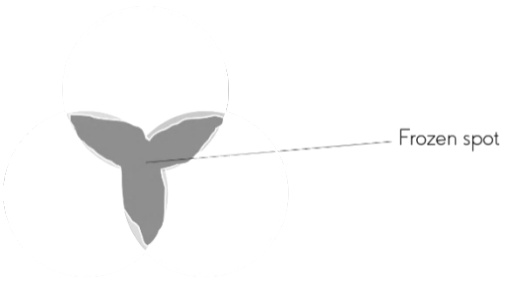
M-Commerce merupakan bentuk transaksi elektronik berbasis jaringan *wireless* sebagaimana halnya *E-Commerce* pada perangkat *mobile* [3]. *M-Commerce* mudah diakses karena menggunakan *smartphone* yang sudah *terkoneksi* internet. *M-Commerce* menjadi alternatif bahkan menjadi pilihan utama bagi sebagian besar orang dengan mobilitas tinggi, memiliki kebutuhan yang banyak dan beragam, serta memiliki waktu yang relatif sempit untuk melakukan perdagangan dan transaksi secara tatap muka. *M-Commerce* digunakan untuk mempercepat transaksi dan memudahkan penggunanya dalam proses jual beli.

Analisis domain dilakukan dengan menganalisis beberapa aplikasi sejenis dengan tujuan untuk mengetahui kesamaan fungsional yang terdapat pada setiap aplikasi. Domain aplikasi *M-Commerce* merupakan aplikasi transaksi jual beli secara *online*. Tentunya, penelitian ini difokuskan pada domain kasus manajemen produk di *M-Commerce*. Penelitian ini menganalisis fungsionalitas tiga buah aplikasi yang menjadi tolok ukur dalam pembangunan *class library M-Commerce*. Aplikasi yang menjadi tolok ukur pada antara lain Lazada, Blibli, dan Bhinneka. Aplikasi tersebut memiliki domain aplikasi yang sama, yaitu *M-Commerce* model *Business to Consumer* dan dipilih berdasarkan *rating* tertinggi dan fungsional yang terdapat dalam ketiga aplikasi tersebut. Terhadap ketiga aplikasi tersebut, dianalisis fungsi yang berjalan, data yang digunakan, dan fitur yang bisa digunakan. Kesamaan fungsi aplikasi ini dimasukkan ke dalam *frozenspot*.

B. Analisis Frozenspot

Frozenspot merupakan fungsi-fungsi tetap yang terdapat pada aplikasi dengan domain kasus tertentu [4]. *Frozenspot* dapat disebut irisan dari fungsional dasar yang terdapat dalam aplikasi. Fungsi-fungsi pada setiap aplikasi pada domain aplikasi tertentu dianalisis. Hasil analisis dari *frozenspot* kemudian digunakan untuk menentukan *hotspot* yang terdapat pada aplikasi sejenis. Ilustrasi *frozenspot* ditunjukkan pada Gbr. 1.

Dari analisis fungsionalitas dari domain aplikasi terdapat kesamaan fungsional yang ditetapkan menjadi *frozenspot*. Fungsional tersebut disajikan pada Tabel I.



Gbr. 1 Ilustrasi *frozenspot.*

TABEL I
FROZENSPOT HASIL ANALISIS

Fungsi	Aplikasi
List kategori	Bhinneka, Blibli, Lazada
Search	Bhinneka, Blibli, Lazada
Penyajian produk	Bhinneka, Blibli, Lazada
Filter	Bhinneka, Blibli, Lazada
Urut	Bhinneka, Blibli, Lazada
Ganti tampilan penyajian	Bhinneka, Blibli, Lazada
Penyajian detil produk	Bhinneka, Blibli, Lazada
Image loader	Bhinneka, Blibli, Lazada
Request	Bhinneka, Blibli, Lazada

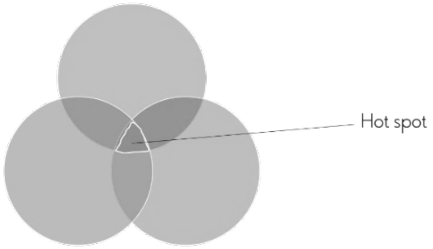
TABEL II
FITUR YANG TERDAPAT PADA *FROZENSPOT*

Fungsi	Fitur	Aplikasi
Search	Produk	Bhinneka, Blibli, Lazada
	Brand	Bhinneka, Blibli, Lazada
	Merchant	Blibli, Lazada
Filter	Sub kategori	Bhinneka, Blibli, Lazada
	Brand	Bhinneka, Blibli, Lazada
	Harga	Bhinneka, Blibli, Lazada
	Rating	Blibli, Lazada
Urut	Relevansi	Bhinneka, Blibli, Lazada
	Terbaru	Blibli
	Name A-Z	Bhinneka, Blibli, Lazada
	Name Z-A	Bhinneka
	Termurah	Bhinneka, Blibli
	Termahal	Bhinneka, Blibli
	Potongan diskon	Blibli
	Terpopuler	Blibli, Lazada
	Rating tinggi	Bhinneka, Blibli, Lazada
	Rating rendah	Bhinneka, Lazada
	Waktu pengiriman	Lazada
	Brand	Lazada
Ganti tampilan penyajian	*Grid*	Bhinneka, Blibli, Lazada
	List	Bhinneka, Blibli, Lazada
Image loader	Menampilkan gambar	Bhinneka, Blibli, Lazada
Request	*Post*	Bhinneka, Blibli, Lazada
	Get	Bhinneka, Blibli, Lazada

Fitur-fitur fungsi dianalisis untuk mendapatkan cakupan fungsi yang akan dimasukkan ke dalam *frozenspot*. Fitur tersebut disajikan pada Tabel II.

C. Analisis Hotspot

Hotspot merupakan titik fleksibilitas aplikasi yang artinya fungsional yang pasti terdapat pada setiap aplikasi pada domain kasus tertentu [4]. Analisis *hotspot* dapat diuraikan dari *frozenspot*. *Hotspot* yang diambil merupakan dasar dari aplikasi dan tidak dapat dihapus. *Hotspot* dirancang untuk lebih spesifik dan dapat disesuaikan dengan kebutuhan domain. Terdapat dua buah *method* untuk mendefinisikan *hotspot*, yaitu *method blackbox* dan *method whitebox*. *Method blackbox* yaitu *method* untuk menentukan *hotspot* berdasarkan hasil analisis dari *frozenspot*. Pada *method blackbox*, fungsional yang dapat dijadikan menjadi *hotspot* harus terdapat pada setiap aplikasi pada domain aplikasi tertentu. Sedangkan menentukan *hotspot* dengan menggunakan *method whitebox* dilakukan berdasarkan kebutuhan dari *hotspot* lainnya. Ilustrasi *hotspot* ditunjukkan pada Gbr. 2.



Gbr. 2 Ilustrasi *hotspot.*

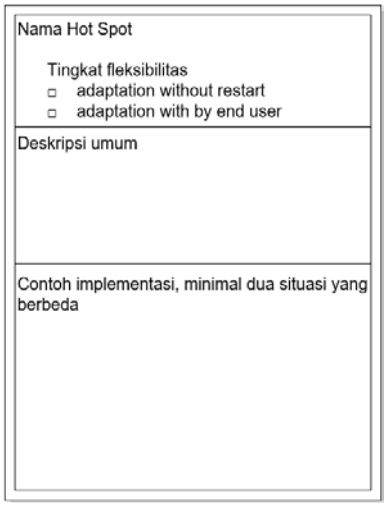
TABEL III
HASIL TERPILIH DAN FITURNYA

Fungsi	Fitur	Aplikasi
Search	Produk	Bhinneka, Blibli, Lazada
	Brand	Bhinneka, Blibli, Lazada
Filter	Sub kategori	Bhinneka, Blibli, Lazada
	Brand	Bhinneka, Blibli, Lazada
	Harga	Bhinneka, Blibli, Lazada
Urut	Relevansi	Bhinneka, Blibli, Lazada
	Name A-Z	Bhinneka, Blibli, Lazada
	Rating tinggi	Bhinneka, Blibli, Lazada
Ganti tampilan penyajian	*Grid*	Bhinneka, Blibli, Lazada
	List	Bhinneka, Blibli, Lazada
Image loader	Menampilkan gambar	Bhinneka, Blibli, Lazada
Request	*Post*	Bhinneka, Blibli, Lazada
	Get	Bhinneka, Blibli, Lazada

Pada makalah ini, penentuan *hotspot* dilakukan menggunakan kombinasi *method whitebox* dan *blackbox* karena terdapat fitur yang bukan irisan dari tiga *M-Commerce* tetapi dinilai penting untuk disolusikan di dalam *class libra*ry. *Hotspot* ditentukan dengan terdapatnya kesamaan pada ketiga aplikasi dan kebutuhan dari *hotspot* lain. Hasil analisis *frozenspot* menetapkan semua fungsi menjadi *hotspot* karena semuanya menjadi irisan fungsi dari ketiga aplikasi *M-Commerce*. Fungsional tersebut dianalisis dengan fitur pada fungsinya. Fungsional *hotspot* dan fitur ditampilkan pada Tabel III.

Setelah *hotspot* ditentukan, selanjutnya setiap fungsional *hotspot* tersebut dirancang proses fungsinya. Untuk mempermudah perancangan setiap *hotspot* dibuatkan kartu *hotspot*. Kartu *hotspot* digunakan untuk mendeskripsikan fungsionalitas dari setiap *hotspot* yang ada [4]. Kartu *hotspot* terdiri atas beberapa bagian, yaitu nama *hotspot*, derajat

fleksibilitas, deskripsi fungsional, dan perilaku *hotspot* terhadap dua situasi yang berbeda. Nama kartu *hotspot* harus menggambarkan fungsionalitas yang akan disediakan agar mudah dipahami oleh pengembang perangkat lunak. Pada derajat fleksibilitas terdapat dua pilihan, yaitu adaptasi tanpa *restart* dan adaptasi oleh *end user*. Adaptasi tanpa *restart* berarti adaptasi dari *hotspot* dapat langsung digunakan, sedangkan adaptasi oleh *end user* berarti *hotspot* tersebut harus diadaptasi oleh *end user* sebelum dapat digunakan. Kartu *hotspot* merupakan pendukung perancangan *class library*. Kartu *hotspot* berisikan tingkat fleksibilitas, deskripsi, dan contoh penerapan. Contoh kartu *hotspot* hasil analisis ditunjukkan pada Gbr. 3.



Gbr. 3 Contoh kartu *hotspot* [4].

TABEL IV
ATURAN TRANSFORMASI KARTU *HOTSPOT* [4]

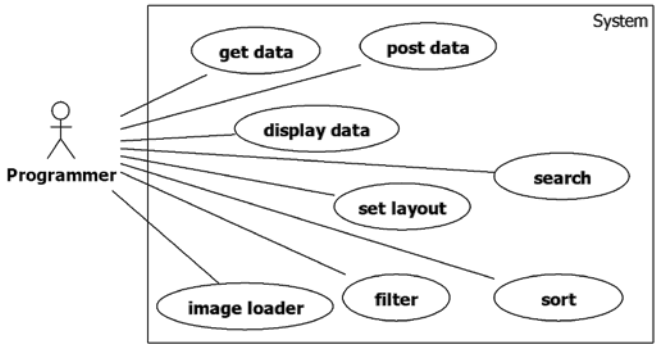
Adaptation	***Adaptation by End User***	**Transformasi Objek Model**
With restart	Tidak	Penambahan *hook method*
Without restart	Tidak	Penambahan *hook method* di *hook class* terpisah
With restart	Ya	Penambahan *hook method* dan *configuration tool*
Without restart	Ya	Penambahan *hook method* di *hook class* terpisah dan *configuration tool*

Perancangan menggunakan kartu *hotspot* memerlukan aturan untuk mengubah kartu *hotspot* menjadi rancangan. Aturan transformasi kartu *hotspot* ditampilkan pada Tabel IV.

D. Perancangan Class Library

Class Library merupakan sebuah *library* (perpustakaan) yang berisi kumpulan *class*, antarmuka, dan tipe data dalam pemrograman berbasis objek. *Class library* memiliki fungsi menyediakan akses ke fungsionalitas sistem dan dirancang untuk menjadi fondasi dari aplikasi, komponen, dan kontrol yang dibangun [5]. Tahap perancangan *class library* dibagi menjadi tiga, yaitu pemodelan *use case*, perancangan *class*, dan perancangan *package*.

Dari fitur-fitur yang sudah terbentuk dari hasil analisis *hotspot*, dibentuklah diagram *use case* seperti pada Gbr. 4.



Gbr. 4 Diagram *use case class library* manajemen produk pada *M-Commerce.*

TABEL V
KATEGORI *CLASS* HASIL ANALISIS

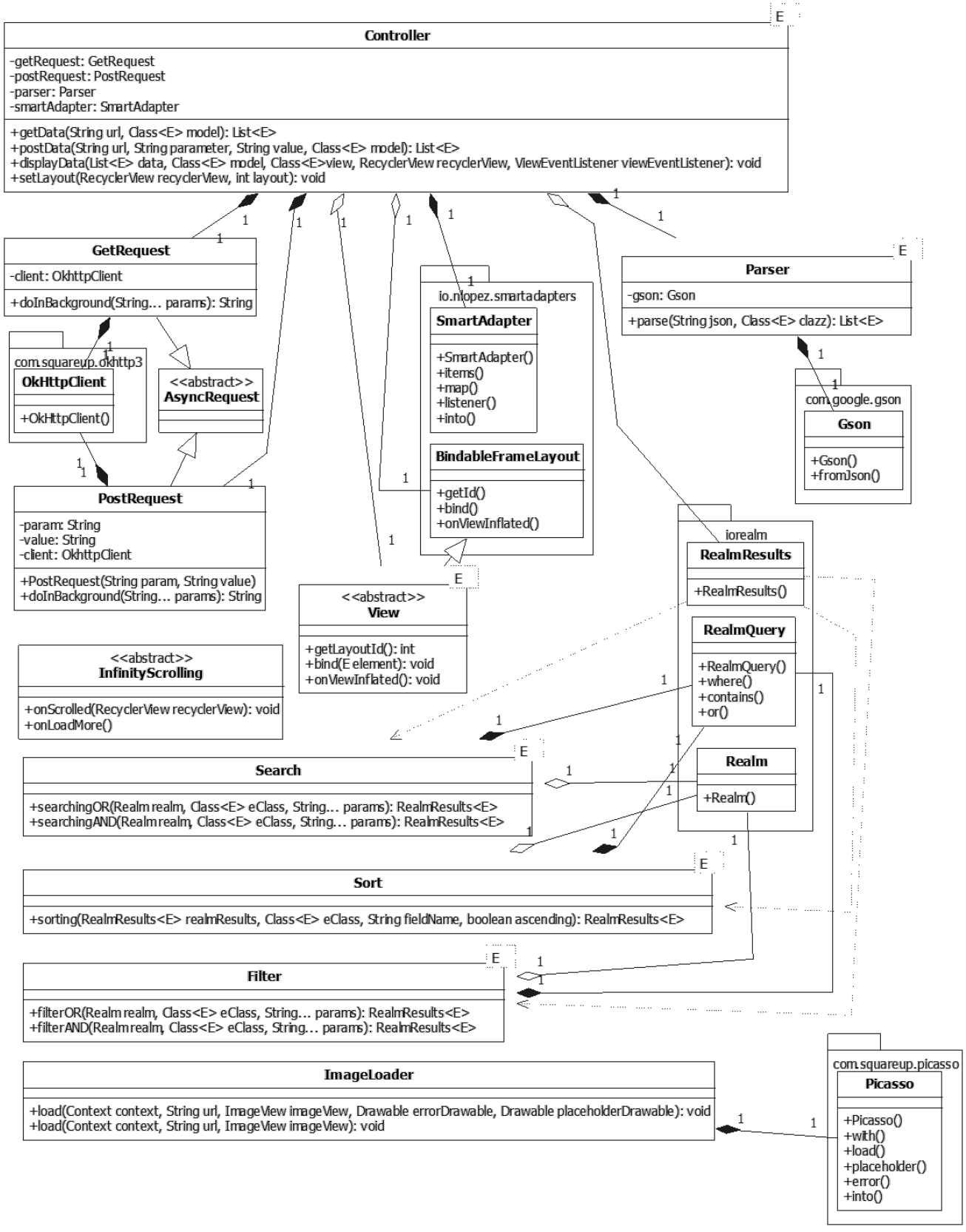
Hot Spot	***Use Case***	***Class***
List kategori	*Get data*	Controller
	Post data	
	Display data	
		<<abstract>> View
Search	*Search*	Search
Penyajian produk	*Get data*	Controller
	Post data	
	Display data	
		<<abstract>> View
Penyajian detil produk	*Get data*	Controller
	Post data	
	Display data	
		<<abstract>> View
Filter	*Filter*	Filter
Sort	*Sort*	Sort
Ganti tampilan penyajian	*Set layout*	Controller
Image loader	*Image loader*	ImageLoader
Request	*Get data*	GetRequest
	Post data	<<abstract>> AsyncRequest
		PostRequest
	Parse	Parser

Setelah memodelkan *use case*, langkah berikutnya adalah melakukan perancangan *class*. Satu *use case* pada diagram *use case* akan bisa diwakili oleh satu atau beberapa *class* yang saling berelasi. Dari *use case* yang dianalisis diperoleh *class-class* yang disajikan pada Tabel V.

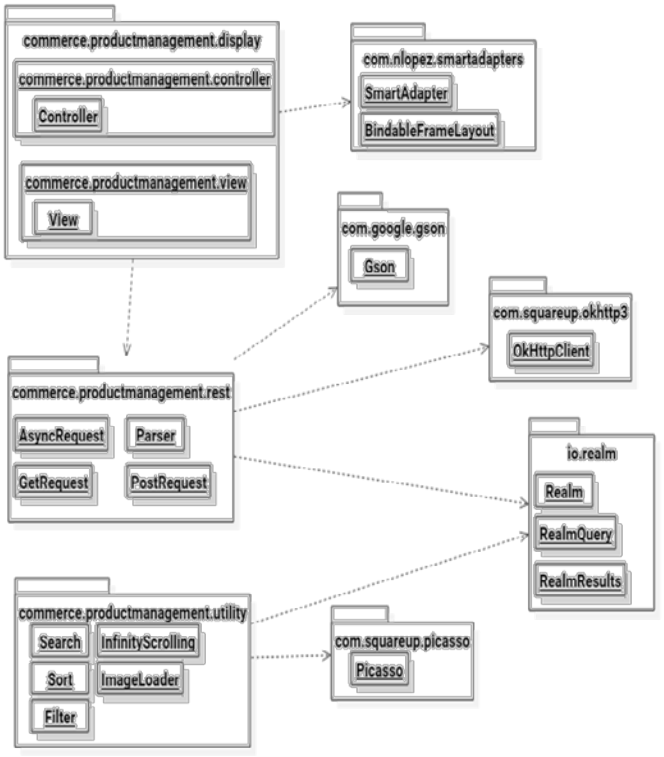
Setelah membentuk *class-class* yang akan dibangun, langkah berikutnya adalah memodelkan *class diagram*. Hasil perancangan *class diagram* ditunjukkan pada Gbr. 5.

Untuk mempermudah akses oleh pengguna *class library*, *class-class* yang sudah dirancang kemudian dikelompokkan menjadi tiga buah *package*. Pemetaan ini didasarkan pada fungsi yang dimiliki oleh setiap *class*, sehingga pengguna *class library* akan mudah memahaminya. *Package-package* tersebut adalah

1. commerce.productmanagement.display,
2. commerce.productmanagement.rest, dan
3. commerce.productmanagement.utility.



Gbr. 5 Hasil perancangan *class* manajemen produk *M-Commerce.*



Gbr. 6 *Package diagram* hasil analisis.

Setiap *package* memiliki komposisi *class-class* yang sesuai dengan tujuan fungsinya. Gbr. 6 menunjukkan komposisi masing-masing *package*.

Untuk bisa mengetahui lebih jelas tentang pengimplementasian *class library* dari hasil perancangan *class* dan *package*, *class* dan *package* ini didistribusikan melalui kanal github https://github.com/arifinbardansyah/product-management. *Library* ini dikemas menjadi *module library* dan didistribusikan ke dalam *Maven repository* agar dapat digunakan oleh *programmer M-Commerce* untuk membangun fungsi manajemen produk di platform android.

E. Pengujian Unit

Setelah dilakukan implementasi terhadap hasil perancangan *class* dan *package*, *class library* yang dibangun diuji dalam dua tahap. Tahap pertama adalah pengujian unit. Untuk melakukan pengujian unit, terlebih dahulu dibuat skenario pengujian unit yang secara umum hasilnya akan menunjukkan pesan kesalahan apabila unit yang diuji tidak berjalan dengan baik dan akan menghasilkan keluaran yang sesuai dengan yang diharapkan jika benar. Adapun detail skenario pengujian unit yang dilakukan disajikan dalam Tabel VI. Pengujian unit ini dilakukan di dalam lingkungan IDE Android Studio yang sudah terintegrasi dengan Android dan Java SDK.

Cara pengujian unit ini adalah dengan membandingkan data aktual yang diperoleh dari API menggunakan *method* pada *class library* yang telah dibuat dengan data ekspektasi. Jika data yang dibandingkan sama, maka *method* yang diuji sudah bekerja dengan benar. Hasil pengujian ditampilkan pada Tabel VII.

TABEL VI
DETAIL SKENARIO PENGUJIAN UNIT

***Class* Uji**	***Method* Uji**	**Poin Pengujian**
Controller	getData()	Jika sukses maka akan memberikan data berbentuk List sesuai ekspektasi
	postData()	Jika sukses maka akan memberikan data berbentuk List sesuai ekspektasi
	displayData()	Jika sukses maka akan menampilkan data sesuai *list* yang dimasukkan
GetRequest	doInBackground()	Jika sukses maka akan memberikan data *response* berbentuk String sesuai dengan ekspektasi
PostRequest	doInBackground()	Jika sukses maka akan memberikan data *response* berbentuk String sesuai dengan ekspektasi
Parser	parse()	Jika sukses maka akan memberikan data hasil *parsing* dari String menjadi *java object* sesuai ekspektasi
Search	searching()	Jika sukses maka akan memberikan data hasil *search* berdasarkan parameter dan sesuai dengan ekspektasi
Filter	filterOR()	Jika sukses maka akan memberikan data hasil *filter* berdasarkan parameter dan sesuai dengan ekspektasi
	filterAND()	Jika sukses maka akan memberikan data hasil *filter* berdasarkan parameter dan sesuai dengan ekspektasi
	filterBetween()	Jika sukses maka akan memberikan data hasil *filter* berdasarkan parameter dan sesuai dengan ekspektasi
Sort	sorting()	Jika sukses maka akan memberikan data hasil *sort* berdasarkan parameter yang dimasukkan dan sesuai dengan ekspektasi
ImageLoader	load()	Jika sukses maka akan memberikan tampilan gambar

TABEL VII
HASIL PENGUJIAN UNIT

***Class* Uji**	***Method* Uji**	**Kesimpulan**
Controller	getData()	*Method* sudah bekerja dengan benar
	postData()	*Method* sudah bekerja dengan benar
	displayData()	*Method* sudah bekerja dengan benar
GetRequest	doInBackground()	*Method* sudah bekerja dengan benar
PostRequest	doInBackground()	*Method* sudah bekerja dengan benar
Parser	parse()	*Method* sudah bekerja dengan benar
Search	searching()	*Method* sudah bekerja dengan benar
Filter	filterOR()	*Method* sudah bekerja dengan benar
	filterAND()	*Method* sudah bekerja dengan benar
	filterBetween()	*Method* sudah bekerja dengan benar
Sort	sorting()	*Method* sudah bekerja dengan benar
ImageLoader	load()	*Method* sudah bekerja dengan benar

F. Pengujian Integrasi

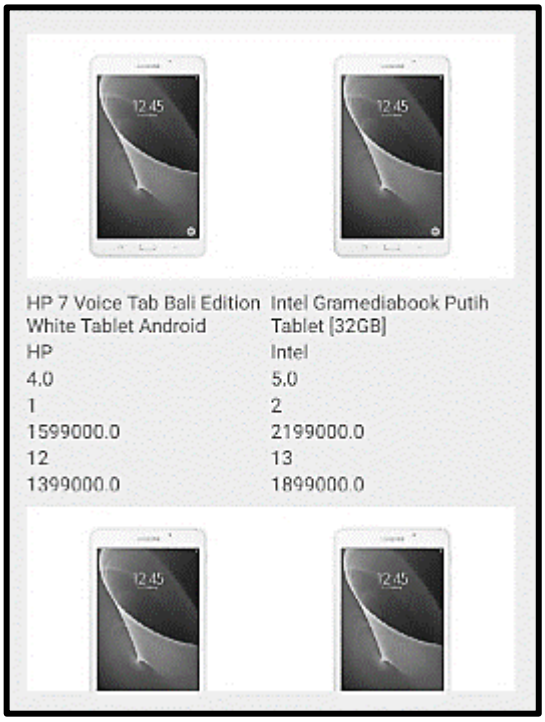
Setelah dicoba secara terpisah, modul-modul pada *class library* dicoba secara bersama untuk membentuk sebuah solusi berupa aplikasi manajemen produk *M-Commerce* di platform Android. Semua *class* diintegrasikan dan dibuat fungsionalitas *product management* aplikasi *M-Commerce*. Pengujian ini sukses jika sistem yang dibuat dapat berjalan sesuai dengan detail analisis di awal.

Untuk bisa membangun prototipe, dibutuhkan data. Sumber data yang digunakan berasal dari API yang berisi data kategori, subkategori, anak dari subkategori, dan data produk. Format API yang diakses pada prototipe memiliki format sebagai berikut.

```
https://skripsicommerce.herokuapp.com/api/product
[
{
  _id: "xx",
  product: "xx",
  category: "xx",
  image: "https://xx",
  brand: "xx",
  rating: 99,
  jumlah_review: 99,
  harga: 99,
  diskon: 99,
  harga_diskon: 99,
  deskripsi: "xx"
},

{
  ... //data kedua dan seterusnya
}
]
```

Device yang digunakan untuk menguji adalah emulator agar spesifikasi dan versi Android dapat diubah untuk menguji *class library* ini. Berikut ini adalah contoh pengujian integrasi pada fungsi penyajian produk. Untuk mendapatkan produk, digunakan *postData* lalu data tersebut disimpan sehingga terhadap data produk dapat dilakukan *query*. Dalam kasus ini, digunakan *query filter* menurut kategorisasi dengan menggunakan *class* Filter dan memanggil filterOR sehingga didapatkan data hasil *filter* berdasarkan *child* yang dipilih. Tampilan berbentuk *list category* dari awal didapatkan dari *class* Controller dengan *method* setLayout. Penyajian dapat diubah ke dalam bentuk *grid* menggunakan method setLayout dengan parameter tambahan untuk mengubah tampilan menjadi *grid*. Gambar pada produk ditampilkan dengan *method* yang terdapat pada *class* ImageLoader. Fungsi pada penyajian produk ini menggunakan beberapa *class* pada *class library*, sehingga fungsi pada *class library* saling berintegrasi. Tampilan *grid* pada produk ditunjukkan pada Gbr. 7.



Gbr. 7 Tampilan *grid* dari hasil pengujian integrasi.

Pengujian integrasi pada fungsi penyajian produk telah berfungsi dengan benar. *Class* dan *method* yang berintegrasi dapat menampilkan produk dari data API sesuai fungsi yang diuji, yaitu penyajian produk. *Class library* untuk fungsi penyajian produk telah berfungsi dengan benar. Pengujian integrasi pada setiap fungsional *hotspot* telah berfungsi menggunakan *class library* dan menjalankan fungsional tersebut. *Class library* yang diuji telah berintegrasi dan menjalankan setiap fungsionalitas pada *hotspot* dengan benar. Selain itu, untuk bisa mengetahui apakah *class library* yang telah diuji secara internal kemudian diujikan kepada lima pengembang perangkat lunak yang punya kemampuan dasar untuk menggunakan *class library* dan kemampuan untuk membangun perangkat lunak di platform Android. Dengan diberikan tugas yang sama untuk membangun prototipe seperti pada Gbr. 7 dan disertai dengan dokumentasi pada

class library, lima pengembang perangkat lunak mampu mengerjakan tugas yang diberikan.

IV. KESIMPULAN

Dari proses penelitian pembangunan *class library* dan pembangunan perangkat lunak uji, dapat disimpulkan bahwa *class library* dapat memudahkan *programmer* dalam pembangunan fungsi dasar manajemen produk pada *M-Commerce* dengan menyediakan *class library* dengan fungsi yang dapat digunakan kembali sesuai kebutuhan dalam pembangunan aplikasi *M-Commerce*.

Penelitian berikutnya perlu diarahkan untuk membangun solusi *class library* pada domain kasus lainnya yang ada pada *M-Commerce* karena pada makalah ini hanya terbatas pada domain kasus manajemen produk di *M-Commerce*.

UCAPAN TERIMA KASIH

Penelitian ini kami dedikasikan untuk kampus kami tercinta, UNIKOM yang telah memberikan kesempatan kami untuk mengembangkan keilmuan melalui penelitian ini. Selain itu, tentunya tidak lupa kami ucapkan untuk keluarga kami masing-masing. Semoga penelitian ini bisa menjadi awal untuk kami dan bagi yang membaca penelitian ini untuk semakin luas menjelajahi ilmu pengetahuan dan semakin rendah hati.

REFERENSI

[1] F. U. Noviadhista, (2015) “Indonesia pimpin pertumbuhan m-commerce ASEAN, ini rahasianya” *Kompas Online*. [Online], Available: https://www.techno.id/tech-news/indonesia-pimpin-pertumbuhan-m-commerce-asean-ini-rahasianya-150724m.html, tanggal akses: 01 Februari 2016.

[2] J. Wolf, (2015) “State of Mobile Commerce Report, Q4 2015: Leaders Widen the G” *Criteo*. [Online]. Available: http://www.criteo.com/resources/mobile-commerce-report/, tanggal akses: 01 Januari 2016.

[3] I. P. A. E. Pratama, *Mobile Commerce*, vol. 1. Bandung: Informatika, 2015.

[4] D. C. S. M. M. Fayad, R. E. Johnson, *Building Application Framework*. New York: John Wiley & Sons, 1999.

[5] Microsoft team, (2016) “Class Library". [Online]. https://msdn.microsoft.com/en-us/library/d11h6832(v=vs.71).aspx, tanggal akses: 29 April 2016.