JOINT

Journal of Informatics and Technology, Vol 2, No 1, Tahun 2013, p 52-61

http://ejournal-s1.undip.ac.id/index.php/joint

Rancang Bangun Aplikasi *Online Music Store* Dengan Menggunakan *In-App Purchase Server Product Model* Pada *Mobile Device* Berbasis iPad iOS

Dhimas Ernowo P., Ragil Saputra, S.Si, M.Cs, Priyo Sidik Sasongko, S.Si, M.Kom

Ilmu Komputer / Informatika FSM Universitas Diponegoro

lordtatsuma@gmail.com, ragil.saputra@undip.ac.id, priyoss@undip.ac.id

Abstrak

iTunes Store adalah sebuah toko online yang diciptakan *Apple Inc.* sebagai wadah untuk menjual konten - konten multimedia, *iTunes Store* tersedia untuk semua pengguna yang sebelumnya telah mengunduh aplikasi pemutar multimedia *iTunes.* Dengan *iTunes Store*, pembuat konten multimedia menaruh konten multimedia di dalam *store* untuk menjualnya, pembayarannya pun sudah diatur dengan *AppleID* yang berhubungan langsung dengan kartu kredit. Permasalahannya adalah, untuk pembuat konten multimedia, *Apple Inc.* belum membuka *iTunes Store* di semua negara. Karena masalah ini, dengan menggunakan Modul *In-App Purchase* yang tersedia pada *iOS SDK, model process sequential linear* dan bahasa pemodelan *Unified Modeling Language*, dibangun sebuah aplikasi *Online Music Store* berbasis iPad iOS yang bersifat *worldwide.*. Model *Server Product* dan *Web Service* digunakan dalam melakukan rancang bangun aplikasi, akan dijelaskan pula hubungan dan tugas - tugas dari tiga elemen penting dari *In-App Purchase Server Product Model*, yaitu *App Store, Application,* dan *Developer Server.* Sehingga dihasilkan aplikasi *Online Music Store* yang tidak terbatas oleh dukungan *Apple Inc.* terhadap *iTunes Store.*

Kata kunci : *Online Music Store, In-App Purchase, Mobile Device, iPad iOS.*

Abstract

iTunes Store is an online store which is created by Apple Inc. for selling multimedia contents. iTunes store is available for every user who had iTunes, Apple's multimedia player, on their computers. With iTunes Store, the multimedia content maker upload their content to the store so the user could buy their products. The Apple ID which obligate to the user for buying the contents in the iTunes Store, is connected to their credit cards. However, for multimedia content maker, the Apple Inc. iTunes Store service is not available yet in all country. Because of this problem, by using In-App Purchase module which is available on iOS SDK, sequential process model and Unified Modeling Language, the writer try to build an Online Music Store that runs on iPad iOS and available worldwide. Server Product Model and Web Service are used for building the application, also, the communications and tasks from three key element on In-App Purchase Server Product Model, which is App Store, Application and Developer Server would be explained. The result of this is an Online Music Store application which is not limited to Apple Inc's iTunes Store Support.

1. Pendahuluan

Di zaman digital seperti sekarang, telah terjadi perubahan pesat dalam sistem penjualan musik yang sebelumnya dilakukan secara konvensional dengan membeli musik ke toko fisik, menjadi penjualan musik yang dilakukan secara online, hal ini secara tidak langsung memaksa konsumer untuk berpindah untuk mengatur koleksi musik mereka ke dalam perpustakaan digital (*digital libraries).* Konsumer mengunduh lagu-lagu dan membagi koleksi musik mereka melalui banyak perangkat dan jaringan [2]. Salah satu toko musik online yang sangat terkenal adalah *Apple iTunes Store.*

Apple iTunes Store adalah sebuah layanan berbasis web yang menjual lagu, film, serial tv, dan banyak konten lain yang berhubungan dengan multimedia, dibuat oleh *Apple inc* dan tersedia untuk semua pengguna yang sebelumnya telah mengunduh iTunes. Layanan *Apple iTunes Store* dapat digunakan sebagai o*nline store* sehingga pengguna tidak perlu pergi ke toko penjualan musik fisik untuk dapat menikmati konten berbasis multimedia. Pengguna hanya perlu membuat Apple ID yang dapat dibuat secara gratis untuk dapat mengakses layanan yang ditawarkan oleh *Apple iTunes Store*

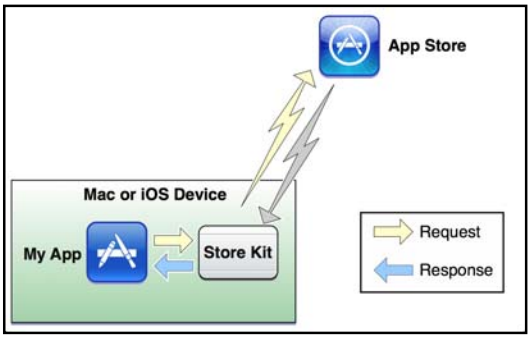
Keyword : Online Music Store, In-App Purchase, Mobile Device, iPad iOS.

In-App Purchase memungkinkan suatu aplikasi untuk menjual fitur tambahan yang bisa dibeli oleh pengguna secara terpisah [3], dalam permasalahan ini fitur yang dimaksud adalah konten-konten multimedia, sehingga untuk menikmati konten-konten yang tersedia di server, pengguna diharuskan melakukan transaksi terlebih dahulu. Fitur, layanan, atau fungsi tambahan yang ditawarkan di dalam suatu aplikasi dapat dibeli dengan menggunakan Apple ID [7].

Mengingat *Apple* belum membuka layanan *Apple iTunes Store* nya di seluruh dunia, sebagai contohnya Indonesia, akibatnya pengguna perangkat *Apple* yang ada di Indonesia tidak bisa mengakses *Apple iTunes Store* dan tentu saja tidak dapat melakukan pembelian konten-konten yang dijual di *Apple iTunes Store*. Dari permasalahan yang sudah dijelaskan sebelumnya, akan dilakukan rancang bangun aplikasi yang mempunyai fungsi yang sama dengan *Apple iTunes Store* dalam hal pembayaran, hal ini memungkinkan dengan menggunakan modul *In-App Purchase* yang ditawarkan iOS Framework.

1. Dasar Teori

2.1. In-App Purchase



Gambar 2.1 Model *In-App Purchase* [1]

In-App Purchase memungkinkan pengembang untuk menambahkan sebuah *store* secara langsung ke dalam aplikasi. Pengimplementasian *In-App Purchase* menggunakan *framework Store Kit*. *Store kit* menghubungkan aplikasi dengan *App Store* untuk mengatur proses pembayaran secara aman dari pengguna [1]. Modul *In-App Purchase* berpusat pada *framework Store Kit*, semua fungsi-fungsi vital mengenai pembayaran di atur oleh *framework* ini, *Store Kit* berkomunikasi dengan *App Store* untuk mendapatkan produk yang ditawarkan dan aplikasi menampilkannya ke pengguna dan pengguna berhak untuk membeli konten tersebut. *Store Kit* digunakan untuk mengumpulkan informasi pembayaran dari pengguna ketika pengguna ingin melakukan transaksi. Gambar 2.1 menunjukan model dasar dari *In-App Purchase*.

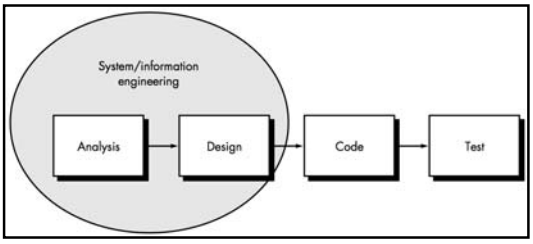
2.2. *iOS*

iOS adalah sistem operasi yang berjalan di iPhone, iPod touch dan iPad. Sistem operasi ini mengatur perangkat keras dari perangkat dan menyediakan teknologi yang dibutuhkan untuk menjalankan aplikasi [6]. iOS dibuat menggunakan konsep yang sama seperti dengan sistem operasi yang ada pada Mac OS X, kebanyakan perangkat dan teknologi yang digunakan pada *platform* ini pun terdapat pula pada Mac OS X. Namun dari semua kesamaan dengan Mac OS X, untuk mengembangkan aplikasi iOS tidak memerlukan pengalaman yang sama pada saat mengembangkan aplikasi Mac OS X. iOS mempunyai *Software Development Kit (SDK)* sendiri untuk mengembangakan aplikasinya, SDK ini menyediakan semua yang diperlukan untuk memulai mengembangkan aplikasi dari iOS.

iOS SDK berisi perangkat dan antarmuka yang dibutuhkan untuk membangun aplikasi, menginstall, menjalankan dan melakukan pengujian terhadap aplikasi yang akan dibuat. Aplikasi dibuat dengan menggunakan *iOS system framework* dan bahasa pemrograman *Objective-C.* Tidak seperti aplikasi berbasis web, aplikasi *iOS* terpasang di perangkat dan selalu tersedia untuk pengguna, bahkan ketika perangkat berada dalam *Airplane mode*.

2.3. Model *Linier Sequential*

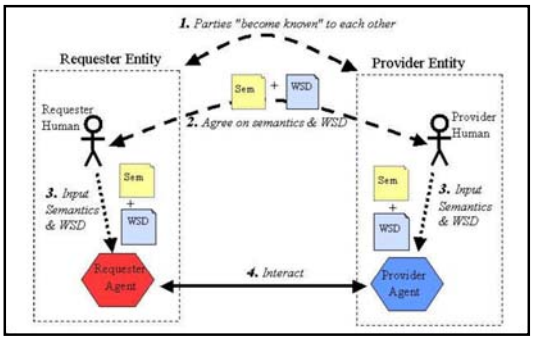
Permodelan pengembangan perangkat lunak sangat beragam, salah satunya adalah *linear sequential*. Model *linear sequential* menyarankan pendekatan yang sistematis dan berurut untuk mengembangkan perangkat lunak yang dimulai dari tingkatan sistem dan berjalan melewati *analysis, design, coding, testing* dan *support* [5]. Model *Linear Sequential* dapat dilihat pada gambar 2.2.



Gambar 2.2 Model *Linier Sequential* [5]

2.4 *Web Service*

Web Service atau layanan web adalah sebuah sistem perangkat lunak yang didesain untuk mendukung interaksi yang dapat melakukan



pertukaran informasi antar mesin melalui jaringan [4].

Gambar 2.3 Proses Umum Untuk Memulai Sebuah *Web Service*

Web Service mempunyai sebuah *interface* dengan format yang dapat dibaca oleh mesin dengan nama *Web Services Descriptor Language* atau yang biasa disebut WSDL. WSDL mendefinisikan layanan dan bantuan apa yang ada dalam sebuah *web service* dan bagaiman cara memanggilnya. Sistem lain berinteraksi dengan *web service* menggunakan pesan SOAP. SOAP merupakan kepanjangan dari *Simple Object Access Protocol*, adalah sebuah spesifikasi protokol untuk pertukaran informasi terstruktur di dalam pengimplementasian layanan web di dalam sebuah jaringan. SOAP menspesifikasikan bagaimana seharusnya sebuah *request* diformat sebelum dikirim ke *server*, dan bagaimana seharusnya *server* memformat respon yang akan dikirimkan kembali ke *client*. Pesan antara *client* dengan *server* tadi dibawa menggunakan HTML dengan XML [4].

2. Analisis dan Perancangan

3.1. Deskripsi Sistem

Aplikasi *Online Music Store* berbasis iPad iOS merupakan aplikasi berbasis *mobile* yang bertujuan menghasilkan sebuah toko yang beroperasi secara online yang menjual konten multimedia dan menjalankan konten multimedia yang telah dibeli.

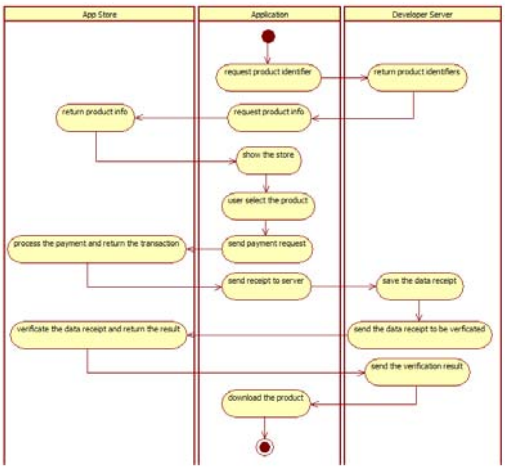
Aplikasi *Online Music Store* yang dibangun mirip dengan layanan *iTunes Store* yang disediakan oleh *Apple Inc.*, namun terbatas pada jumlah artis yang kontennya akan dijual.

Cara kerja sistem pada dasarnya melakukan *data fetching* via *web service* antara *developer server*, *app store* dan aplikasi itu sendiri dengan menggunakan *eXtensible Markup Language* (XML), sedangkan mengenai penanganan transaksi produk ditangani sepenuhnya oleh modul *In-App Purchase.* Model yang digunakan untuk pengantaran produk adalah *Server Product Model*, alasan dalam memakai model ini diantara lain adalah :

1.) *Server Product Model* memungkinkan melakukan pertambahan data dan manajemen data tanpa harus mengubah kode dari program, dilakukan penambahan data, harus dilakukan *re-release* aplikasi.

2.) *Server Product Model* menggunakan *developer server* dalam *Workflow* dibagi menjadi 3 bagian, *App Store*, Aplikasi, dan Server. Pertama-tama untuk mendapatkan produk mana saja yang tersedia untuk dijual aplikasi mengirim *request* ke server untuk meminta list identifier, list identifier adalah sekumpulan kode pengenal suatu produk, setiap kode pengenal mewakilkan suatu produk yang akan dijual. Setelah list identifier diterima dari server, aplikasi lalu mengirimkan list tadi ke *app store* untuk meminta informasi produk yang akan dijual, *app store* akan mengembalikan informasi produk sesuai dengan identifier yang tadi diberikan. Setelah informasi produk diterima, aplikasi menampilkan produk - produk yang dijual. Apabila pengguna ingin membeli suatu produk, aplikasi akan mengirimkan request pembelian / pembayaran ke *app store*, dan apabila transaksi berhasil maka *app store* akan mengembalikan suatu *receipt* ke aplikasi, *receipt* adalah bukti transaksi yang dihasilkan bilamana transaksi sudah berhasil dilakukan, aplikasi lalu meneruskan *receipt* tadi ke server, lalu server mengirim *receipt* ke *app store* untuk mengkonfirmasi apakah *receipt* bernilai

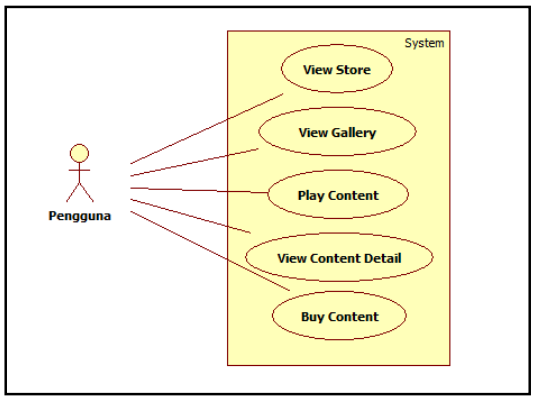
valid. Receipt yang akan dikirim ke *app store* harus ter-*encode* base64. Seteleh *app store* memberi tahu bahwa *receipt* valid, barulah server mengirimkan produk ke dalam aplikasi. Untuk lebih jelasnya dapat dilihat pada Gambar 3.1



Activity Diagram Workflow In-app Purchase Server Product Model

3.2 *Use Case Diagram*

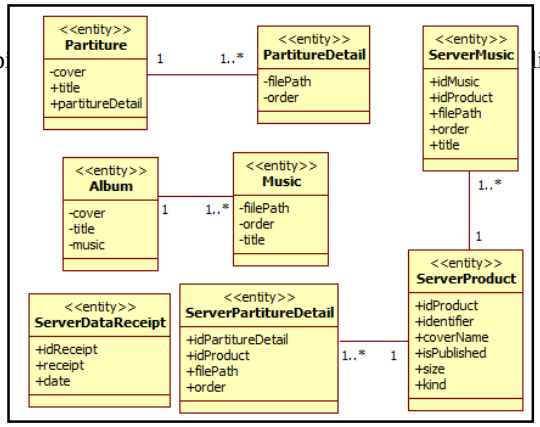
Use Case Diagram digunakan untuk menjelaskan fungsi - fungsi pada sistem. *Use Case* disusun atas aktor, *use case*, dan hubungan diantara keduanya. *Use case diagram* sistem lebih jelasnya dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram *Use Case*

3.3 Perancangan *Database*

Dalam pengembangan perangkat lunak, data perlu disimpan untuk kemudian diolah menjadi sebuah informasi, oleh karena itu perlu dirancang sebuah *database* untuk menggambarkan entitas (objek data) dan relasi antar sesamanya. *Database* yang digunakan adalah *relational database* sedangkan dalam analisis dan perancangan digunakan permodelan berbasis objek, untuk itu dilakukanlah *Object Relational Mapping*, untuk mengubah konsep objek menjadi *relational*.



Gambar 3.3 *Association* pada *Entity Class*

Pada *entity class* terdapat hubungan antar kelas berupa asosiasi yang mengandung *multiplicity*. Hubungan yang terdapat pada *entity class* digambarkan pada Gambar 3.3.

Penjelasan hubungan *association* pada Gambar 3.3. dijelaskan sebagai berikut :

1.) *Class* **Partiture** dengan *Class* **PartitureDetail** berhubungan

2.) *Class* **Album** dengan *Class* **Music** berhubungan *one-to-many*, yang artinya satu *Album* minimal mempunyai satu *Music*.

3.) *Class* **ServerProduct** dengan *Class* **ServerMusic** berhubungan *one-to-many*, yang artinya satu **ServerProduct** minimal mempun

4.) *Class* **ServerProduct** dengan *Class* **ServerPartitureDetail** berhubungan *one-to- many*, yang artinya satu **ServerProduct** minimal mempunyai satu **ServerPartitureDetail**.

3. Implementasi dan Pengujian

4.1. Implementasi Basis Data

Basis data pada aplikasi *Online Music Store* terbagi menjadi dua bagian, sisi aplikasi dan sisi server, keduanya menggunakan *relational database*. Kedua basis data dihubungkan dengan

XML dalam melakukan komunikasi. Dalam pengimplimentasiannya aplikasi *Online Music Store* menggunakan *SQLite* sebagai basis data pada sisi aplikasi dan MySQL sebagai basis data pada sisi server.

Implementasi basis data dari aplikasi *Online Music Store* dapat dilihat pada Tabel 4.1 sampai dengan Tabel 4.8.

Tabel 4.1 Keterangan *Field* Tabel Album Pada Sisi Aplikasi

Nama *Field*	Tipe Data	Keterangan
objectID	int(11)	Primary key tabel.
cover	varchar(150)	Alamat berkas yang berfungsi sebagai cover dari suatu album di dalam sistem.
title	varchar(50)	Judul dari album

Tabel 4.2 Keterangan *Field* Tabel Music Pada Sisi Aplika

Nama *Field*	Tipe Data	Keterangan
objectID	int(11)	Primary key tabel.
filePath	varchar(150)	Alamat berkas konten.
title	varchar(50)	Judul dari musik
order	int(3)	Order *playlist* dari musik

Tabel 4.3 Keterangan *Field* Tabel Partiture Pada Sisi Ap

Nama *Field*	Tipe Data	Keterangan
objectID	int(11)	Primary key tabel.
cover	varchar(150)	Alamat berkas yang berfungsi sebagai cover dari suatu judul partiture di dalam sistem.
title	varchar(50)	Judul dari partiture.

Tabel 4.4 Keterangan *Field* Tabel PartitureDetail Pada S

Nama *Field*	Tipe Data	Keterangan
objectID	int(11)	Primary key tabel.
filePath	varchar(150)	Alamat berkas konten.
order	int(3)	Order dari partiture

Tabel 4.5 Keterangan *Field* Tabel product Pada Sisi *Developer Server*

Nama *Field*	Tipe Data	Keterangan
idProduct	int(11)	Primary key tabel.
identifier	varchar(50)	Tanda pengenal produk
coverName	varchar(100)	Alamat berkas *cover* dari produk
isPublished	tinyint(1)	Sebuah *flag* apakah produk boleh dipublikasikan ke *store* atau tidak.
size	float(10)	Ukuran dari produk dalam MB
kind	varchar(10)	Jenis produk

Tabel 4.6 Keterangan *Field* Tabel dataReceipt Pada Sisi *Developer Server*

Nama *Field*	Tipe Data	Keterangan
idReceipt	int(11)	Primary key tabel.
receipt	varchar(3000)	String *transaction receipt* yang sudah di-*encode* menggunakan metode *base64*
date	date	Tanggal tupel dibuat

Tabel 4.7 Keterangan *Field* Tabel music Pada Sisi *Developer S*

Nama *Field*	Tipe Data	Keterangan
idMusic	int(11)	Primary key tabel.

Nama *Field*	Tipe Data	Keterangan
idProduct	int(11)	*Foreign key* dari tabel Produk
title	varchar(50)	Judul dari musik
order	int(3)	Urutan musik dalam suatu album
filePath	varchar(100)	Alamat berkas

Tabel 4.8 Keterangan *Field* Tabel partitureDetail Pada Sisi *Developer Server*

Nama *Field*	Tipe Data	Keterangan
idPartiture Detail	int(11)	Primary key tabel.
idProduct	int(11)	*Foreign key* dari tabel Produk
order	int(3)	Urutan detail partitur dalam suatu kumpulan partitur
filePath	varchar(100)	Alamat berkas

4.2 *Class Implementation*

Aplikasi *Online Music Store* mempunyai 3 komponen yang saling bekerja sama, yaitu aplikasi itu sendiri, *App Store*, dan *Developer Server*. Implementasi kelas berdasarkan urutan *workflow* aplikasi *Online Music Store* yang pada bab sebelumnya telah dijelaskan, dikelompokkan dan dijelaskan lebih lanjut sebagai berikut.

4.2.1 *Server Initialization*

App Store berguna sebagai *payment method* dan bertanggung jawab atas produk - produk apa saja yang boleh dijual dalam sebuah aplikasi, oleh karena itu, aplikasi dan produk - produk yang ingin dijual di dalam *In-App Purchase* perlu didaftarkan ke dalam *App Store* terlebih dahulu. Sebelumnya, untuk dapat menjual aplikasi berbasis *Apple*, pengembang wajib mempunyai *Apple Developer ID* yang dapat dibuat pada situs resmi *Apple*, jenis keanggotaan terbatas hanya untuk satu tahun dan berbayar. Bila pengembang sudah memiliki *Apple Developer ID*, pengembang mendapatkan akses ke *iTunes Connect*, disinilah program yang akan dijual ke dalam *App Store* diatur, seperti penentuan *product identifier*, pemilihan harga, penambahan deskripsi produk, dan lain lain. Pengembang dapat men-*submit* aplikasi yang akan dijual dan menunggu respon dari *apple review team* yang kemudian akan ditentukan apakah aplikasi yang dibuat layak masuk *App Store* atau tidak dan perlu revisi. Fungsi lain dari *iTunes Connect* adalah untuk mendaftarkan produk - produk *In-App Purchase* yang akan dijual dalam suatu aplikasi. Setiap produk *Apple* sebuah ditandai dengan *identifier* unik, *identifier* inilah yang digunakan untuk berkomunikasi antara aplikasi atau produk dengan *iTunes Connect* dalam melakukan *fetching* list produk. Selain itu, perlu dipersiapkan produk yang akan dijual dan produk tersebut dimasukkan ke dalam *developer server*, *developer server* yang dipakai pada penelitian ini berada di Houston, Texas. Hal ini disebabkan pada saat *apple review team* mengetest aplikasi yang akan dimasukkan ke *app store*, melakukan *fetching*, dan ternyata *response* dari *developer server* cukup lama, *apple review team* akan menganggap ini *bug* dan aplikasi yang akan dimasukkan ke *app store* pun akan otomatis ditolak. Karena itu *developer server* ditempatkan satu negara dengan *Server App Store.*

4.2.2 *Request* list *identifier.*

Workflow Online Music Store akan dimulai apabila pengguna masuk ke dalam menu *Store*, aplikasi akan mengirimkan *request* ke *developer server.*

Kode 4.1 Alamat *Request* yang Dikirimkan dari Aplikasi

```
http://RootServer/onlinemusicstore/omsBusinessObject.php?request=getAllProducts
```

4.2.3 *Response from developer server*

IAPHelper pada sisi aplikasi memanggil **OmsXmlCreator** pada sisi *developer server* yang akan mengembalikan *xml string* berupa *identifier* produk yang telah didaftarkan di dalam *iTunes Connect* dan beberapa informasi tambahan dari produk yang tersedia untuk ditawarkan kepada pengguna. *list* ini kemudian disimpan di dalam *instance variable* bernama productIdentifiers.

Kode 4.3 *Response* dari *developer server*

```
<outputs>
  <product>
    <idProduct>1</idProduct>
<identifier>onlinemusicstore.firstal
bum</identifier>

<coverName>http://RootServer/onlinem
usicstore/thumbnails/or1.jpg</coverN
ame>
    <isPublished>1</isPublished>
      <size>25.7</size>
    <kind>music</kind>
  </product>
</outputs>
```

4.2.4 *Request to App Store*

Response yang dikembalikan dari *developer server* berupa *list identifier* produk yang telah didaftarkan di *App Store.* Kemudian *list identifier* ini dikirimkan ke *App Store* untuk mendapatkan info produk. **IAPHelper** pada sisi aplikasi akan mengirimkan *request* dengan parameter *list identifier* ke *App Store.*

4.2.5 Menampilkan Informasi Produk

App Store akan mengembalikan informasi produk kepada aplikasi, aplikasi akan secara otomatis menangkap *response* dari *App Store,* dan menjalankan *method delegate didReceiveResponse.* Informasi produk akan disamakan dan digabungkan terhadap informasi produk yang sebelumnya didapatkan dari *developer server* yang kemudian akan disimpan pada objek **Product**.

didReceiveResponse lalu akan mengirimkan sebuah notifikasi yang mengabarkan bahwa produk sudah terunduh. **UIStore** kemudian akan memanfaatkan objek - objek **Product** dan akan menampilkan informasi yang ada untuk kemudian disajikan kepada pengguna.

4.2.6 Pemilihan Produk

UIStore akan menampilkan produk yang tersedia untuk dijual, apabila pengguna melakukan *tapping* pada salah satu produk sebuah *pop-up* yang menampilkan detail dari produk akan muncul.

Sistem juga akan mengecek, apakah produk yang dipilih sudah ada di sistem atau belum, bila belum akan ditampilkan tombol “Buy”, dan tombol “Purchased” apabila produk sudah ada di dalam sistem.

4.2.7 Pembelian Produk

Pada saat detail produk ditampilkan dan tombol “Buy” ditekan **IAPHelper** akan dipanggil untuk memulai proses pembayaran, proses pembayaran sudah sepenuhnya diatur oleh *StoreKit framework*, pengguna akan diminta untuk memasukkan *AppleID* dan *password* untuk dapat membeli sebuah produk.

4.2.8 Pengiriman *Request* Pembayaran

Ketika metode pembelian dilakukan, objek *SKPayment* dan *SKPaymentQueue* dari *StoreKit framework* yang berada pada aplikasi akan berkomunikasi dengan *App Store*, mengirimkan sebuah *product identifier*, dan kemudian *Response* dari *App Store* akan secara otomatis ditangkap oleh *updatedTransaction delegate method.*

Metode *updatedTransactions* akan memisahkan *response* berdasarkan *transactionState*, jika *transactionState* menunjukan nilai *SKPaymentTrasactionPurchase* atau jika produk berhasil dibeli, metode *completeTransaction* akan dipanggil. Jika *transactionState* menunjukan nilai *SKPaymentTransactionFailed* atau jika produk gagal dibeli, metode *failedTransaction* akan dipanggil. Jika *transactionState* menunjukan nilai *SKPaymentStateRestored* atau jika produk berhasil diunduh kembail, metode *restoreTransaction* akan dipanggil.

4.2.9 Pengiriman *Transaction Receipt* oleh Aplikasi

Transaction Receipt yang sebelumnya telah didapatkan dari *App Store* kemudianakan diteruskan ke *Developer Server* untuk dilakukan proses pencatatan dan akan dijadikan sebagai bukti pembayaran. Data *transaction receipt* yang dikirimkan ke *developer server* sebelumnya harus di-*encode* dengan metode *encode base64*. Hal ini dilakukan karena nantinya *developer server* akan mengirimkan *transaction receipt* kepada *App Store* untuk kemudian dilakukan pengecekan apakah *transaction receipt* bernilai *valid. App Store* meminta data *transaction receipt* yang dikirimkan harus ter-*encode base64.*

Kode 4.11 Sebuah String yang Di-*Encode* dengan *Base64*

```
Sebelum : YourTextHere
Sesudah : WW91clRleHRIZXJl
```

Dalam kelas **IAPHelper** *data receipt* akan di *encode* dengan metode *base64*, kemudian V**erifyReceipt** yang ada pada *developer server*

akan mengirimkan *data receipt* hasil *encode* kepada *App Store* untuk dilakukan validasi.

4.2.10 Server Menyimpan dan Melakukan Verifikasi *Receipt*

Transaction receipt yang telah di-*encode* dengan metode *base64* akan disimpan pada *server developer*, kemudian dilakukan verifikasi, apakah *receipt* yang dikirimkan dari aplikasi bernilai *valid*. Kelas **VerifyReceipt** pada sisi *developer server* akan mengirimkan *transaction receipt* menggunakan objek JSON dengan kunci bernama "receipt-data" dan dikirimkan melalui *HTTP Post Request* kealamat https://sandbox.itunes.apple.com/verifyReceipt.

Kode 4.13 Kode JSON yang Dikirimkan Ke *App Store*

```
{
      "receipt-data" :
"(receipt bytes here)"
}
```

Response dari *App Store* berupa objek JSON dengan dua kunci, yaitu "receipt" dan "status". Nilai yang ada pada kunci "status" bila bernilai 0 (nol) menjelaskan bahwa *transaction receipt* yang dikirmkan bernilai *valid*, selain itu maka *transaction receipt* tidak *valid.*

Kode 4.14 Kode JSON yang Dikembalikan dari *App Sto*

```
{
      "status" : 0,
      "receipt" : { (receipt here) }
}
```

4.2.11 Pengunduhan Produk

Langkah terakhir adalah, aplikasi mengunduh produk. *Developer Server* mengembalikan *Response* ke aplikasi yang berisi *Response* yang didapat pada saat proses verifikasi *transaction receipt.* Aplikasi akan mengunduh produk yang dipilih pengguna apabila *Response* yang didapat bernilai *valid.* Kelas **IAPHelper** mengunduh produk yang sebelumnya dipilih oleh pengguna. Kelas ini berhubungan dengan kelas **OmsXmlCreator** pada *developer server* untuk mengambil data musik atau detail partitur yang dibutuhkan.

Kode 4.15 Alamat *Request* Untuk Mengunduh Berkas

```
// Untuk berkas musik
http://RootServer/onlinemusicstore/o
msBusinessObject.php?request=getMusi
cs&idProduct=?

// Untuk berkas partiture
http://RootServer/onlinemusicstore/o
msBusinessObject.php?request=getPart
itureDetails&idProduct=?
```

Kode 4.16 *Response* dari *Developer Server* Untuk *Request* Un

```
<outputs>
  <music>
   <idMusic>1</idMusic>
    <idProduct>1</idProduct>

<filePath>http://RootServer/onlinemu
sicstore/musics/1.mp3</filePath>
    <order>1</order>
   <title>Cinta kan membawamu
kembali disini</title>
  </music>
  <music>
   <idMusic>2</idMusic>
   <idProduct>1</idProduct>

<filePath>http://RootServer/onlinemu
sicstore/musics/2.mp3</
   filePath>
   <order>2</order>
   <title>Diary</title>
  </music>
</outputs>
```

Setelah semua berkas terunduh, objek dari kelas **AppDelegate** kemudian dipanggil untuk melakukan pencatatan dan penyimpanan ke dalam aplikasi.

4. Kesimpulan dan Saran

5.1. Kesimpulan

Kesimpulan yang dapat diambil dari penyusunan tugas akhir ini adalah dengan memanfaatkan teknologi *mobile device, web service* dan modul *In-App Purchase Server Product Model* yang ditawarkan *Apple*, dihasilkan aplikasi *Online Music Store* pada *Mobile Device* berbasis iPad iOS. Aplikasi ini dapat melakukan penjualan konten multimedia tidak hanya di negara yang didukung oleh *iTunes Apple Store*, akan tetapi di seluruh

dunia. Aplikasi ini juga dapat memainkan dan/atau menampilkan konten multimedia yang dijual di dalam *store*. Sedangkan kendala yang dihadapi dalam pembuatan tugas akhir ini adalah, pengujian tidak bisa dilakukan melalui simulator, simulator tidak bisa membaca resi yang didapat dari *App Store* karena *App Store* membutuhkan *device identifier* untuk melakukan proses transaksi, karena itu dibutuhkan *device* aslinya. Selain itu karena menggunakan *device* aslinya, proses *install* harus dilakukan setiap kali hendak melakukan pengujian.

5.2. Saran

Aplikasi *Online Music Store* masih dapat dilakukan beberapa pengembangan lebih lanjut. Salah satunya dengan menampilkan konten multimedia selain musik dan partitur. Selain itu aplikasi ini dapat dikembangkan menjadi tidak hanya berorientasi kepada satu artis namun banyak artis.

REFERENSI

[1] Apple Inc. 2012. "*StoreKitGuide"*. Apple Inc.

[2] Chorianopoulus, Konstantinos. *et al*. 2005. "*Cross Media Digital Rights Management for Online Stores"*. Imperial College London.

[3] DeVoe, Jiva. 2010. "*Cocoa Touch™ for iPhone OS 3"*. Indianapolis. Willey Publishing, Inc.

[4]Haas, Hugo. Brown, Allen. 2004. "*Web Service Arcitecture"*. W3C.

[5]Pressman, Roger S. 2001. "*Software engineering: a practitioner's approach - 5th ed"*. McGraw-Hill.

[6]Ray, John. 2012. "*Sams Teach Yourself iOS 5 Application Development in 24 Hours"*. Indianapolis. Pearson Education, Inc.

[7]Sadun, Erica. 2010. "*The iPhone™ Developer's Cookbook : Building Applications with the iPhone 3.0 SDK, Second Editi*