IMPLEMENTASI REST WEB SERVICE UNTUK *SALES ORDER* DAN *SALES TRACKING BERBASIS MOBILE*

Erick Kurniawan

Abstrak

Peran tenaga penjual pada sebuah perusahaan sangatlah vital, karena mereka adalah ujung tombak dalam penjualan produk, karena itu perusahaan membutuhkan sistem yang dapat memantau aktivitas dan mempercepat proses pemesanan produk. Aplikasi berbasis mobile dapat menjadi solusi untuk menyelesaikan masalah tersebut. Aplikasi mobile yang akan dibuat memanfaatkan data dari GPS untuk memastikan lokasi dari tenaga penjual. Aplikasi yang dibuat juga memiliki fasilitas untuk membaca barcode barang menggunakan kamera untuk mempercepat input data barang. Aplikasi ini menggunakan REST Services untuk memanipulasi data yang ada pada layanan komputasi awan. Dengan menggunakan aplikasi mobile ini perusahaan dapat dengan mudah memantau tenaga penjual dan melakukan pemesanan barang dengan lebih cepat dan efisien.

Kata Kunci : *aplikasi mobile, aplikasi berbasis lokasi, REST API, Web Services*

1. Pendahuluan

Divisi Penjualan adalah ujung tombak perusahaan yang mempunyai tugas utama untuk memasarkan produk. Dengan menggunakan sistem informasi penjualan berbasis desktop/web, perusahaan sudah dapat mencatat transaksi penjualan. Namun ada beberapa masalah yang dihadapi oleh perusahaan dan tidak dapat terselesaikan dengan sistem informasi penjualan berbasis web/desktop. Masalah yang pertama adalah tentang pengawasan pegawai yang bertugas untuk menawarkan barang ke pelanggan. Perusahaan mempunyai pegawai yang bertugas untuk menjual dan menawarkan produk biasa disebut sebagai tenaga penjual, dan mereka yang akan datang ke pelanggan untuk menawarkan barang dan mencatat pesanan dari pelanggan. Masalah yang dihadapi adalah perusahaan merasa kesulitan untuk memastikan apakah tenaga penjual tersebut benar mendatangi pelanggan sesuai dengan target perusahaan. Untuk itu dibutuhkan sebuah aplikasi berbasis *mobile* yang dapat mencatat posisi dari tenaga penjual ketika mengunjungi pelanggan.

Masalah kedua yang dihadapi adalah perusahaan tidak dapat memantau secara *real-time* barang apa saja yang dipesan oleh pelanggan. Saat ini yang terjadi adalah tenaga penjual akan mencatat barang yang dibutuhkan oleh pelanggan, jika dibutuhkan informasi tentang ketersediaan barang yang dipesan maka tenaga penjual akan menanyakan informasi tersebut melalui telepon. Tenaga penjual juga tidak tahu pasti ketersediaan stok barang secara *real-time,* untuk itu dibutuhakan aplikasi *mobile* yang terkoneksi dengan internet untuk mengakses data barang di server yang berada di *Cloud*. Selain itu dibutuhkan juga metode yang lebih efisien untuk memasukan data barang tanpa harus mengetikan kode atau nama barang secara manual kedalam aplikasi berbasis *mobile* yang akan dibuat.

Tujuan dari penelitian ini adalah untuk menerapkan teknologi *Web Services* dengan arsitektur REST yang dapat digunakan oleh berbagai macam jenis client seperti aplikasi *mobile*, aplikasi Web, dan aplikasi Desktop yang dapat membantu perusahaan untuk melakukan pelacakan atau *tracking* terhadap tenaga penjual yang ditugaskan untuk menawarkan barang atau penagihan ke pelanggan. Dengan REST *Web Services* yang akan dibuat diharapkan perusahaan akan dapat memastikan bahwa semua tenaga penjual akan mengunjungi pelanggan sesuai dengan target yang sudah ditentukan oleh perusahaan. Selain itu penelitian ini juga akan menerapkan cara yang lebih

cepat untuk memasukan data pesanan barang kedalam sistem dengan memanfaatkan kamera pada perangkat *mobile* untuk pembacaan *barcode*.

2. Tinjauan Pustaka

a. Perangkat Mobile

Perkembangan perangkat mobile saat ini sudah sangat pesat. Ini ditandai dengan munculnya perangkat mobile cerdas (*smartphone*). *Smartphone* sendiri adalah jenis mobile phone yang memiliki berbagai macam fitur yang ada pada perangkat komputer seperti kalender, notifikasi, *task manager*, *browser*, *game*, dan berbagai aplikasi modern lainnya. Menurut survei dari KPCB Internet Trend 2013 pada tahun 2013 pengguna *mobile phone* meningkat sangat signifikan sebanyak 70% dan 20% diatantaranya adalah pengguna *smartphone* (KPCB, 2013). Indonesia sendiri menempati peringkat ke 8 untuk pertumbuhan pengguna *smartphone* terbesar, hanya satu tingkat dibawah Korsel. Sesuai dengan data pengguna smartphone tersebut, di Indonesia sendiri sudah banyak aplikasi web populer untuk layanan berita, perbankan, dan *e-commerce* yang mulai menyediakan layanan aplikasi berbasis *mobile* untuk penggunanya.

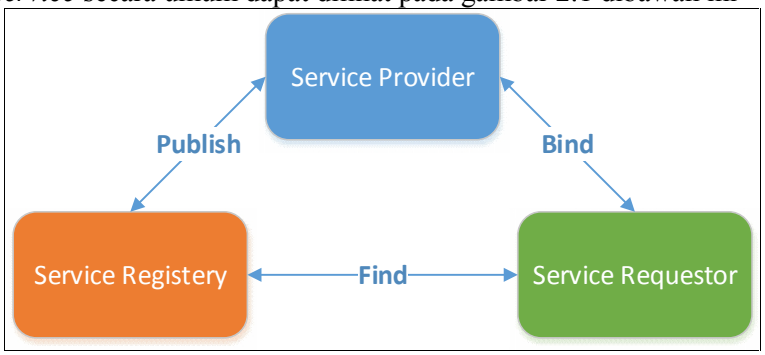
b. Web Services

Web Service adalah layanan yang tersedia di Internet. *Web Service* menggunakan format standar XML untuk pengiriman pesannya. *Web Services* juga tidak terikat kepada bahasa pemrograman atau sistem operasi tertentu (Ethan Cerami, 2002).

Web Services adalah antar muka yang mendeskripsikan koleksi yang dapat diakses dalam jaringan menggunakan format standar XML untuk pertukaran pesan. *Web Services* mengerjakan tugas yang spesifik. *Web Services* dideskripsikan menggunakan format standar notasi XML yang disebut services description (Gottschalk, 2002).

c. Arsitektur Web Services

Arsitektur *web service* secara umum dapat dilihat pada gambar 2.1 dibawah ini



Gambar 2.1 Arsitektur web service
(sumber: Brittenham, 2002)

Pada gambar diatas, ada tiga komponen utama dari web service yaitu:

- *Service provider*: Penyedia *web service* yang berfungsi menyediakan kumpulan *web services* yang dapat diakses oleh pengguna.
- *Service requestor*: Adalah aplikasi yang bertindak sebagai pengguna yang melakukan permintaan layanan (berupa *web services*) ke *service provider*.
- *Service registry*: Adalah tempat dimana service provider mempublikasikan layanannya. Pada arsitektur *Web service*, *Service registry* bersifat opsional.

d. REST (Representational State Transfer)

REST adalah filosofi desain yang mendorong kita untuk menggunakan protokol dan fitur yang sudah ada pada Web untuk memetakan permintaan terhadap sumber daya pada berbagai macam representasi dan manipulasi data di Internet (Scribner, 2009).

REST adalah gaya arsitektural yang memiliki aturan seperti antar muka yang seragam, sehingga jika aturan tersebut diterapkan pada web services akan dapat memaksimalkan kinerja *web services* terutama pada performa, skalabilitas, dan kemudahan untuk dimodifikasi. Pada arsitektur REST data dan fungsi dianggap sebagai sumber daya yang dapat diakses lewat *Uniform Resource Identifier* (URI), biasanya berupa tautan pada web.

REST menggunakan protokol HTTP yang bersifat *stateless*. Perintah HTTP yang bisa digunakan adalah fungsi GET, POST, PUT atau DELETE. Hasil yang dikirimkan dari server biasanya dalam bentuk format XML atau JSON sederhana tanpa ada protokol pemaketan data, sehingga informasi yang diterima lebih mudah dibaca dan diparsing disisi *client*.

Dalam penerapannya, REST lebih banyak digunakan untuk *web service* yang berorientasi pada *resource*. Maksud orientasi pada sumber daya adalah orientasi yang menyediakan sumber daya sebagai layanannya dan bukan kumpulan-kumpulan dari aktifitas yang mengolah sumber daya itu. Bentuk *web service* menggunakan REST *style* sangat cocok digunakan sebagai *backend* dari aplikasi berbasis mobile karena cara aksesnya yang mudah dan hasil data yang dikirimkan berformat JSON sehingga ukuran file menjadi lebih kecil.

e. Web API

Web API adalah antar muka program dari sistem yang dapat diakses lewat method dan header pada protokol HTTP yang standar. Web API dapat diakses dari berbagai macam HTTP *client* seperti browser dan perangkat *mobile*. Web API juga memiliki keuntungan karena menggunakan infrastruktur yang juga digunakan oleh web terutama untuk penggunaan *caching* dan *concurrency (*Block, 2014*)*.

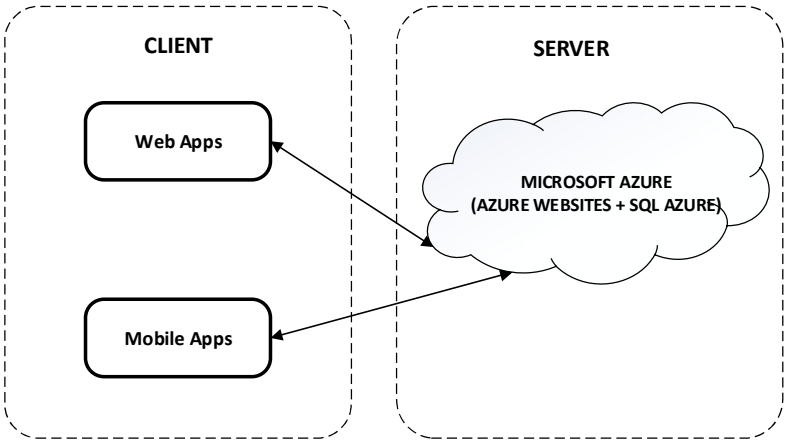
f. *GPS*

Global Positioning System (GPS) adalah metode penentuan posisi suatu objek di bumi, dalam semua kondisi cuaca. GPS menggunakan sejumlah satelit di orbit bumi untuk melakukan pelacakan posisi. GPS bekerja dengan menghitung jarak dari satelit penerima lokasi, minimal ada tiga satelit yang diperlukan untuk posisi dua dimensi dan empat satelit untuk posisi tiga dimensi. Satelit yang lebih banyak dapat menemukan posisi yang lebih akurat, sehingga titik persimpangan menjadi lebih kecil. Saat ini GPS semakin terintegrasi ke dalam kehidupan sehari-hari, contohnya GPS yang sudah tertanam di *smartphone* atau mobil.

3. Perancangan Sistem

a. Arsitektur Sistem

Secara umum aplikasi yang akan dibuat terdiri dari dua bagian besar yaitu aplikasi yang ada pada bagian *client* dan aplikasi yang ada pada bagian *server*. Aplikasi yang ada di server bertugas untuk menyediakan data yang dapat dikonsumsi oleh aplikasi *client*. Sedangkan aplikasi *client* digunakan untuk meminta data dari aplikasi yang ada di *server*.

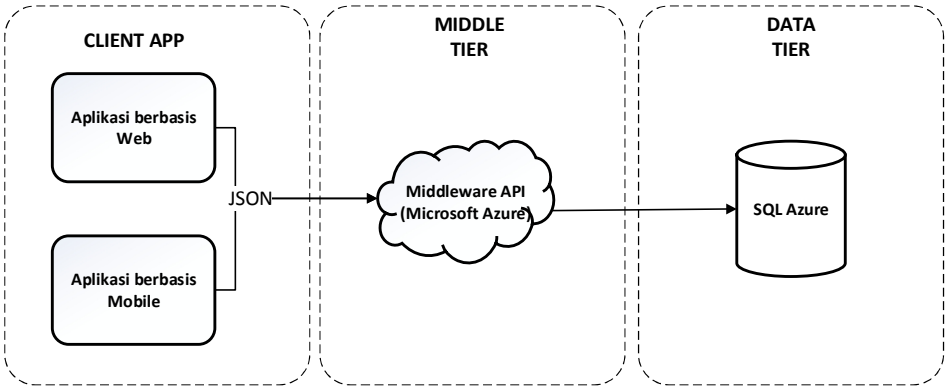


Gambar 3.1 Arsitektur Aplikasi

Pada Gambar 3.1 dapat dilihat arsitektur yang akan digunakan pada sistem yang akan dibuat. Pada gambar di sebelah kiri dapat dilihat bahwa aplikasi client yang akan dibuat terdiri dari tiga jenis yaitu aplikasi yang berbasis web, dan aplikasi berbasis *mobile*. Sedangkan di sisi *server* aplikasi yang akan dibuat adalah aplikasi *web services* yang menggunakan arsitektur REST.

Aplikasi *web services* dipilih dengan beberapa pertimbangan yaitu agar memudahkan pembangunan proses bisnis yang dapat digunakan untuk berbagai macam client tanpa harus membuat proses tersebut secara spesifik berdasarkan teknologi *client* yang digunakan. Dengan menggunakan *web services* maka hanya diperlukan sekali pengembangan aplikasi yang berisi bisnis proses, dan aplikasi ini tidak terikat dengan aplikasi *client* yang akan mengaksesnya. Aplikasi yang akan dibangun ini menggunakan tiga teknologi yang berbeda pada sisi client yaitu teknologi ASP.NET MVC untuk membuat aplikasi berbasis web, dan Android untuk aplikasi berbasis *mobile*. Tiga aplikasi *client* tersebut akan mengakses satu aplikasi *web services* yang sama untuk dapat saling terintegrasi antar aplikasi.

Jika dilihat dari pembagian antar *layer*, aplikasi yang akan dikembangkan terdiri dari tiga bagian yaitu *data layer* yang menggunakan database SQL Server, *middle layer* yang akan menggunakan teknologi ASP.NET Web API untuk pembuatan REST *Web Services*, dan terakhir aplikasi *client* yang berupa aplikasi berbasis *Desktop*, *Web*, dan *Mobile*. Pada penelitian ini akan digunakan layanan komputasi awan dari Microsoft yaitu Microsoft Azure untuk menaruh bagian *data layer* dan *middle layer*. Layanan Azure Website akan digunakan untuk menaruh aplikasi services berupa Web API dan layanan SQL Azure digunakan untuk penyimpanan data. Gambar pembagian layer dari sistem yang akan dibuat dapat dilihat pada Gambar 3.2 dibawah ini.

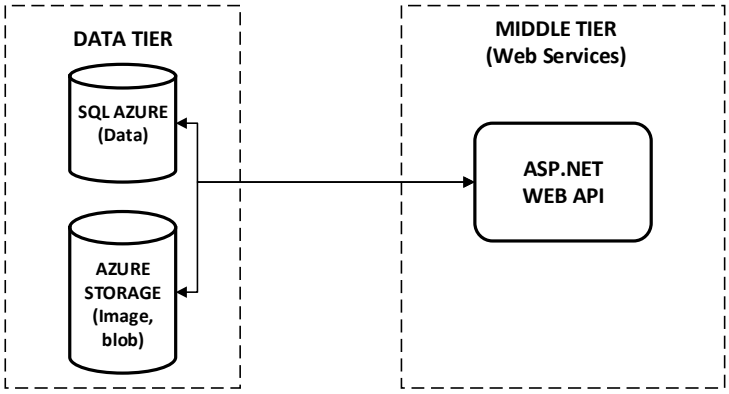


Gambar 3.2 Pembagian Antar Layer

b. Perancangan *Data Layer*

Data Layer adalah salah satu bagian penting yang menangani masalah penyimpanan data di basis data. Pada penelitian ini digunakan database *SQL Server* untuk penyimpanan data. Data yang disimpan pada basis data terdiri dari data Pelanggan, Barang, Transaksi Penjualan, dan Tenaga penjual. Basis data yang sudah dibuat di *development server* nantinya akan di pindahkan ke layanan komputasi awan yaitu SQL Azure. SQL Azure dipilih menjadi alternatif penyimpanan data karena kapasitas yang elastis (kapasitas penyimpanan dapat ditambah atau dikurangi secara mudah).

Pada penyimpanan file lain selain data, misalnya file image, pdf, atau file dengan format blob digunakan layanan Azure Blob Storage. Gambar 3.3 dibawah ini menjelaskan arsitektur perancangan *data layer* yang digunakan pada sistem.



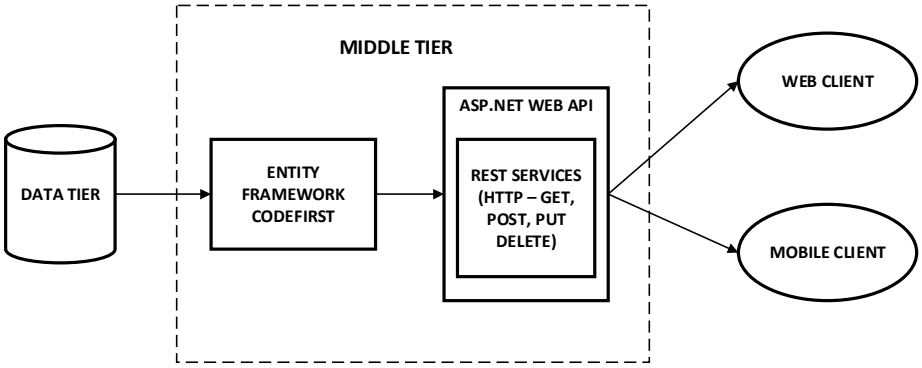
Gambar 3.3 Arsitektur Data Layer

c. Perancangan *Middle Layer*

Pada penelitian ini bagian *middle layer* adalah inti dari aplikasi yang dibuat. Pada *layer* inilah dibangun aplikasi *web services* dengan menggunakan arsitektur REST. Untuk membangun *web services* dengan arsitektur REST digunakan teknologi ASP.NET Web API. ASP.NET Web API merupakan teknologi yang menggunakan *framework* .NET untuk membangun *web services* menggunakan arsitektur REST. Pada platform .NET ada beberapa teknologi yang dapat dipilih sebagai alternatif dalam mengembangkan aplikasi *web services*, beberapa teknologi tersebut diantaranya adalah ASP.NET Web Services, WCF (*Windows Communication Foundation*), dan

ASP.NET Web API. Perbedaan antara ketiga teknologi tersebut adalah bahwa ASP.NET Web Services dan WCF menggunakan protokol SOAP (*Simple Object Access Protocol*) dalam pengiriman datanya, sedangkan ASP.NET Web API menggunakan arsitektur REST.

Pemilihan penggunaan ASP.NET Web API dikarenakan Web API sudah mendukung arsitektur REST secara default. *Web Services* dengan arsitektur REST sangat cocok digunakan oleh aplikasi berbasis web maupun aplikasi mobile karena format data yang dikirimkan ke *client* menggunakan JSON (*Javascript Object Notation*) yang berukuran lebih kecil dibandingkan dengan format XML. Gambar 3.4 menunjukan arsitektur dari *middle layer* dalam sistem yang dibuat.



Gambar 3.4 Arsitektur Middle Layer

Untuk berkomunikasi dengan database digunakan *framework* EF (*Entity Framework*). Pada penelitian ini framework ORM (*Object Relational Mapping*) yang digunakan adalah *Entity Framework* dengan pendekatan *Code First*. Dengan menggunakan *Code First* maka langkah pembuatan basis data dapat langsung dibuat melalui kode, tanpa harus membuat basis data dan tabel-tabelnya terlebih dahulu. *EF Code First* akan secara otomatis membuat basis data berserta tabel-tabel ketika aplikasi dijalankan.

Dengan *EF Code First* juga dimungkinkan untuk merubah skema dari tabel dan kolom yang ada pada basis data dengan cara memodifikasi kode yang ada di program. Contoh penggunaan *EF Code First* untuk membuat tabel Barang dapat dilihat pada Gambar 3.5 berikut:

```
public class Barang {
    [Key]
    public string KodeBarang { get; set; }
    public string NamaBarang { get; set; }
    public int Stok { get; set; }
    public decimal HargaBeli { get; set; }
    public decimal HargaJual { get; set; }
    public string PicUrl { get; set; }
}
```

Gambar 3.5 Contoh Penggunaan EF CodeFirst

Untuk membuat REST *web services* dengan ASP.NET Web API langkah pertama yang harus dilakukan adalah membuat *class Controller* yang diturunkan dari class API *Controller*. Contoh pembuatan REST API untuk menampilkan data Barang dapat dilihat pada Gambar 3.6 berikut ini.

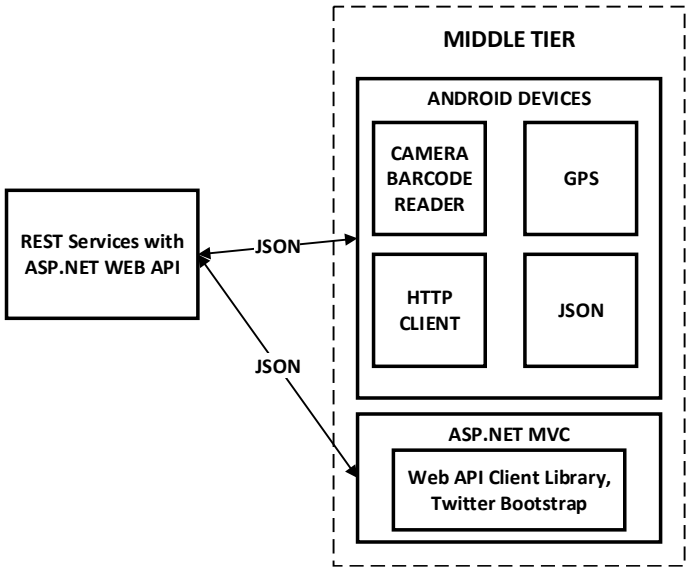
```
public class BarangController : ApiController
{
        private SampleWebApiContext db = new SampleWebApiContext();
        // GET api/Barang
        public IQueryable<Barang> GetBarangs()
        {
            var result = from b in db.Barangs
                         orderby b.NamaBarang descending
                         select b;
            return result;
        }
}
```

Gambar 3.6 Contoh Pembuatan Web API Controller

d. Perancangan Aplikasi Client

Pada sistem ini terdapat dua macam aplikasi yang digunakan untuk mengakses REST *web services*. Aplikasi *client* yang pertama adalah aplikasi berbasis web yang dibuat menggunakan teknologi ASP.NET MVC, dan aplikasi yang kedua adalah aplikasi berbasis mobile yang dibuat dengan menggunakan Android.

Aplikasi berbasis web yang dibuat akan menggunakan pustaka *Web API Client Library* untuk mengakses *web services*. Sedangkan untuk aplikasi *mobile* yang menggunakan android digunakan pustaka *HttpClient* dan JSON parser untuk mengambil data dan membaca data tersebut dari *web services*.



Gambar 3.7 Arsitektur Aplikasi Client

Pada aplikasi *mobile* yang dibuat juga digunakan sensor GPS untuk melakukan pencatatan lokasi. Informasi ini kemudian akan disimpan kedalam basis data menggunakan API *service*. Aplikasi mobile ini juga akan memanfaatkan library untuk pembacaan *barcode* dari kamera.

4. Hasil Dan Pembahasan

a. Penerapan REST Web Service

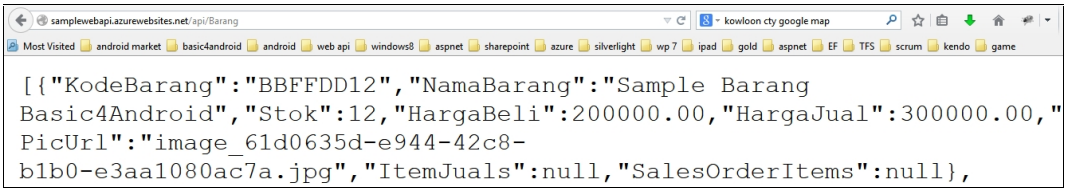
Pada bagian *Controller* kita juga harus mendefinisikan metode yang akan digunakan untuk pemanggilan *web services* tersebut.

Karena *web service Web Services* yang dibangun dalam sistem ini adalah bagian yang terpenting. *Web Services* yang dibuat pada sistem ini menggunakan teknologi ASP.NET Web API. Karena ASP.NET Web API adalah bagian dari ASP.NET MVC maka langkah pertama untuk membuat services adalah dengan membuat *Controller*. *Controller* disini berperan untuk menentukan informasi apa yang akan dikirimkan ketika pengguna mengakses *web services* yang kita buat.yang dibuat menggunakan arsitektur REST maka *method* yang dapat digunakan adalah *method* yang didukung oleh protokol HTTP seperti method GET, POST, DELETE, dan PUT. Berikut adalah contoh potongan kode untuk pembuatan *services*.

```
// GET api/Barang
public IQueryable<Barang> GetBarangs(){
     var result = from b in db.Barangs
        orderby b.NamaBarang descending
        select b;
     return result;
}
```

Gambar 4.1 Service dengan method GET

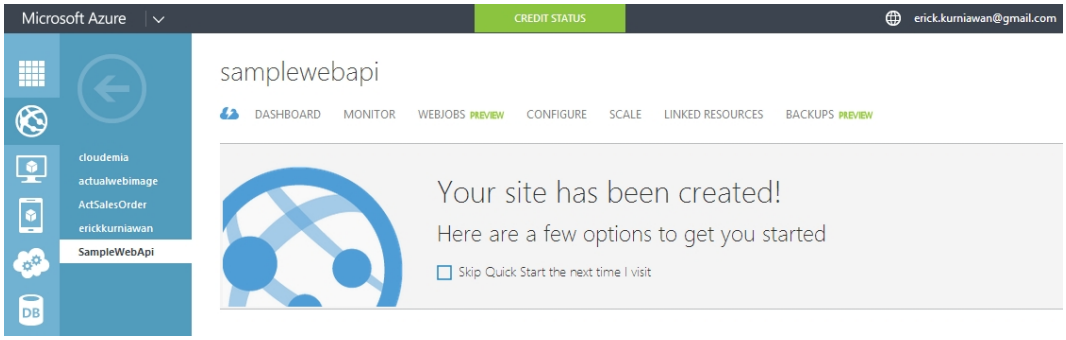
Pada Gambar 4.1 diatas dapat dilihat kode untuk menampilkan data barang. Layanan tersebut dapat diakses dengan menggunakan alamat tertentu. Contoh format alamat yang digunakan adalah sebagai berikut http://samplewebapi.azurewebsites.net/api/Barang. Bagian pertama dari alamat tersebut adalah nama domain, kemudian diikuti dengan prefik ‘api’ yang digunakan sebagai penanda bahwa yang diakses adalah sebuah API *service*. Bagian yang terakhir adalah nama *Controller* yaitu ‘Barang’. Hasil pemanggilan *service* dapat dilihat pada gambar 4.2 dibawah ini.



Gambar 4.2 Hasil dokumen dalam format JSON

Hasil yang didapatkan berupa dokumen JSON yang nantinya akan diunduh dan ditampilkan kedalam aplikasi *mobile*. Untuk operasi seperti tambah, ubah, dan delete data dapat digunakan method POST, UPDATE, dan DELETE.

Pada penelitian ini service yang dibuat dipasang pada layanan komputasi awan dari Microsoft yaitu Microsoft Azure. Layanan yang digunakan adalah Azure Website dimana *Web Service* yang dibuat akan di taruh pada layanan ini, sedangkan untuk data akan disimpan kedalam layanan lain yaitu SQL Azure. Tampilan pengaturan dari Microsoft Azure dapat dilihat pada gambar 4.3 berikut ini.

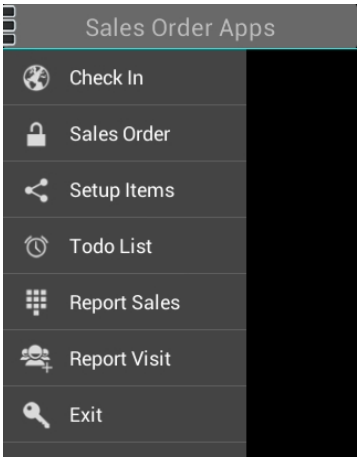


Gambar 4.3 Tampilan Manajemen Microsoft Azure

b. Aplikasi *Client*

Ada dua jenis aplikasi *client* yang dibuat pada penelitian ini. Yang pertama adalah aplikasi berbasis web. Aplikasi web akan digunakan untuk menampilkan pelaporan-pelaporan yang akan dapat digunakan oleh pimpinan perusahaan untuk melihat performa tenaga penjual. Laporan yang ditampilkan meliputi pencatatan kegiatan harian seperti kunjungan ke pelanggan, jam kunjungan, jam kedatangan, jarak lokasi ketika melakukan *check-in* dengan jarak lokasi pelanggan,

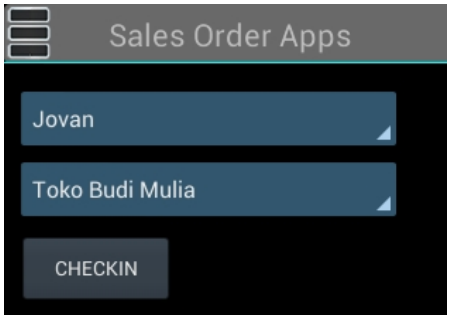
Tampilan menu dari aplikasi *mobile* yang dibuat dapat dilihat pada gambar 4.4 dibawah ini. Menu yang disediakan pada aplikasi ini adalah menu *check-in* untuk tenaga penjual, pemesanan barang dengan fasilitas pembacaan kode *barcode* dari kamera, dan beberapa laporan seperti penjualan yang dilakukan tenaga penjual perbulan.



Gambar 4.4 Menu Aplikasi Mobile

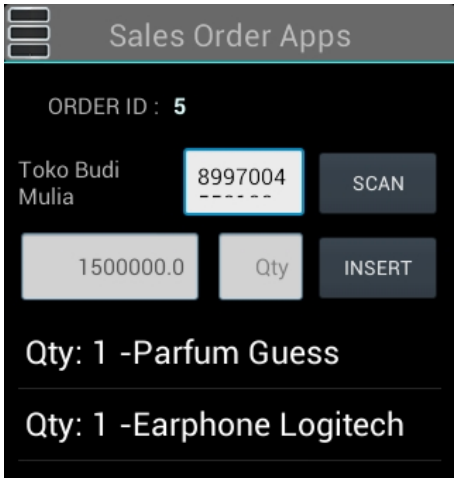
Fitur yang pertama adalah untuk *check-in* tenaga penjual. Fitur ini akan mencatat posisi dari tenaga penjual dan membandingkannya dengan lokasi pelanggan. Jika jarak antara lokasi *check-*

in dan pelanggan melebihi jarak yang ditentukan, maka sistem akan menolak proses *check-in*. Jadi sistem ini diharapkan dapat membantu atasan dalam mengawasi tenaga penjual yang ditugaskan untuk mengunjungi pelanggan. Gambar 4.5 dibawah ini adalah tampilan untuk *check-in* tenaga penjual.



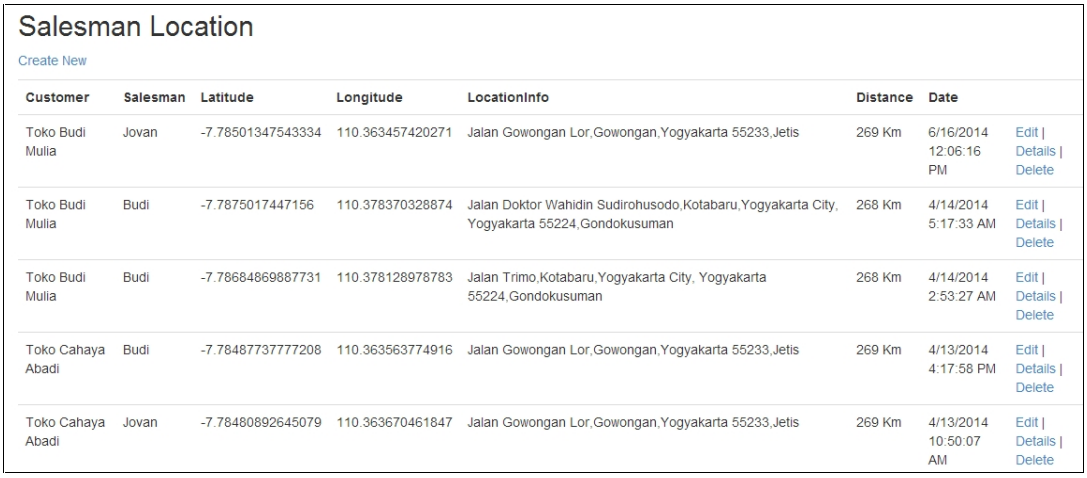
Gambar 4.5 Fitur untuk Check-In Tenaga Penjual

Fitur selanjutnya adalah *sales order*, fitur ini digunakan untuk mencatat barang-barang yang akan dipesan oleh pelanggan. Pada fitur ini perangkat kamera akan digunakan sebagai alat untuk membaca kode *barcode* dari barang yang akan dipesan. Setelah tenaga penjual selesai melakukan pemesanan maka data barang yang akan dipesan akan dikirimkan ke *server*, dengan begitu persediaan dari barang bias diketahui terlebih dahulu. Jika persediaan barang masih ada maka barang dapat dipesan, namun jika persediaan barang sudah habis maka akan muncul pesan peringatan. Gambar 4.6 dibawah ini menunjukan bagaimana cara untuk memesan barang.



Gambar 4.6 Fitur untuk Sales Order

Aplikasi berbasis web akan digunakan oleh atasan untuk memastikan bahwa semua tenaga penjual telah memenuhi target untuk mengunjungi pelanggan sesuai dengan jadwal yang sudah ditentukan oleh perusahaan.



Salesman Location

Create New

Customer	Salesman	Latitude	Longitude	LocationInfo	Distance	Date	
Toko Budi Mulia	Jovan	-7.78501347543334	110.363457420271	Jalan Gowongan Lor,Gowongan,Yogyakarta 55233,Jetis	269 Km	6/16/2014 12:06:16 PM	Edit | Details | Delete
Toko Budi Mulia	Budi	-7.7875017447156	110.378370328874	Jalan Doktor Wahidin Sudirohusodo,Kotabaru,Yogyakarta City, Yogyakarta 55224,Gondokusuman	268 Km	4/14/2014 5:17:33 AM	Edit | Details | Delete
Toko Budi Mulia	Budi	-7.78684869887731	110.378128978783	Jalan Trimo,Kotabaru,Yogyakarta City, Yogyakarta 55224,Gondokusuman	268 Km	4/14/2014 2:53:27 AM	Edit | Details | Delete
Toko Cahaya Abadi	Budi	-7.78487737777208	110.363563774916	Jalan Gowongan Lor,Gowongan,Yogyakarta 55233,Jetis	269 Km	4/13/2014 4:17:58 PM	Edit | Details | Delete
Toko Cahaya Abadi	Jovan	-7.78480892645079	110.363670461847	Jalan Gowongan Lor,Gowongan,Yogyakarta 55233,Jetis	269 Km	4/13/2014 10:50:07 AM	Edit | Details | Delete

Gambar 4.7 Melihat Data Check-in

Fitur lain yang disediakan pada aplikasi berbasis web ini adalah memonitor transaksi pemesanan barang. Perusahaan akan dengan mudah dapat memantau barang apa saja yang dipesan oleh pelanggan. Gambar 4.8 dibawah ini menunjukan bagaimana tampilan monitoring pemesanan barang.

Sales Order with ID = 5

NamaBarang	HargaJual	Quantity	
Parfum Guess	1500000.00	1	Edit | Details | Delete
Earphone Logitech	1200000.00	1	Edit | Details | Delete

Gambar 4.8 Melihat Data Pemesanan Barang

5. Penutup

Kesimpulan yang dapat diambil pada penelitian penerapan REST *service* sebagai *backend* dari aplikasi *mobile* ini adalah:

- Penerapan REST service dengan format JSON sebagai *backend* pada sistem ini sangat cocok digunakan karena dengan format dokumen JSON yang kecil maka proses pengunduhan data dari *web service* lebih cepat dibandingkan dengan penggunaan dokumen XML yang ukurannya relatif lebih besar.
- Perancangan arsitektur *multi-tier* yang digunakan pada sistem ini dapat memudahkan dalam proses pengembangan aplikasi *client*, dengan menyediakan *middle layer* berupa REST *service*, maka tidak perlu dikembangkan bisnis proses yang sama untuk setiap aplikasi *client* yang akan dibuat.
- Pencatatan lokasi dengan memanfaatkan sensor GPS yang ada di aplikasi ini dapat membantu perusahaan untuk dapat melakukan pemantauan tenaga penjual dengan lebih mudah dan akurat.

- Pemanfaatan teknologi kamera untuk pembacaan *barcode* yang digunakan pada aplikasi ini dapat mempercepat proses pemesanan barang. Waktu yang digunakan oleh tenaga penjual untuk memasukan data kedalam sistem juga lebih cepat.

Daftar Pustaka

Block, G., Cibaro, P., Felix, P., Dierking, H., & Miller, D. (2014). *Designing Evolvable Web APIs with ASP.NET*. O'Reilly

Brittenham, Peter. (2002). *An overview of the Web Services Inspection Language*, http://www.ibm.com/developerworks/library/ws-wsilover/, diakses 1 Februari 2014

Cerami, E. (2002). Web Services Essential. O'Reilly

Dyche, Jill (2004). *The CRM Handbook: A Business Guide to Customer Relationship Management*. Addison-Wesley.

Gottschalk, K. (2002), *Introduction to Web Services Architecture. IBM System Journal*, Vol 41, No 2.

KPCB. (2013), *Internet Trends Report*, http://www.slideshare.net/kleinerperkins/kpcb-internet-trends-2013, diakses 1 Februari 2014

Scribner, Kenn., Seely Scott. (2009). *Effective REST Services via .NET*. Pearson Education, Inc.