DESARROLLO ÁGIL DE APLICACIONES EMPRESARIALES NATIVAS DE ANDROID QUE CONSUMEN SERVICIOS WEB SOAP EN LA NUBE.

AGILE DEVELOPMENT OF ANDROID NATIVE BUSINESS APPLICATIONS WHICH USE SOAP WEB SERVICES IN THE CLOUD

(Entregado 03/09/2015 – Revisado 29/10/2015)

MSc. Diego Javier Trejo España

M. Sc. En Gerencia Informática, Ingeniero en Sistemas Computacionales. Actualmente es Docente a Tiempo Completo en la Carrera de Ingeniería en Sistemas Computacionales de la Facultad de Ingeniería en Ciencias Aplicadas de la Universidad Técnica del Norte, Ibarra-Ecuador. Fue consultor informático de la Presidencia de la República, desempeñó varias Jefatura de Departamentos de Tecnología, ha dirigido varios Proyectos de Tecnología y actualmente es Consultor Privado. Tiene varios Méritos Académicos, entre ellos el de ser el mejor egresados de todos los Programas de Maestría en la Universidad Católica del Ecuador en el año 2010.

Universidad Técnica del Norte
Facultad de Ingeniería en Ciencias Aplicadas, Carrera de Ingeniería en Sistemas Computacionales, , Av. 17 de Julio 5-21, Ibarra, Imbabura
djtrejo@utn.edu.ec

INFORMÁTICA

Resumen.

El creciente uso de dispositivos móviles inteligentes, ha traído además una demanda cada vez más elevada de aplicaciones móviles. El mercado actual lo lideran los dispositivos con Sistema Operativo Android, por tanto es necesario satisfacer de una manera ágil la demanda de aplicaciones Android. Al respecto, existen especificaciones generales que deben cumplir las aplicaciones móviles, así como una arquitectura que incluye el consumo de Web Services en formato XML SOAP que encapsulan Bases de Datos empresariales. Los IDE's de desarrollo para Android más prometedores no incluyen librerías con soporte total para el consumo de Web Services, por lo que hay que evaluar librerías de terceros o incluso crear nuestras propias librerías para acelerar el desarrollo de aplicacines Android con acceso a Web Services Empresariales.

En el presente trabajo se plantea una solución que reduce sustancialmente el tiempo de desarrollo del tipo de aplicaciones móviles nativas para Android.

Palabras Claves

Servicios Web, Android, SOAP, Parser XML, Comunicación Móvil, Metodología Ágil de Desarrollo.

Abstract.

The increasing use of smart mobile devices, it has also brought an increasingly higher demand for mobile applications. The current market as the leading Platform Android devices, so it is necessary to satisfy a demand for agile Android applications. In this regard, there are general specifications to be met by mobile applications and an architecture that includes the use of XML Web Services SOAP format that encapsulate business data bases. The IDE's development for Android more promising exclude libraries with full support for consuming Web Services, so you need to evaluate third-party libraries or even create our own libraries to accelerate the development of Android App with access to Web Services Business.
In this paper a solution that substantially reduces development time the type of native mobile applications for Android.

Keywords

Web Services, Android, SOAP, XML Parser, Mobile Comunication, Agile Development Methodology

Introducción

En Ecuador, a mayo del 2014, se ha contabilizado cerca de 20 millones de abonados de telefonía móvil [1], es decir, frente a la población proyectada de 16 millones, según el último censo nacional [2], el índice de penetración de este servicio bordea el 125%, representando uno de los más altos de Latinoamérica, incluso supera la media mundial que es el 59% [3]. Así también, la cantidad de cuentas de acceso a Internet ha superado los 11.5 millones, de los cuales 4.3 (más del 37%) se hacen a través de telefonía móvil [1]. En este contexto es razonable que el mercado de aplicaciones para dispositivos móviles se haya convertido en una economía de crecimiento acelerado. Según la compañía tecnológica estadounidense Compuware, en 2011 se efectuó 15.000 millones de descargas y en 2012 se realizaron alrededor de 50.000 millones. De manera proporcional, estas cifras definitivamente tienen presencia en Ecuador, aunque hasta hoy no hay fuentes que precisen la cantidad, tipo de dispositivo, tipo de aplicación, la frecuencia, los costos y el target de usuarios. Sin embargo, cualquiera sea el caso de la oferta y demanda de aplicaciones móviles, representan una oportunidad de desarrollo económico basada en el conocimiento, lo cual encaja con la actual política pública a través de la nueva matriz productiva propuesta.

Dado los antecedentes de la creciente demanda de aplicaciones móviles, han debido investigarse e innovarse metodologías ágiles de desarrollo, a pesar de la presencia de plataformas incompatibles como son: Windows Mobile, iPhone, Android, Symbian, entre otros, que en definitiva hacen que a los desarrolladores les tome esfuerzos extra para la fabricación de software móvil multi-plataforma [4, 5]. Entonces, en 2011 se definió que una Metodología Ágil de Desarrollo de Software debe permitir a los equipos crear aplicaciones de manera rápida y responder a los cambios que puedan surgir mientras se ejecuta el proyecto, evitando los burocráticos caminos de las metodologías tradicionales [5]. Así también nació la necesidad de crear herramientas de desarrollo de software móvil concomitantes con las metodologías ágiles, para lo cual existen varias opciones explicadas en [6-12].

Por otro lado, existen particularidades en el desarrollo de software móvil, esto viene dado por: la infraestructura de comunicaciones, los tipos de dispositivos, el sistema operativo, las especificaciones de diseño, uso de batería, almacenamiento interno/externo limitado, gestión de conexión/desconexión, disponibilidad de red, tolerancia a fallos en la red, costos de conexión a la red, entre

otros [13]. Lo cual ha llevado a establecer ciertas especificaciones aplicables de manera general en el desarrollo de software móvil, estas son: a) El software se limita al uso de las API del SO [12]; b) El software no es una solución completa, es parte de una; es decir, se limita por ejemplo a ser una interface de captura o consumo de información de un sistema de software completo [13]; c) Se debe considerar las intermitencias de disponibilidad en las comunicaciones [13]; d) Asumir condiciones mínimas de operación, bajo consumo de energía, poco almacenamiento, poco uso de red [13].

En [11, 13-23] analizan y concluyen en una solución de arquitectura que satisface las limitantes expuestas; es de mencionar que esta solución se considera como más viable dado que utiliza estándares abiertos en protocolos de comunicación e intercambio de datos. Básicamente se trata de dos aplicaciones: a) El servidor, acompañado con una infraestructura de firewall y sistemas de Bases de Datos Empresariales; entre todos conforman una "Middleware Layer" (Capa Intermediaria) cuya interface de acceso son Web Services. b) La aplicación cliente que consume los Web Services desde y a través de dispositivos y redes móviles. A todo esto se le puede añadir un eje transversal de seguridades implementado con certificados de autenticidad o también con encabezados conteniendo identidad encriptada.

No olvidar que una de las limitantes de las aplicaciones móviles es que dependen de las API's del Sistema Operativo (SO) que las aloja, al respecto, es muy bien sabido que en el mercado se imponen: Android, iPhone y Windows Phone, en ése orden; un estudio que sustenta el liderazgo de Android se puede encontrar en [12, 24, 25]. Allí se destacan como factores de éxito a que se trata de una plataforma Open Source, basado en el kernel de Linux, patrocinada por Google y sobre todo porque ofrece un SDK muy completo. En todo caso, cualquiera sea el SO donde se ejecuten, existen tres tipos de aplicaciones móviles: a) Nativas, b) Web, c) Híbridas [26]. Y para cada una de ellas existen varios IDE's y SDK's de desarrollo, mismos que además pueden ser Open Source o Propietarios.

Android brinda un completo kit de herramientas para planeación, diseño, desarrollo y despliegue de Aplicaciones Nativas [27], son de libre acceso y tienen una curva de aprendizaje muy corta [12]. En el sitio oficial de Desarrolladores para Android [27] se pueden encontrar absolutamente todas las herramientas entre las que se destacan tres: a) Android SDK, que es la base de todo desarrollo; b) Android Developer Tools (ADT), un plugin que proporciona un IDE para Eclipse; c) Android Studio, un IDE que aunque se encuentra en versión Beta, los creadores prometen mejoras respecto a ADT, sobre todo en la simplificación de tareas de desarrollo.

Los siguientes son los más destacados IDE's para desarrollo de Aplicaciones Android Nativas: a) ADT con Eclipse, ya descrito antes; b) Android Studio, también ya mencionado antes; c) Motodev, patrocinado por Motorola y basado en ADT; d) Xamarin (privativo), ahora auspiciado por Microsoft; e) B4A (privativo pero de muy bajo costo), usa Visual Basic como lenguaje de programación, con una creciente comunidad de desarrolladores; f) Appcelerator Titanium (free), igual requiere el SDK de Android; g) PhoneGap, basada en node.js y Apache Cordova, donde se desarrolla la interface de usuario en HTML5 y CSS3 y se genera código nativo; h) Icenium (privativa), de la empresa Telerik y de similar naturaleza que Phonegap.

El problema

Apegados a cualquier Metodología Ágil de Desarrollo, así como encuadrados en una arquitectura que use Web Services Empresariales y aplicaciones cliente nativas de Android, y que además garanticen el cumplimiento de los requisitos generales que una aplicación móvil debe cumplir, se pretende reducir el tiempo de desarrollo del consumos de los Web Services que encapsulan Bases de Datos Empresariales, como lo son Oracle, SqlSevrer, Sybase, Informix, etc. Además, los mencionados servicios web, obedecerán al formato estándar XML SOAP 1.1 o 1.2 definidos en [28].

La solución propuesta deberá usar ADT como el IDE de desarrollo de libre acceso y propio de Android; de las versiones privativas, se usará a la más económica, B4A, además que ofrece un desarrollo acelerado de aplicaciones (RAD) [29]. Entre las dos se deberá analizar la que menos tiempo tome el mencionado desarrollo.

Para resaltar la importancia a la búsqueda de una solución, es preciso describir lo más relevante de la mecánica operativa de un Web Service SOAP [28]. Primero, por cada Web Service existen varios Web Method, que son los que ejecutan las operaciones en el servidor y devuelven datos de resultado generalmente con diferente estructura cada uno. Segundo, estos Web Method son invocados por las aplicaciones cliente (en nuestro caso las aplicaciones Android) y reciben los mencionados resultados. Tercero, las aplicaciones cliente en principio no conocen la estructura de los datos de resultado, por tanto no pueden interpretarlos. Sin embargo, se pueden llegar a conocer consultando la definición WSDL [28], que también se publica en el Web Service. Cuarto, las aplicaciones cliente previo a la invocación de un Web Method deben tener definidas similares estructuras de datos para poder recibir los resultados y poder interpretarlos o usarlos. En aplicaciones empresariales no se puede precisar la cantidad de Web Method por cada Web Service que podrían existir, pero tomemos como ejemplo al menos unos 20 Web Method y 5 Web Service, lo que en total serán 100. En este contexto, el desarrollador de una aplicación cliente podría llegar a programar hasta 100 estructuras de datos (comúnmente clases en POO). Entonces aquí radica la importancia de la solución a buscar, reducir el tiempo de desarrollo no solamente de las 100 clases o estructuras de datos, sino además de las líneas de código para las 100 invocaciones a los Web Method, así como las líneas de código para la recepción de los 100 tipos de resultado.

La situación actual

De seguro, cualquier programador podría decir que una solución son las librerías de software, pero ha de decirse que según [27] Android SDK no proporciona librerías para este fin, sin embargo existe un fabricante de software para Android SDK que se destaca por haber creado una librería que ayuda sustancialmente a este caso, nos referimos a Ksoap2.Android [9].

Por otro lado en [29] encontramos otras librerías que en conjunto también son de gran ayuda, se trata de HTTPUtils2 y XML SAX. Aunque estas estén destinadas a usarse en B4A, pero su resultado es similar a Ksoap2.Android.

Existen aplicaciones nativas Android que han usado las librerías Ksoap2.Android y cuyos resultados son buenos según sus autores. Por ejemplo se tiene en [14] una aplicación de Login Seguro; en [30] se explica una aplicación de monitoreo de datos espaciales; en [15] se habla de una aplicación de Learning; otra solución es vigilancia de predios y notificaciones a través de dispositivos móviles [16]; un servicio de localización vía GPS explicado en [17]; en [13] si bien no desarrolla una aplicación, pero destaca la importancia del uso de Ksoap2; en [22] desarrollan una aplicación para couriers; en [23] implementan una aplicación para seguimientos de tareas.

En cuanto a aplicaciones que hayan usado HTTPutils2 y XML SAX no se ha encontrado datos oficiales, más allá de las experiencias difundidas en los foros de la comunidad B4A [29], donde por cierto se puede encontrar mucha información enriquecedora para un desarrollador.

Objetivos

General: Reducir al máximo el tiempo de desarrollo del consumo de Web Services Empre-

sariales en formato XML SOAP, desde aplicaciones nativas de Android desarrolladas ya sea con ADT o B4A, con la finalidad de contribuir con los preceptos de cualquier Metodología de Desarrollo Ágil y aplicando las normas generales que una aplicación móvil debe cumplir.

Específico 1: Analizar los tiempos de desarrollo del consumo de los Web Services para ambos IDE's.

Específico 2: Diseñar una nueva librería, basada o no en las existentes, que permita alcanzar menores tiempos de desarrollo.

Materiales y Métodos

Métodos

Para analizar la reducción del tiempo de un desarrollo hay que utilizar métricas definidas en Ingeniería de Software, así se tiene:

• Métricas de Tamaño de Producto

• Métricas de Productividad de Desarrolladores

• Métricas de Reusabilidad

• Métricas de Tiempo de Desarrollo

Las métricas de tamaño generalmente toman en cuenta las líneas de código fuente del software resultante, sin incluir las líneas de comentarios. Las métricas de productividad de desarrolladores toman en cuenta las líneas de código fuente "escrito", es decir no incluye comentarios ni líneas de código generadas de manera automática por herramientas de apoyo como son los frameworks. Estas medidas conllevan al cálculo del tiempo real invertido en una tarea de programación, lo cual conduce a determinar el costo. Como otra medida, también consta la curva de aprendizaje de los desarrolladores en cuanto a herramientas y comprensión de los procesos inherentes.

La métrica de reusabilidad hace referencia a la cantidad de subrutinas existentes y el número de veces que se invocan. Esto denota el nivel de modularidad de un programa y la complejidad de comprensión para futuros mantenimientos.

Por tanto, la métrica de tiempo de desarrollo involucra la combinación de las anteriores, pues a mayor reusabilidad menor será el tamaño del producto, a menor tamaño de producto menor esfuerzo del desarrollador, por tanto menor será el tiempo de desarrollo.

Adicionalmente, nótese las ventajas: En aplicaciones móviles, cuando el tamaño de producto es pequeño menor es el tiempo de transmisión cuando se instala, y menor será el costo de comunicaciones consumido, así como menor será el espacio de alojamiento en el dispositivo móvil. Por otro lado mayor será la productividad del desarrollador en tiempo y costo.

Finalmente, se podría medir el tiempo consumido o la cantidad de bytes transmitidos en los servicios de red; datos muy importantes en aplicaciones móviles.

Materiales

Software requerido para el desarrollo de la aplicación Android: Sistema Operativo Windows 8 64bits; Java SDK 6 para Windows 64bits; IDE de desarrollo Eclipse / Windows 64bits; Android SDK para Windows 64bits; Android Developer Tools (ADT) para Eclipse / Windows 64bits; Librerías

KSOAP2.Android para ADT; IDE de desarrollo B4A para Windows 64bits; Librerías HHTPUtils2 y XML SAX para B4A; Droid at Screeen (Para visualizar la pantalla del teléfono Android en el PC).

Software requerido para el desarrollo de la aplicación que publica los Web Services XML SOAP: Sistema Operativo Windows 8 64bits; IDE de desarrollo Visual Studio .NET 2012; Servidor Web IIS 7; Base de datos Oracle 11g XE para Windows 64 bits; Librerías nativas .NET para conexión a Oracle (ODT – Oracle Data Tools) Ver.12.

Las siguientes herramientas de hardware: PC Intel Core i5, 8Gb en Ram, y 500 Gb en disco, acceso a red mediante Wifi o tarjeta Ethernet; Teléfono Samsung Galaxy S4 Mini con SO Android 4.2.2 con conexión USB y Wifi/3G; Cable de conexión USB.

Implementación de la plataforma experimental de Web Services XML SOAP.

En el PC se instaló el siguiente software y en el orden indicado:

- Sistema Operativo Windows 8 64bits
- IIS 7
- Visual Studio 2012
- Base de Datos Oracle 11g XE con el esquema de ejemplo HR.
- Librerías ODT de Oracle

En el esquema de ejemplo HR de Oracle, se procedió a crear dos Stored Procedures con el siguientes

```
CREATE OR REPLACE PROCEDURE ELECT_DEPARTAMENTOS(

        p_result    OUT SYS_REFCURSOR )

AS BEGIN

        OPEN    p_result FOR
        SELECT  *
        FROM    DEPARTMENTS
        ORDER   BY      DEPARTMENT_NAME;

END;
CREATE OR REPLACE

PROCEDURE SELECT_EMPLEADOS(

        p_departament_id    INT,
        p_result                        OUT SYS_REFCURSOR )

AS BEGIN

        OPEN    p_result FOR
        SELECT  *
        FROM    EMPLOYEES
        WHERE   DEPARTMENT_ID = p_departament_id
        ORDER   BY FIRST_NAME, LAST_NAME;

END;
```

Script 1. Creación de Stored Procedures en el esquema HR de Oracle

No hubo necesidad de crear script para alimentar datos de ejemplo, pues las tablas ya vienen creadas en el esquema HR e incluyen dichos datos de ejemplo.

Seguidamente, en Visual Studio .NET se crea un nuevo WebSite con el nombre “ws”:

Figura 1. Menú de Visual Studio 2012 para crear nuevo web site.

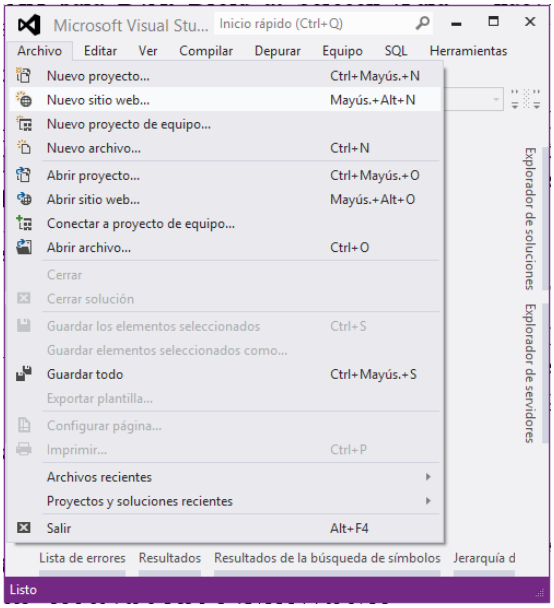
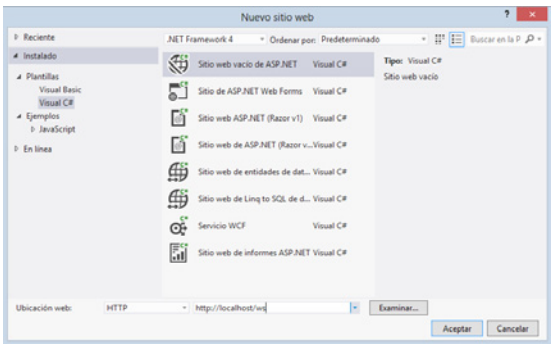


Figura 2. Creación de web site “ws”.



Una vez creado el web site, se agrega la referencia a las librerías ODT de Oracle.

Figura 3. Menú Agregar referencia

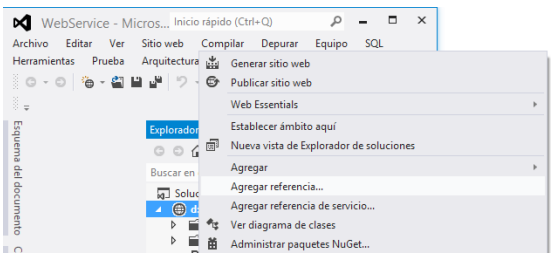
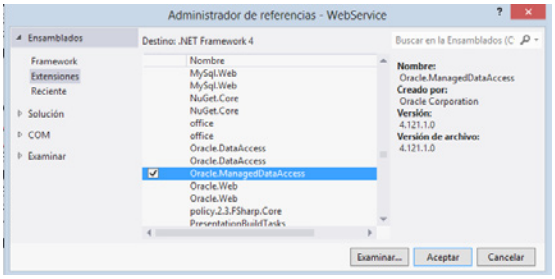


Figura 4. Agregar referencia de ODT Oracle.



Luego, se agrega un archivo de configuración del sitio web: web.config

Figura 5. Menú agregar nuevo elemento.

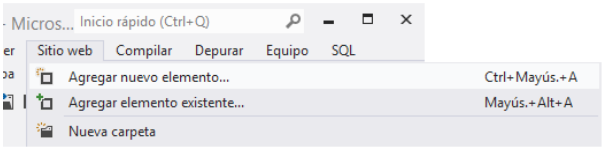
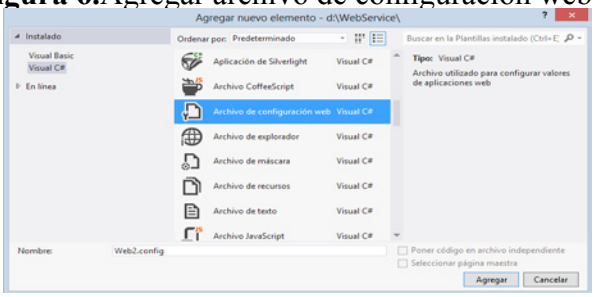


Figura 6. Agregar archivo de configuración web.



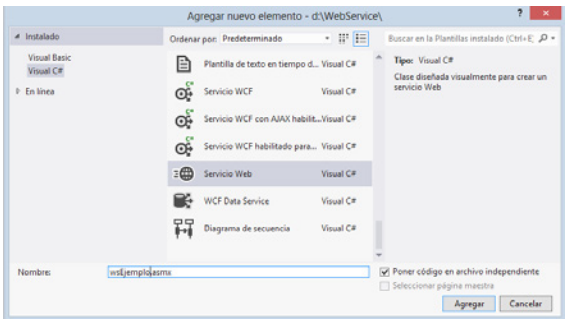
En el recientemente archivo agregado web.config se agrega las líneas de código necesarias para que se vea del siguiente modo:

Script 2. Contenido del archivo web.config

```
<?xml version="1.0"?>

<!--

  Para obtener más información sobre cómo configurar la aplicación d
e ASP.NET, visite

  http://go.microsoft.com/fwlink/?LinkId=169433

  -->

<configuration>

  <connectionStrings>

    <add name="OraConnection" connectionString="User Id=system;Pass-
word=system;Data Source=(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)
(HOST = PC-DIEGO)(PORT = 1521))(CONNECT_DATA =(SERVER = DEDICATED)
(SERVICE_NAME = XE)))" providerName="Oracle.ManagedDataAccess.Cli-
ent" />

  </connectionStrings>

   <system.web>

    <compilation debug="false" targetFramework="4.0">

      <assemblies>

        <add assembly="Oracle.ManagedDataAccess, Ver-
sion=4.121.1.0, Culture=neutral, PublicKeyToken=89B483F429C47342"/>

      </assemblies>

    </compilation>

  </system.web>

</configuration>
```

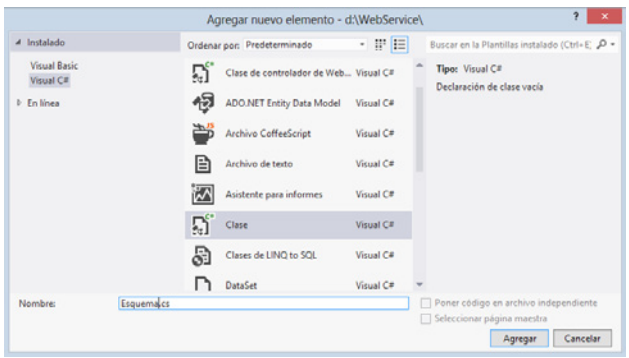
Basándose en la figura 5, se agrega un nuevo elemento Servicio Web con el nombre wsEjemplo. asmx:

Figura 7. Agregar servicio web wsEjemplo.asmx



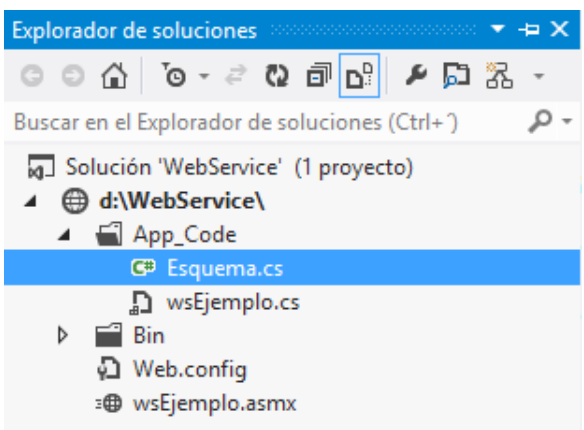
De la misma manera se añade un nuevo elemento Archivo de Clase llamado Esquema.cs

Figura 8. Agregar archivo de clase Esquema.cs



Desde el explorador de soluciones se hacer doble click en el archivo esquema.cs

Figura 9. Abrir el archivo esquema.cs



Se modifica las líneas de código necesarias para que se vea del siguiente modo:

Script 3. Contenido del archivo esquema.cs

```
using GLOBALSistemas.DataMapper;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;

public class Departamento: DataMapper
{
    public Departamento() { }
    public Departamento(DataRow row) : base(row) [31]
    public Departamento(DataTable table) : base(table) { }
    public Departamento(DataRow row, params MappDef[] mappDefs) : base(row, mappDefs) { }
    public Departamento(DataTable table, params MappDef[] mappDefs) : base(table, mappDefs) { }

    public int
        Department_Id,
        Manager_Id,
        Location_Id;

    public string
        Department_Name;
}

public class Empleado : DataMapper
{
    public Empleado() { }
    public Empleado(DataRow row) : base(row) { }
    public Empleado(DataTable table) : base(table) { }
    public Empleado(DataRow row, params MappDef[] mappDefs) : base(row, mappDefs) { }
    public Empleado(DataTable table, params MappDef[] mappDefs) : base(table, mappDefs) { }

    public int
        Employee_Id,
        Manager_Id,
        Department_Id;

    public decimal
        Salary,
        Commission_Pct;

    public string
        First_Name,
        Last_Name,
        Email,
        Phone_Number,
        Job_Id;

    public DateTime?
        Hire_Date;
}
```

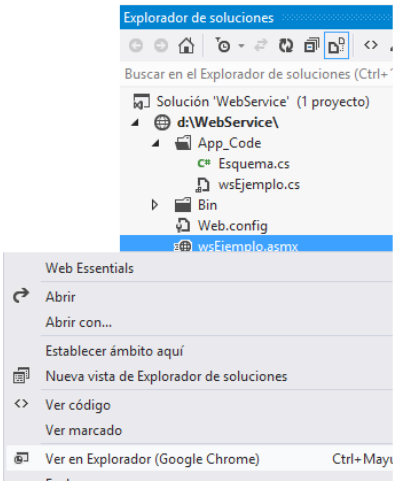
Del mismo modo se abre el archivo wsEjemplo.cs y se modifican las líneas de código para que se vea de la siguiente manera:

Script 4. Contenido del archivo wsEjemplo.cs

```
using GLOBALSistemas.GeneralTools;
using GLOBALSistemas.DataMapper;
using GLOBALSistemas.SQLHelper.Oracle;
using System.Configuration;
using System.Data;
using Oracle.ManagedDataAccess.Client;
using System.Web.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
public class wsEjemplo : WebService {

    public wsEjemplo () {}

    private string ORA_CONNECTION =
tionString;ConfigurationManager.ConnectionStrings["OraConnection"].Connec-

    [WebMethod]
    public Departamento [] select_Departamentos() {

        using (var qry = new SQLNoTransaction(ORA_CONNECTION))
        {
            // agregar parametros
            qry.Parameters.
                Add("p_result", OracleDbType.RefCursor).
                Direction = ParameterDirection.Output;

            // invocar Stored Procedure
            var r = qry.ExecSP_DataTable("HR.SELECT_DEPARTAMENTOS");

            // devolver datos mapeados a un array de objetos
            return (Departamento[])
                new Departamento(r).ToArray();
        }
    }

    [WebMethod]
    public Empleado[] select_Empleados( int department_id )
    {
        using (var qry = new SQLNoTransaction(ORA_CONNECTION))
        {
            // agregar parametros
            qry.Parameters.
                Add("p_departament_id", OracleDbType.Int32).
                Value = department_id;

            qry.Parameters.
                Add("p_result", OracleDbType.RefCursor).
                Direction = ParameterDirection.Output;

            // invocar Stored Procedure
            var r = qry.ExecSP_DataTable("HR.SELECT_EMPLEADOS");

            // devolver datos mapeados a un array de objetos
            return (Empleado[])
                new Empleado(r).ToArray();
        }
    }
}
```

Finalmente se comprueba el funcionamiento de los Web Services desde el browser:

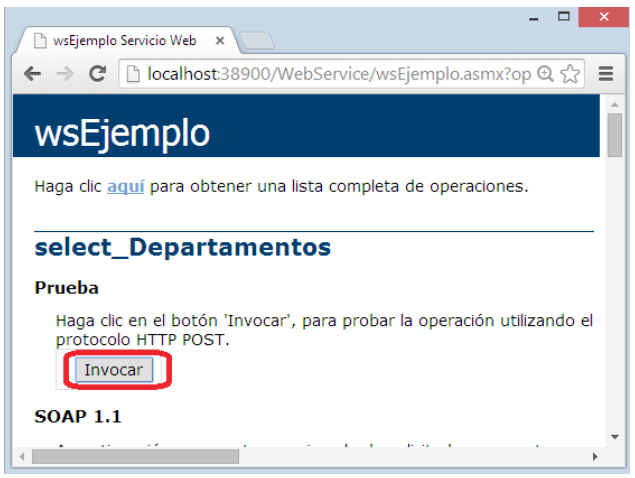
Figura 9. Menú Ver en el explorador.



Para invocar un web method, se hace click sobre el que se desea, en este caso en select_Departamentos:

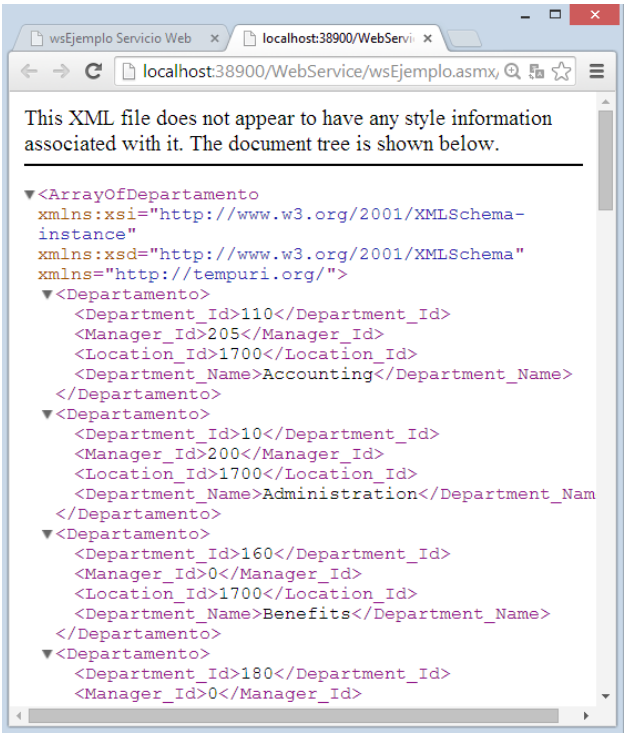
Figura 10. Web Service listo para probar desde el browser.

Figura 11. Invocar web method select_Departamentos.



Se verifica que el resultado sea como el que se muestra a continuación:

Figura 12. Resultado en formato XML de la invocación al web method select_Departamentos.



Desarrollo del consumo de los Web Services desde una aplicación Android Nativa creada con ADT y KSOAP2.Android

A continuación los fragmentos de código de programa que basándose en [31] implementan lo necesario:

Script 5. Código de programa para la invocación al Web Service.

```
1. String NAMESPACE = “http://tempuri.org/”;
2. String URL=”http://localhost/ws/wsEjemplo.asmx”;
3. String METHOD_NAME = “select_Empleados “;
4. String SOAP_ACTION = “ http://tempuri.org/select_Empleados”;
5. SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
6. request.addProperty(“id_departamento”, txtIdDep.getText().toString());
7. SoapSerializationEnvelope envelope =     new SoapSerializationEnvelope(SoapEnvelope.
   VER11);
8. envelope.dotNet = true;}
9. envelope.setOutputSoapObject(request);
10. HttpTransportSE transporte = new HttpTransportSE(URL);
11. transporte.call(SOAP_ACTION, envelope);
12. SoapPrimitive resultado_xml =(SoapPrimitive)envelope.getResponse();
13. String res = resultado_xml.toString();
14. if(res.equals(“”))
15. txtResultado.setText(“error de ejecución”);
```

Donde:

- NAMESPACE: Es el espacio de nombres utilizado en el servicio web.
- URL: Es la dirección URL para realizar la conexión con el servicio web.
- METHOD_NAME: El nombre del método web que va a ejecutar.
- SOAP_ACTION: El similar al anterior, pero en la notación definida por SOAP.

Luego se pasa el parámetro id_departamento y se construye un objeto tipo "envelope" conteniendo la solicitud de ejecución del web method del web service (Líneas 5 a 9).

Se define una variable "transporte" que luego se encarga de invocar el web method y espera que el resultado sea devuelto en "envelope" (Líneas 10 y 11).

Finalmente se reciben los resultados y se guardan en una cadena (res), sin importar la estructura que define los datos recibidos (Líneas 12 a 15).

Hasta aquí, se ha realizado la primera parte de la aplicación Android, que es solicitar la ejecución del web method y esperar el resultado. La segunda parte, y de hecho la más complicada, es interpretar los datos de resultado, que como se vio en la línea 13 del Script 5 de momento se ha guardado en la variable de cadena "res", es decir ésta contiene un texto en formato XML que hay que parsear (interpretar). Lo que significa para el caso del web method select_Empleados hay que programar una clase donde recibir los datos de un empleado con sus respectivos atributos, ejemplo: Nombres, salario, fecha ingreso, id_jefe, etc. Luego ir leyendo cada nodo XML y buscar los datos de cada atributo para cada empleado de la lista de resultados recibida. Y todo esto hay que programar de manera diferente para cada tipo de resultado devuelto por otros web method como es el caso de select_Departamentos.

Entonces a continuación el código de programa que hace lo necesario para solicitar la ejecución, recepción y parseo de resultados del web method select_Empleados.

Script 6. Código de programa para la invocación al web method select_Empleados, parseo y uso de resultados

```
1.  private class TareaWSConsulta
2.  extends AsyncTask<String,Integer,Boolean>{
3.  private Empleado[] listaEmpleados;
4.  protected Boolean doInBackground(String params){
5.          boolean resul = true;
6.          final String NAMESPACE = "http://tempuri.org/";
7.     final String URL = "http://localhost/ws/wsEmpleados.asmx";
8.     final String METHOD_NAME = "select_Empleados";
9.     final String SOAP_ACTION = "http://tempuri.org/
    select_Empleados ";
10.      SoapObject request = new SoapObject(NAMESPACE,
    METHOD_NAME);
11.      SoapSerializationEnvelope envelope = new
    SoapSerializationEnvelope(SoapEnvelope.VER11);
12.    envelope.dotNet = true;
13.     envelope.setOutputSoapObject(request);
14.     HttpTransportSE transporte = new HttpTransportSE(URL);
15.     try   {
16.      transporte.call(SOAP_ACTION, envelope);
17.        SoapObject resSoap =
    (SoapObject)envelope.getResponse();
18.        listaEmpleados = new
    Empleado[resSoap.getPropertyCount()];
19.        for (int i = 0; i < listaEmpleados.length; i++)       {
20.           SoapObject ic = (SoapObject)resSoap.getProperty(i);
21.            Empleado cli = new Empleado();
22.           cli.id = Integer.parseInt(ic.getProperty(0).toString());
23.           cli.nombre = ic.getProperty(1).toString();
24.           cli.telefono =
    Integer.parseInt(ic.getProperty(2).toString());
25.             listaEmpleados[i] = cli;
26.         }
27.    }
28.    catch (Exception e)    {
29.      resul = false;
30.    }
31.       return resul;
32.   }
33.    protected void onPostExecute(Boolean result) {
34.       if (result)       {
35.       final String[] datos = new String[listaEmpleados.length];
36.        for(int i=0; i<listaEmpleados.length; i++)
37.          datos[i] = listaEmpleados[i].nombre;
38.        ArrayAdapter<String> adaptador =  new
    ArrayAdapter<String>(MainActivity.this,
    android.R.layout.simple_list_item_1, datos);
39.        lstEmpleados.setAdapter(adaptador);
40.       }
41.       else      {
42.         txtResultado.setText("Error!");
43.       }
44.    }
45. }
46. btnConsultar.setOnClickListener(new OnClickListener() {
47.      @Override    public void onClick(View v) {
48.       TareaWSConsulta tarea = new TareaWSConsulta();
49.         tarea.execute();
50.    }
51. };
52. @Override
53.  public Object getProperty(int arg0) {
54.      switch(arg0)
55.       {       case 0:
56.          return id;
57.       case 1:
58.          return nombre;
```

Desarrollo del consumo de los Web Services desde una aplicación Android Nativa creada con B4A y HTTUtils2 y XML SAX.

Basados en los tutoriales de [29], a continuación el código de programa que implementa la solución deseada.

Script 7. Invocación al web method select_Empleados, parseo y uso de resultados con B4A.

```
1.  Sub Activity_Create(FirstTime As Boolean)
2.  Dim job1 As HttpJob
3.  Dim Envelope As String = “aquí el texto xml del envelope”
4.  job1.Initialize(“Job1”, Me)
5.  job1.PostString(“http://localhost/ws/wsEjemplo.asmx”,”id_departamento=” & txtIdDep.Text)
6.  job1.GetRequest.SetHeader(“SOAPAction”, “http://tempuri.org/select_Empleados”)
7.  job1. GetRequest.SetContentType(“text/xml;charset=UTF8”)
8.  End Sub
9.  Sub JobDone (Job As HttpJob)
10. If Job.Success = True Then
11. Select Job.JobName
12. Case “Job1”
13.    Dim res as String = job1.RespoinseString
14. End Select
15. Else
16. Log(“Error: “ & Job.ErrorMessage)
17. End If
18. Job.Release
19. End Sub
20. Sub Parser_EndElement (Uri As String, Name As String, Text As StringBuilder)
21.    If parser.Parents.IndexOf(“Empleado”) > -1 Then
22.       If Name = “Nombres” Then
23.          Nombres = Text.ToString
24.       Else If Name = “Salario” Then
25.          Salario = Text.ToString
26.       End If
27.    End If
28.    If Name = “Mgr” Then
29.       ListView1.AddSingleLine2(Nombres, Salario)
30.    End If
31. End Sub
```

Para aplicar las métricas seleccionadas, hay que cuantificar las líneas de código fuente para cada caso, y la presencia de subrutinas, así tenemos:

Caso 1: Usando ADT y KSOAP2.Android

Total líneas de código fuente: 101

Total líneas de código fuente escrito: 101

Total subrutinas: 7

Total invocación de las subrutinas: 7

Caso 2: Usando B4A, HTTPutilsS2 y XML SAX

Total líneas de código: 31

Total líneas de código fuente escrito: 31

Total subrutinas: 3

Total invocación de las subrutinas: 3

Fórmula para el cálculo de la Métrica de Tamaño de Producto:

$$TP = TLCF \qquad (1)$$

Donde:
TLFC = Total líneas de código fuente

TP = Tamaño de producto
Fórmula para el cálculo de la Métrica de Productividad de Desarrolladores cuantificando el código escrito:

$$PD1 = 1 / TLCE \qquad (2)$$

Donde:

PD1 = Productividad de desarrolladores cuantificando el código escrito.

TLCE = Total líneas de código escrito

Fórmula para el cálculo de la Métrica de Reusabilidad:

$$R = TIS / TS \qquad (3)$$

Donde:

R = Reusabilidad

TS = Total subrutinas

TIS = Total de Invocación de subrutinas

Fórmula para el cálculo de Métrica de Tiempo de Desarrollo:

$$TD = TLCE * T \qquad (4)$$

Donde:

TLCE = Total líneas de código escrito

T = Tiempo promedio invertido en la escritura de cada línea de código fuente.

TD = Tiempo de desarrollo

Aplicando las formulas (1), (2), (3) y (4) al Caso 1 y Caso 2 tenemos los siguientes valores:

Tabla 1. Cálculo de fórmulas para Caso 1 y Caso 2.

	Caso 1	Caso 2
Fórmula (1) Tamaño de Producto	101	31
Fórmula (2) Productividad cuantificando el código escrito.	0.010	0.033
Fórmula (3) Reusabilidad	1	1
Fórmula (4) Tiempo desarrollo (Se toma en cuenta 10 seg. para T)	1010	310

Nótese que el Tiempo de Desarrollo es mayor para el Caso 1 versus Caso 2.

Por otro lado, nótese que en ambos casos casi todo el código se dedica a solucionar el consumo de un único web method, lo que significa que si fueran "n" consumos diferentes, habría que escribir casi TLCF * n líneas de código. Solución: se propone crear una librería propia que minimice el tiempo de desarrollo en B4A basados en HTTPUtils2.

Desarrollo de una librería propia que minimice el tiempo de desarrollo con B4A y la librería HTTPUtils2.

La estrategia es crear un software que de manera automática genere código para reducir el valor de TLCE, además de una librería para B4A que reduzca aún más el valor de TP. De esta manera el TD se reduciría a lo más mínimo. Al reducir el TLCE se amplía la productividad, en consecuencia se reduce el tiempo de desarrollo.

El software que generará código de manera automática se encargará de analizar el WSDL de los servicios web, es decir descifrará la información publicada que describe la estructura de invocación y recepción de resultados de los web services, a este proceso se denomina "Descubrimiento de Servicios Web", de allí que WSDL signifique Web Services Discovery Languaje, aunque otros en lugar de Discovery utilizan el término Description. Con esto, el software a fabricarse podrá crear de manera automática archivos de "clases" para B4A que permitan recibir los datos de resultados.

La librería para B4A se basará en HTTPutils2 para aprovechar su funcionalidad, permitiendo mayor reusabilidad para reducir el tamaño de producto.

El código fuente tanto del software para analizar WSDL como de la librería para B4A, una vez desarrollado resultó muy extenso como para mostrar en este documento, sin embargo se encuentra disponible para toda audiencia si lo solicitan al email del autor.

Ahora, a continuación se muestra el código de programa de lo que sería el "Caso 3" que es la aplicación Android que consume el web service pero ya con código generado de manera automática y una librería que minimice el código escrito.

Script 8. Programa B4A para consumir web services con código generado y uso de librería propia.

```
1.  Sub Process_Globals
2.  Private ws As WebServiceRequest
3.  End Sub
4.  Sub Globals
5.  Private ListView1 As ListView
6.  Private Spinner1 As Spinner
7.  End Sub
8.  Sub Activity_Create(FirstTime As Boolean)
9.  Activity.LoadLayout("Layout1")
10. If FirstTime Then
11. ws.Initialize("wsEjemplo", Me)
12. End If
13. Dim params As Map
14. params.Initialize()
15. ws.InvokeWebMethod("job1", "select_Departamentos",
    params)
16. End Sub
17. Sub WebServiceDone(Response As WebServiceResponse)
18. If Response.Success Then
19. Select Response.JobName
20. Case "job1"
21. Dim lst As List = Response.ResponseObject
22. Spinner1.Clear
23. For Each dep As Departamento In lst
24. Spinner1.Add( dep.Department_Id & " - " & dep.Depart-
    ment_Name )
25. Next
26. End Select
27. Else
28. Msgbox(Response.ErrorMessage, "Error")
29. End If
30. End Sub
```

Como puede observarse, se ha reducido de 31 a 30 líneas de código, y solamente se invoca a un único web method. Pero si se desea invocar además a select_Empleados, por ejemplo cuando en

el dispositivo móvil se elija un departamento, entonces solo hay que añadir las siguientes líneas de código:

Script 9. Líneas de código adicionales para consumir web method select_Empleados.

```
' Estas líneas insertar en la línea # 26 del script 8.
1. Case "job2"
2. Dim lst As List = Response.ResponseObject
3. ListView1.Clear()
4. For Each emp As Empleado In lst
5. ListView1.AddTwoLines( emp.Job_Id , emp.First_Name )
6. Next
'' Estas líneas insertar al final del script 8.
7. Sub Spinner1_ItemClick (Position As Int, Value As Object)
8.     Dim id_dep As Int = Utils.Substring4( Value, "-", 0)
9.     Dim params As Map
10.    params.Initialize()
11.    params.Put("department_id", id_dep)
12.    ws.InvokeWebMethod("job2", "select_Empleados", params)
13. End Sub
```

Como puede apreciarse, tan solo 13 líneas de código adicional satisfacen el consumo de otro web method. Pero en el Caso 2 serían alrededor de 25 líneas. Con estos datos veamos entonces el cálculo de las métricas propuestas y aplicadas a los Casos 2 y 3 pero tomando en cuenta el consumo de los dos web method.

Caso 2: Usando B4A, HTTPutilsS2 y XML SAX
Total líneas de código: 31+25=56
Total líneas de código fuente escrito: 31+25=56
Total subrutinas: 3+2=5
Total invocación de las subrutinas: 3+2=5

Caso 4: Usando B4A, código autogenerado y librería propia
Total líneas de código fuente: 30+13=43
Total líneas de código fuente escrito: 30+13=43
Total subrutinas: 4+1=5
Total invocación de las subrutinas: 4+1=5

Tabla 2. Cálculo de fórmulas para Caso 2 y Caso 3.

	Caso 2	Caso 3
Fórmula (1) Tamaño de Producto	56	43
Fórmula (2) Productividad cuantificando el código escrito.	0.018	0.023
Fórmula (3) Reusabilidad	1	1
Fórmula (4) Tiempo desarrollo (Se toma en cuenta 10 seg. para T)	560	430

Nótese que el Tiempo de Desarrollo es mayor para el Caso 2 versus Caso 3.

Resultados

Con la aplicación del software que analiza el WSDL y genera automáticamente código fuente, así también con una librería B4A que optimicen la cantidad de líneas de código escrito, se logra demostrar que el Tiempo de Desarrollo se minimiza drásticamente, tanto para el consumo de un único web method como para el consumo de varios de ellos. Véase el siguiente cuadro comparativo:

Tabla 3. Cuadro comparativo de Cálculo de fórmulas para Casos 1, 2 y 3

	Caso 1	Caso 2	Caso 2	Caso 3
Número de Web Method's	1	1	2	2
Fórmula (1) Tamaño de Producto	101	31	56	43
Fórmula (2) Productividad cuantificando el código escrito.	0.010	0.032	0.018	0.023
Fórmula (3) Reusabilidad	1	1	1	1
Fórmula (4) Tiempo desarrollo (Se toma en cuenta 10 seg. para T)	1010	310	560	430
Tasa de reducción de tiempo frente al valor más demoroso	0%	69.31%	0%	23.21%

En el presente análisis, cuando se consume un único web method, el Caso 1 frente al Caso 2, tiene un tiempo de desarrollo más elevado, nótese que el Caso 2 es 69.31% más rápido. Por tanto, es más difícil escribir código en ADT que con B4A

La relación del Caso 2 frente al Caso 3, cuando se consumen dos web method, tiene un tiempo de desarrollo más elevado, nótese que el Caso 3 es aún 23% más rápido que el anterior. Por tanto, es más difícil escribir código en Caso 2 que con Caso 3.

Conclusiones

Se logró reducir sustancialmente el tiempo de desarrollo en el consumo de Servicios Web en el formato XML SOAP, en un primer experimentos se redujo hasta en un 69.31%, y en otro segundo experimento se redujo un 23.21% adicional y con el doble de operaciones de consumo.

En el presente análisis, cuando se consume un único web method, el Caso 1 frente al Caso 2, a pesar de tener el mismo valor de reusabilidad, no logra reducir la complejidad del código fuente como lo hace el Caso 2. Por tanto, se concluye que es más difícil escribir código en ADT que con B4A.

Un símil sucede en la relación del Caso 2 frente al Caso 3, cuando se consumen dos web method, a pesar de tener el mismo valor de reusabilidad, no logra reducir la complejidad del código fuente como lo hace el Caso 3. Por tanto, se concluye que es más difícil escribir código en Caso 2 que con Caso 3.

Con esto, la Productividad de Desarrolladores también mejora así como el Tamaño de Producto cada vez se contrae.

Nótese que si mayor fuera el valor de reusabilidad, menor debería ser el tamaño de producto, por tanto, mayor debe ser la productividad de código escrito. Luego, a mayor productividad, menor es el tiempo de desarrollo.

Agradecimientos

Un reconocimiento a las autoridades de la Universidad Técnica del Norte por todo el contingente recibido durante la presente investigación.

Un agradecimiento especial a la Facultad de Ingeniería en Ciencias Aplicadas y sus directivos por el compromiso demostrado con el presente estudio.

De manera muy personal, una gratitud al Dr. Phd Julio Armas, por su total y desmesurado apoyo durante todo el tiempo, sin él y sus revisiones, este proyecto no hubiese llegado a su meta.

Bibliografía

[1] S. d. T. d. Ecuador. (2014). Estadísticas de telefonía fija y móvil, acceso a internet, cibercafés, televisión pagada y Servicios Portadores. Available: http://www.supertel.gob.ec/index.php?option=com_k2&view=item&id=21:servicios-de-telecomunicaciones&Itemid=90

[2] I.-I. d. E. y. C. d. Ecuador. (2014). Resultados Censo 2010. Available: http://www.ecuadorencifras.gob.ec/resultados/

[3] D. León Santín, F. Carrillo Díaz, C. M. Yépez, and B. P. Ramos, „ANÁLISIS DE LOS NIVELES DE PENETRACIÓN DE LOS SERVICIOS DE TELECOMUNICACIONES MÁS DESTACADOS QUE SE OFRECEN EN EL ECUADOR „ p. 9, 2010.

[4] M. C. Gasca Mantilla, L. L. Camargo Ariza, and B. Medina Delgado, „Metodología para el desarrollo de aplicaciones móviles. (Spanish),“ Methodology for mobile application development. (English), vol. 18, pp. 20-35, 04//abr-jun2014 2014.

[5] P. Blanco, J. Camarero, A. Fumero, A. Werterski, and P. Rodríguez, „Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone,“ *Universidad Politécnica de Madrid. Documento recuperado de http://www.adamwesterski.com/wp-t/files/docsCursos/Agile_doc_TemasAnv.pdf,* 2009.

[6] C. A. Vanegas, „Desarrollo de aplicaciones sobre Android,“ *Vínculos,* vol. 9, pp. 129-145, 2013.

[7] M. Moser, „Open Source Android Development Tools," *[Online][Cited: Julio, 2011.],* 2011.

[8] M. Li, G. Lei, and W. Jin, „Research and Development of Mobile Application for Android Platform," *International Journal of Multimedia & Ubiquitous Engineering,* vol. 9, pp. 187-198, 2014.

[9] s. t. inc, „ksoap2-android," 3.3.0 Release, 2014-05-09 ed: simpligility technologies inc, 2014, pp. The ksoap2-android project provides a **lightweight and efficient SOAP client library** for the Android platform.

[10] kObjects.org, „kXML on Android," 2 ed: kObjects.org, 2014, pp. kXML is a small XML pull parser, specially designed for constrained environments such as Applets, Personal Java or MIDP devices. In contrast to kXML 1, kXML 2 is based on the common XML pull API.

[11] H. Fang, J. Chen, and B. Xu, „The Interaction Mechanism based on JSON for Android Database Application," *Information Technology Journal,* vol. 12, pp. 224-228, 2013.

[12] K. J. Carvajal Valdivieso, S. Suntasig, and C. Fernando, „Análisis comparativo entre las plataformas de desarrollo de aplicaciones móviles para los sistemas operativos Android y Ios," 2013.

[13] M. C. Jose, R. E. C. Valencia, and R. N. M. Castro, „Entornos para el desarrollo de aplicaciones móviles," *Vínculos,* vol. 9, pp. 146-156, 2013.

[14] C. Aldea, „JAX-WS LOGIN WEB SERVICE AND ANDROID CLIENT," *Bulletin of the Transilvania University of Brasov, Series III: Mathematics, Informatics, Physics,* vol. 6, pp. 51-60, 2013.

[15] P. Pocatilu, „Developing Mobile Learning Applications for Android using Web Services," *Informatica Economica,* vol. 14, pp. 106-115, 2010.

[16] V. Monfort and S. Cherif, „Bridging the Gap between Technical Heterogeneity of Context-Aware Platforms: Experimenting a Service Based Connectivity between Adaptable Android, WComp and OpenORB," *International Journal of Computer Science Issues (IJCSI),* vol. 8, pp. 1-12, 07// 2011.

[17] M. Singhal and A. Shukla, „Implementation of Location based Services in Android using GPS and Web Services," *International Journal of Computer Science Issues (IJCSI),* vol. 9, pp. 237-242, 01// 2012.

[18] A. Radovici and G. Dragoi, „WEB SERVICES ENGINE FOR THE ANDROID PLATFORM," *Annals of DAAAM & Proceedings,* pp. 669-670, 2010.

[19] B. Prajapat and M. Shrivastava, „Performance Evaluation Using J2ME with Android over Cloud Services: A Simulation Approach," *International Journal of Advanced Computer Research,* vol. 3, pp. 14-17, 2013.

[20] O. Dospinescu and M. Perca, „Web Services in Mobile Applications," *Informatica Economica,* vol. 17, pp. 17-26, 2013.

[21] S. Chakaravarthi and K. Selvamani, „A FLEXIBLE WEB SERVICE ENVIRONMENT WITH HIGH RANGE OF SECURITY AND QoS MECHANISMS," *Journal of Theoretical & Applied Information Technology,* vol. 61, pp. 60-66, 2014.

[22] J. Barranco Mesa, „SML Android Satélite," 2012.

[23] A. de la Nuez Romero, „ÁGORA, Sistema de Seguimiento de Tareas para Android," 2013.

[24] Y. Hyeon-Ju, „A Study on the Performance of Android Platform," *International Journal on Computer Science & Engineering,* vol. 4, pp. 532-537, 04// 2012.

[25] D. Sánchez, „El Primer Paso a las Aplicaciones Android," in *Congreso Nacional de Tecnologías de la Información y Comunicación: CONATIC 2013*, 2013, p. 46.

[26] L. Delía, N. Galdamez, P. J. Thomas, and P. Pesado, „Un análisis experimental tipo de aplicaciones para dispositivos móviles," in *XVIII Congreso Argentino de Ciencias de la Computación*, 2013.

[27] G. Inc. (2014). *Android Developers*. Available: http://developer.android.com/index.html

[28] W3C, „SOAP Version 1.2 Part 0: Primer (Second Edition)," in *W3C Recommendation 27 April 2007*, ed: W3C, 2007.

[29] A. Software, „B4A Basic for Android," 3.82 ed, 2014, p. Basic4android is the simplest and most powerful Rapid Application Development (RAD) tool available for the Android platform.

[30] J. Masner, J. Vaněk, and M. Stočes, „Spatial Data Monitoring and Mobile Applications - Comparison of Methods for Parsing JSON in Android Operating System," *Agris On-Line Papers in Economics & Informatics,* vol. 6, pp. 37-46, 2014.

[31] S. Gómez Oliver, „Acceso a Servicios Web SOAP en Android (2/2)," in *sgoliver.net blog*, ed. 27/02/2012, 2014.

INFORMÁTICA