Shared Web Service Architecture for Websites with Related Mobile Applications

Waseem I. Bader
Al-Salt College for Human Sciences
Al-Balqa Applied University, Al-Salt, Jordan

ABSTRACT
Mobile application development is growing rapidly in the last few years that it hasn't become solely for big named brands and companies, Today more and more small and midsize businesses and institutions are following the trend of providing mobile based applications to support their work and popularity. This process has had great benefits on the growth of businesses and institutions, but in the same time has come with some burdens to overcome for the developers and programmers of such applications to ensure the consistency between the original business website functionality and the extended mobile applications that supports the websites. In this paper, a proposed architecture for a business employee services website and its supporting mobile application is presented which uses a single web service to support both applications, providing all the functionality needed to ensure consistency between them while at the same time reducing the time and effort needed for programmers and developers to maintain both of them in the best possible way. The code samples in this paper were implemented using the ASP.net programming language for the web service and web site and android programming for the mobile application, but the same architecture can be implemented in any other development language and environment.

Keywords
Web Development, Mobile Development, Web Services

1. INTRODUCTION
Application development has grown quickly in the last few years, everyday new technologies and ideas are adopted to fulfill the growing needs of businesses and institutions, and in the same time to provide their users and clients with the best possible experience. Developers today are very well familiar with developing websites that support the different services of a company or a business, but during the last few years mobile development has become very important and required by most companies and businesses because of the benefits it brings to the growth and popularity of their work and widespread. Although Mobile Development has grown fast and became very popular among application developers, but in the same time it cannot replace the ordinary web site that users use every day. This has forced developers of a business to support both websites and mobile applications for the business at the same time and keep them both at sync in the best possible way. [1]

In earlier web development, developers didn't often need to worry much about the consistency of their code in websites because they simply provide the code within a website and they can change it any time they need in case of new upgrades or fixes. But with the introduction of Mobile Development and with the increasing demand of providing Mobile based applications for businesses, which is ever growing in popularity as new statistics reveal that the internet usage worldwide by mobile and tablet devices exceeded that of desktop for the first time in October 2016 **[2]**, developers started to think more and more about the code they develop in both web and mobile applications.

Nowadays the business trend is to allow users to access the same services from their websites or their mobile applications, and application developers have to guarantee, as much as possible, to provide the best possible experience and the most consistent information for business clients at any time and from any application they use.

An Architecture of a shared business employee's web service that supports both an employee services website and its related mobile application is presented and discussed in details to suggest the best techniques that can be implemented to provide as much consistency, flexibility and effectivity as possible for developers in such a development architectural scenario, as shown in the figure:

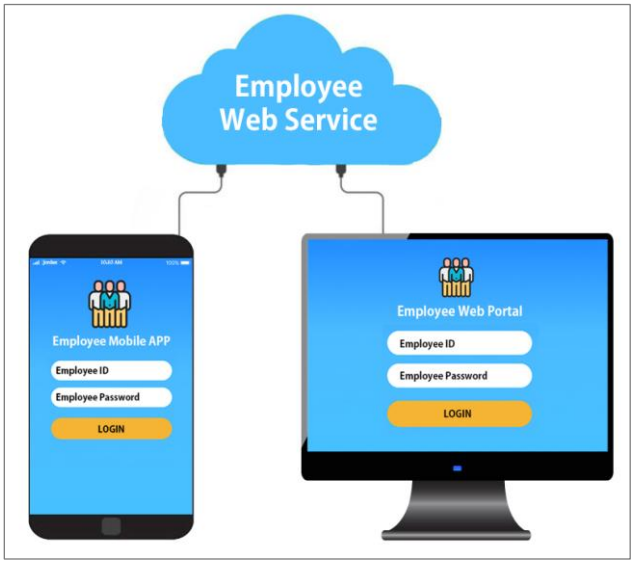

Figure 1: Employee Web Service, Website and Mobile Application Architecture

2. WEB SERVICE ARCHITECTURE
A web service is a piece of software that makes itself available over the internet and uses a standardized XML messaging system to provide a method of communication between different applications that might use different programming languages or development environments **[3]**. It can be used to provide the programming logic base and application specific information among websites and their related mobile applications by providing the shared services and coding between them.

A web service is basically a web application which is generally a class containing a list of web methods that could be used by other applications. The web service should be deployed on a web server with a specific URL address in order for other applications to be able to reach the web service and consume its available web methods **[4]**. In case of a web service built with ASP.net, it will have a URL similar to the following:

http://www.myBusinessDomain.com/myBusinessWebServices/EmployeeWebService.asmx

Each supported functionality that the web service should provide will be presented as separate web method that any external application can access by passing the appropriate parameters to the web method and receiving the corresponding return value from the web service to the accessing application. The following figure shows the structure of an Employee Web Service with a sample web method called **isValidLogin** that receives two string parameters corresponding to the employee Id and password, and then the method would return a Boolean value corresponding to a valid or invalid login:

```
namespace myBusinessWebServices
{
  [WebService(Namespace = " http://www.myBusinessDomain.com
")]
public class EmployeeWebServices:
System.Web.Services.WebService
  {
    [WebMethod]
    public Boolean isValidLogin ( String empId, String pass )
    {
      Boolean valid = …
//the code to check the validation of the user login would be placed
here
      return valid;
    }
  }
}
```

Figure 2: Employee Web Service Structure

The returned value of a web method can vary depending on the functionality of the service provided by that method, it can be as simple as a single string, integer or even a Boolean true/false value as shown in the previous example. But in some case these simple return values are not enough, because an application might need to receive complex data structures from the web method to be displayed or to be further manipulated by the application. There are mainly two common techniques to pass or receive complex data structures from the web service method:

- User-Defined Structures.
- JSON

2.1 User-Defined Structure Return Value

In this technique there must be an agreed standard to be used among the web service and all of the applications that will use it, for example, if a set of records must be returned as key/value pairs, a semi-comma separated string of the key/value pairs must be returned and each calling application must divide the pairs based on the agreed separation symbol and use it as needed, as shown in the following example:

Key1=Value1;Key2=Value2;….; KeyN=ValueN

2.2 JSON

JSON (JavaScript Object Notation) is based on key-value data exchanging format. It has a simple structure and lightweight format, JSON is quite easy for humans to read and write, as well as for applications to generate and parse, most programming languages provide different libraries for using JSON objects. [5]

A JSON object looks something like this:

```
{
 "empid": "24220",
 "empname": "Omar Ayouni",
 "job": "Programmer",
 "isManager" : true,
 "salary": 3250,
  …
 }
```

Figure 3: JSON Object Format

A JSON object can contain other nested JSON objects within which resembles an Array, Vector or Hashtable in programming languages.

A nested JSON object looks something like this:

```
{
 "24220" : {
    "empname": "Omar Ayouni",
    "job": "Programmer",
    "isManager" : true,
    "salary": 3250,
  …
 },
 "29795" : {
    "empname": "Ayat Hayati",
    "job": "Supervisor",
    "isManager" : false,
    "salary": 4375,
  …
 },
 …
 ,
 "empidN" : {
    "empname": "EmployeeN",
    "job": "JobN",
    "isManager" : N,
    "salary": SalN,
  …
 }
}
```

Figure 4: Nested JSON Object Format

Using the JSON format in the communication between the web service and the applications that access it has more advantages than using the User-Defined format technique, because in the User-Defined format any changes in the format of the data will force the developers to change the way they consume it in every application that uses the web service, for example if the separation symbol is changed in the returned string format we need to change that in every single application that uses this web method and this is very time and effort consuming, besides that all the mobile application clients should update their apps to reflect these new changes which is also time, effort and expense consuming on behalf of the clients as well. On the other hand using JSON is standard for the web service and the different applications regardless of the programming language they are built on, because there is no need to hardcode any user-defined formats within it.

For example to prepare a web method in the web service that would return the detailed information of a specific employee ID, the following code can be used:

```
//User-Defined EmpInfo Class
private class EmpInfo
{
    String empid;
    String empname;
    String job;
    Boolean isManager;
    double salary;
}

 [WebMethod]
public String getEmployeeInfo ( String empId )
{
EmpInfo emp = new EmpInfo ( );
//Read Emplyee info from Database or external file and fill the
//UserInfo Object members

emp.empid = "24220";
emp.empname = "Omar Ayouni";
…
return new
System.Web.Script.Serialization.JavaScriptSerializer().Serialize(emp)
;
}
```

Figure 5: Employee Web Service getEmplyeeInfo Web Method

In the previous code the ASP.net built-in **JavaScriptSerializer** class **Serialize** method is used to convert the **EmpInfo** object into a JSON string to be returned from the web method in the following structure: **[6]**

```
{
 "empid": "24220"
 "empname": "Omar Ayouni",
 "job": "Programmer",
 "isManager" : true,
 "salary": 3250,
 }
```

Figure 6: EmpInfo JSON Object Format returned from the getEmplyeeInfo Web Method

Any consuming application that uses the **getEmplyeeInfo** web method should later **deserialize** the JSON object to use it as it needs.

2.3 Consuming the Web Service from the Website

In order to use the web service from an ASP.net website, a reference to the Web Service must be added in Visual Studio by using the URL address of the desired web service **[7]**. By right clicking the project in the solution explorer and choosing "Add Service Reference" option from the context menu as shown in the following figure:

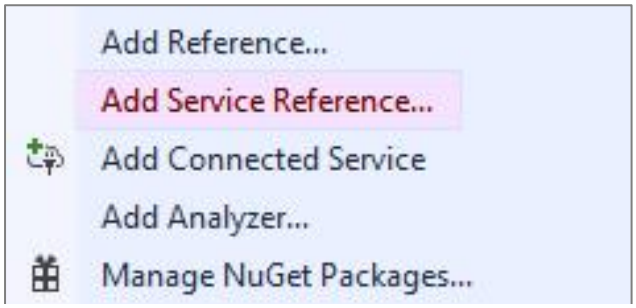

Figure 7: Adding a reference to a web service in Visual Studio

In the **Advanced** section of the **Add Service Reference** Window, the **Add Web Reference** button should be used and the web service URL address and reference name in the website should be set before adding the web service reference as shown in the following figure:

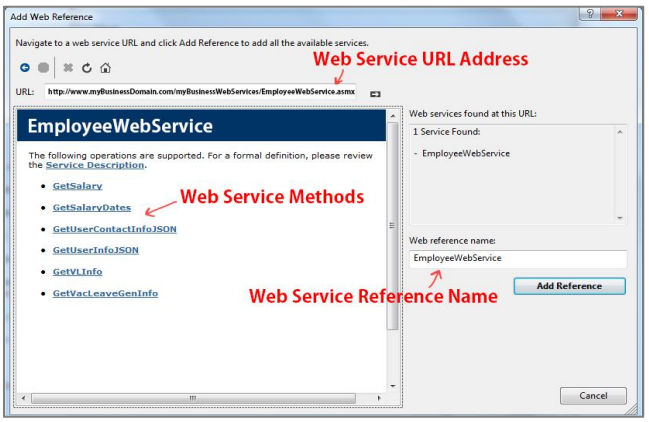

Figure 8: Adding a reference to a web service in Visual Studio

After adding the web service reference in the ASP.net website project, the reference name set in the previous window will be used to access the web service methods in the required web pages of the web site as shown in the following example:

```
EmployeeWebService empws = new EmployeeWebService ( );
String jsonResponse = empws.getEmployeeInfo ("24220"); //empId
Hashtable jsonHT = new
System.Web.Script.Serialization.JavaScriptSerializer().Deserialize<H
ashtable>(jsonResponse);
String empname = jsonHT["empname"].ToString( );
String job = jsonHT["job"].ToString( );
String salary = jsonHT["salary "].ToString( );
…
```

Figure 9: Consuming getEmplyeeInfo from the website

The **Hashtable** class used in the previous sample code is a built-in C# class that consist of a tabled structure with every entry consisting of a key/value pair. The Hashtable structure can be used to convert the JSON string returned from the web method into an ASP.net object in order to access its values or even iterate in the returned key/value pairs, as shown in the following figure:

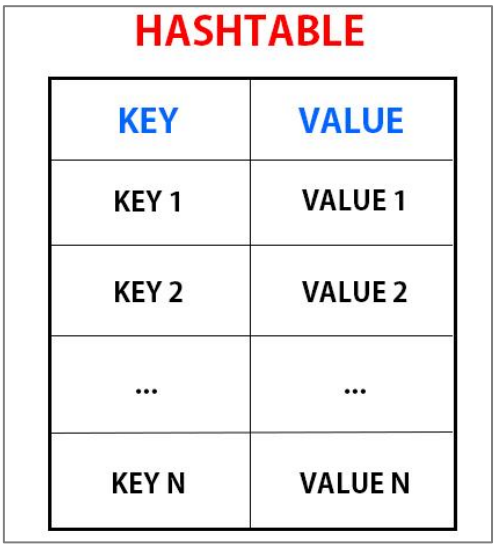

Figure 10: HashTable Structure

A HashTable object also allows the storage of complex structures in its key/value entries which corresponds to the nested objects in JSON. **[8]**

Other Structures can also be used to deserialize JSON strings such as Dictionaries, Hashmaps, Lists and others.

2.4 Consuming the Web Service from the Mobile Application

To use the web service from an Android mobile application, any Soap supporting build-in or third-party libraries can be used **[9]**. For Example, **KSoap2** is a lightweight open source library that can be used to interoperate with most popular SOAP engines and hence can be used to access the employee web service in hand. **[10]**

In order to access a web method in the web service a **SoapObject** instance must be created with the web service URL address, method name and its corresponding parameter names and values, in addition to the use of an **HttpTransport** and **SoapSerializationEnvelope** objects as showing in the following code sample:

```
SoapObject soapRequest = new
SoapObject("http://www.myBusinessDomain.com/", "getEmployeeInfo");

//Adding the corresponding parameters to the Soap request
PropertyInfo pi = new PropertyInfo();
pi.setName("empId");
pi.setValue("24220"); //empId
pi.setType(String.class);
soapRequest.addProperty(pi);

SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
SoapEnvelope.VER11);

envelope.dotNet = true;
envelope.setOutputSoapObject(soapRequest);

HttpTransportSE httpTransport = new
HttpTransportSE("http://www.myBusinessDomain.com/myBusinessWebServic
es/EmployeeWebService.asmx");

Object jsonResponse = null;
try {
    httpTransport.call("http://www.myBusinessDomain.com/getEmployeeInfo",
envelope);
    jsonResponse = envelope.getResponse(); //JSON Response is Received
Here
} catch (Exception exception) {
// Deal with any Exceptions here
        }
…
```

Figure 10: Accessing a web service method from an Android Application

After receiving the JSON string response from the web service method, it should be converted back into objects that Android can deal with. There is also many structures and libraries that can be used to manipulate the JSON String, in this sample code another open source JSON Library is used to deal with the received JSON String as shown:

```
JSONObject jsonObj =new JSONObject(jsonResponse.toString());
String empname = jsonObj.getString("empname");
String job = jsonObj.getString("job");
String salary = jsonObj.getString("salary");
…
```

Figure 11: Dealing with JSON String in an Android Application

2.5 Web Service Form Validation & Manipulation

One of the most common services that need to be available in either a website or a mobile application, besides retrieving information from the server and displaying it to the user as seen in the previous examples, is that ability to submit form data from the client to the server. In this operation, information sent by the client must be validated and manipulated by the server whether the data is sent from the business website or mobile application.

For Example, if we have a service that allows business employees to submit their absences from work online, an employee should be able to do so from the company website or mobile application as shown in the following figures:

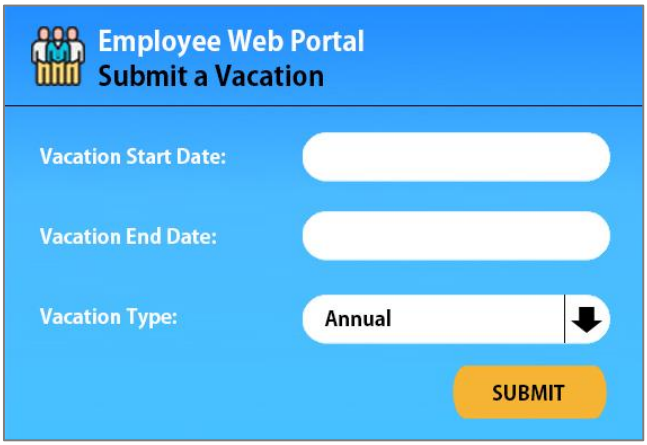

Figure 12: Employee Vacation Website Screen

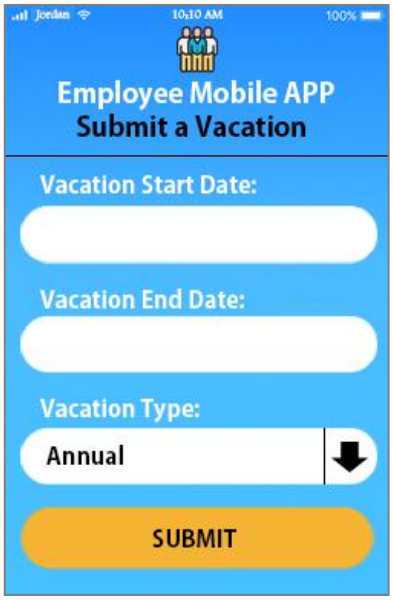

Figure 12: Employee Vacation Mobile Screen

In order to provide as much consistency for such client forms, a web service method will be provided in the Employee Web Service which will be responsible of validating the client input information and manipulating the data. After processing the client input, a suggested response structure, based on Hashtables, is returned to the calling application as shown in the following code samples:

```
 [WebMethod]
public String submitVacation ( String empId , String VacationStartDate, String
VacationEndDate, String VacationType )
{
Boolean valid = true;
Hashtable responseTable = new Hashtable( );

if (VacationStartDate.Equals(""))
{
    valid = false;
```

```
    responseTable.Add("error_VacStartDate", "Empty Vacation Start Date");
}
if (VacationEndDate.Equals(""))
{
    valid = false;
    responseTable.Add("error_VacationEndDate ", "Empty Vacation End
Date");
}

if ( valid )
{
       Boolean dataInserted = false;
       //Perform Insert Data into a Database or External Storage or File
       …
       if ( dataInserted )
       {
              responseTable.Add("success", "Vacation Submitted Successfully.");
       }
       else
       {
              responseTable.Add("error_InsertData", "Error Inserting Data");
       }
}

return new
System.Web.Script.Serialization.JavaScriptSerializer().Serialize(responseTable)
;
}
```

Figure 13: Employee Web Service submitVacation Web Method

The **submitVacation** web service method prepares an empty Hashtable before performing the data validation and manipulation, in case of any invalid data or error, a new entry in the generated Hashtable is added with a key starting with the prefix **error_** and then a descriptive name of the invalidation or error is followed. In case of valid inputs and successful manipulation of the data, a record is added in the Hashtable with a **success** key and a descriptive value to be displayed later to the client. The Hashtable is then returned back to the calling application whether it was a website or a mobile application as a serialized JSON string.

The Following code samples shows how to deal with the **submitVacation** web method returned JSON string in the calling ASP.net website and Android Mobile Application:

```
EmployeeWebService empws = new EmployeeWebService ( );
String jsonResponse = empws.submitVacation ("24220","01-01-
2021","02-01-2021","Annual");
//The previous sample data would be entered by the client

Hashtable jsonHT = new
System.Web.Script.Serialization.JavaScriptSerializer().Deserialize<H
ashtable>(jsonResponse);

String status = "";
if (jsonHT.ContainsKey("success") )
{
    status = jsonHT["success"].ToString( );
}
else
{
    foreach ( String key in jsonHT.Keys)
    {
        String errorText = jsonHT[key].ToString( );
        status += errorText+"<br/>";
    }
}
//status is then displayed to the client
…
```

Figure 14: Consuming submitVacation from the website

```
//Connecting to the Employee WebService submitVacation method code

JSONObject jsonObj =new JSONObject(jsonResponse.toString( ));

String status = "";
if ( jsonObj.has("success"))
{
    status = jsonObj.getString("success");
}
else
{
    for ( int i = 0 ; i < jsonObj.names().length(); i++)
    {
        String errorkey = jsonObj.names( ).getString(i);
        String errorText = jsonObj.getString(errorkey);

        status += errorText +"\n";
    }
}
//status is then displayed to the client
…
```

Figure 15: Dealing with JSON String in an Android Application

Note that each application should display the status to the client and in case of an unsuccessful operation, the client should fix the errors displayed as shown in the following figure:

Figure 16: Sample Errors from submitVacation Web Method

Although performing the validation process separately in each application instead of having the validation performed in the web service method might have an advantage of saving some time on behalf of the client instead of sending and receiving the data to the web service to perform the validation until a successful input is provided, but there is also a huge advantage of performing such validation in the web service, because in this way we can easily add any further validation on the client input data, such as checking that the vacation end date is greater than or equal to the vacation start date, just once in the web method instead of having to do so in every accessing application, and in the case of mobile applications the client must also update their application so that the new effects could take place.

3. DYNAMIC FORM BUILDING

In the previous sections, data validation and manipulation have been separated from the application and have been handled by the web service, but the structure of each user form in the web page and mobile activity is hard-coded in the application, this means that if any other piece of information

need to be provided by the client such as an employee contact information during his vacation, a separate field must be added to the website form, which is an easy step to do, but the same field must be added to the mobile application and every client must update his application to have this new input field available.

To overcome such an obstacle, dynamic form building is a technique that can be used to provide maximum flexibility to an application user form, in the sense that the form used in the application to collect any information from the client is not hard-coded in the application itself, instead the form is dynamically generated upon the load of the website page or mobile activity.

A separate web service method should be prepared to generate the client form, and a standard syntax of coding should be used among the web service and every accessing application in order to generate the desired client form. A sample coding syntax is shown in the following figure with their equivalent ASP.net and Android controls:

CODE	ASP.net	Android
LBL	Label	TextView
TB	TextBox	EditText
CB	CheckBox	CheckBox
LIST	DropDownList	Spinner
...	...	...

Figure 17: Sample Code Syntax for Dynamic Form Elements

This dynamic form controls code syntax will be used to generate a response from the form generator web service method to specify each form element type and its corresponding parameter name that will be sent later to the web method responsible of client input validation and manipulation as shown in the following figure:

```
private struct ElementDataType
{
      public static String STRING = "STRING";
      public static String NUMBER = "NUMBER";
      public static String BOOLEAN = "BOOLEAN";
      public static String DATE = "DATE";
      public static String TIME = "TIME";
      … //Other Element Data Types are added here
}
private struct ElementControlType
{
      public static String LBL = "LBL";
      public static String TB = "TB";
      public static String CB = "CB";
      public static String LIST = "LIST";
      … //Other Element Control Types are added here
}
private class DynamicFormElement
{
      public String elementDatatype;
      public String elementControlType;
      public String elementParamater;
      public String elementData;
      … //Other Element information are added here
}

[WebMethod]
public String generateVacationForm ( )
{
   String res = null;
   System.Collections.Hashtable formInfo = new System.Collections.Hashtable(
);

   //VacationStartDate Element
   DynamicFormElement elem1 = new DynamicFormElement( );
   elem1.elementDatatype = ElementDataType.DATE;
   elem1.elementControlType = ElementControlType.TB;
   elem1.elementParamater = "VacationStartDate";
   elem1.elementData = DateTime.Now.ToShortDateString( );
   formInfo.Add("elem1", elem1);

   //likewise for the VacationEndDate Element

   //VacationType Element
   DynamicFormElement elem3 = new DynamicFormElement( );
   elem3.elementDatatype = ElementDataType.STRING;
   elem3.elementControlType = ElementControlType.LIST;
   elem3.elementParamater = "VacationType";
   elem3.elementData = "Annual;Sickness;Other";
   formInfo.Add("elem3", elem3);

   return new
       System.Web.Script.Serialization.JavaScriptSerializer(
).Serialize(formInfo);
}
```

Figure 18: generateVacationForm Web Service Method

The **generateVacationForm** web method once requested will return a JSON string containing the desired form elements with all of their properties returned as a nested JSON value specified by the complex user-defined object **DynamicFormElement** that every consuming application should use to build the appropriate client form in its own language and environment specific format as shown in the following figure:

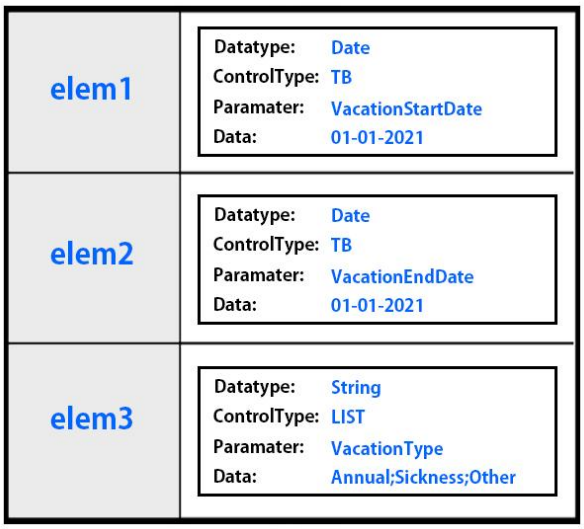

Figure 19: generateVacationForm Web Method Response

3.1 Consuming the generateVacationForm Web Method from the Website

In an ASP.net webpage, a request to the generateVacationForm web service method should be made in the **Page_Load** method in order to build the form before the page is displayed to the client. An ASP.net **Panel** control must

be prepared in the page to use it as a container to add the form elements to it dynamically at runtime.

Once the previous JSON string response in returned to the webpage, some manipulation on the response must be performed to check the type of every element, the initial element data, equivalent parameter name and any further information needed to build the appropriate form as shown in the following figure:

```
protected void Page_Load(object sender, EventArgs e)
{
//Connect to the appropriate web service method here

Hashtable jsonHT = jss.Deserialize<Hashtable>(jsonResponse);

foreach ( String key in jsonHT.Keys)
{
    String elem = key;

    Dictionary<String,Object> elemInfo =
            (Dictionary<String,Object>) jsonHT [key];

    String elementDatatype = elemInfo["elementDatatype"].ToString( );
    String elementControlType = elemInfo["elementControlType"].ToString( );
    String elementParamater = elemInfo["elementParamater"].ToString( );
    String elementData = elemInfo["elementData"].ToString( );

    if ( elementControlType.Equals("TB"))
    {
        TextBox tb = new TextBox();
        tb.ID = elementParamater;
        tb.Text = elementData;

        formPanel.Controls.Add(tb);
    }
  else if ( elementControlType.Equals("LIST"))
  {
        DropDownList list = new DropDownList();
        list.ID = elementParamater;

        String[] items = elementData.Split(new char[] { ';' });
        foreach ( String item in items)
        {
            list.Items.Add(item);
        }

        formPanel.Controls.Add(list);
  }
  else …
//continue with every element type that might be returned from the web service

 }
}
```

Figure 20: Consuming generateVacationForm Web Method from an ASP.net web page

3.2 Consuming the generateVacationForm Web Method from the Mobile Application

In the same way, a request to the generateVacationForm web service method must be done in the **onViewCreated** method of an Android mobile application in order to build the client form before the mobile activity is shown to the client and the JSON response must be manipulated in the same way but with the corresponding Android controls and using An Android **TableLayout** instead on an ASP.net Panel to add the form controls dynamically to the activity as shown in the following figure:

```
//Connecting to the Employee WebService generateVacationForm method code

JSONObject jsonObj =new JSONObject(jsonResponse.toString( ));

for ( int i = 0 ; i < jsonObj.names().length( ); i++)
{
    String elem = jsonObj.names( ).getString(i);
    JSONObject elemInfo = jsonObj.getJSONObject (elem);

    String elementDatatype = elemInfo.getString("elementDatatype");
    String elementControlType = elemInfo.getString("elementControlType");
    String elementParamater = elemInfo.getString("elementParamater");
    String elementData = elemInfo.getString("elementData");

    if ( elementControlType.equals("TB"))
    {
        EditText tb = new EditText(getContext());
        tb.setId(View.generateViewId());
        tb.setTag(elementParamater);
        tb.setText(elementData);

        TableRow tr = new TableRow(getContext());
        tr.addView(tb);

        formTable.addView(tb);
    }
    else if ( elementControlType.equals("LIST"))
    {
        List<String> list = new ArrayList<String>();
        StringTokenizer st = new StringTokenizer(elementData,";");
        while ( st.hasMoreTokens())
        {
            list.add(st.nextToken());
        }
        ArrayAdapter<String> dataAdapter = new
            ArrayAdapter<String>(getContext(),R.layout. spinner_style, list);

        Spinner spinner = new Spinner(getContext());
        spinner.setId(View.generateViewId());
        spinner.setTag(elementParamater);
        spinner.setAdapter(dataAdapter);

        TableRow tr = new TableRow(getContext());
        tr.addView(spinner);

        salTable.addView(tr);

    }
//continue with every element type that might be returned from the web service
}
```

Figure 21: Consuming generateVacationForm Web Method from an Android Mobile Application

Note that in order to submit the form data either from the website or mobile application, all the form element parameter names must be iterated and the corresponding form controls must be accessed to retrieve the input provided by the client and the element control type must be considered in order to pass the correct parameters to the previously prepared **submitVacation** web method as shown in the following figures:

```
Control control = Form.FindControl(elementParamater);
```

Figure 22: Reaching a specific control in ASP.net

```
View control = getView( ).findViewWithTag(elementParamater);
```

Figure 23: Reaching a specific control in Android

3.3 Upgrading the Dynamic Form Building Technique

An upgrade can also be made in the Dynamic Form Building Technique as to provide a single web service for generating all the needed client forms in the application, and the desired form is passed as a parameter to this general form builder method as shown in the following figure:

```
[WebMethod]
public String generalFormBuilder ( String form )
{
   String res = null;

   System.Collections.Hashtable formInfo = new System.Collections.Hashtable(
);

if ( form.Equals("Vacation") )
{
   //VacationStartDate Element
```

```
    DynamicFormElement elem1 = new DynamicFormElement( );
    elem1.elementDatatype = ElementDataType.DATE;
    elem1.elementControlType = ElementControlType.TB;
    elem1.elementParamater = "VacationStartDate";
    elem1.elementData = DateTime.Now.ToShortDateString( );
    formInfo.Add("elem1", elem1);

    ….
}
else if ( form.Equals("Contact") )
{
    …
}
else …. // All other needed forms

    return new
        System.Web.Script.Serialization.JavaScriptSerializer(
).Serialize(formInfo);
}
```

Figure 24: generalFormBuilder Web Service Method

Another upgrade can be made as well in the applications accessing the web service, is to provide a single website page or android activity for all the client forms needed in the application instead of creating a separate page or activity for each client form in the project as shown in the following figure:

```
EmployeeWebService empws = new EmployeeWebService ( );

String formName;

if ( // Vacation Form should be displayed )
    formName = "Vacation";
else if ( // Contact Form should be displayed )
    formName = "Contact";
else //any other needed forms
…
String jsonResponse = empws.generalFormBuilder (formName);
…
```

Figure 25: General Form Builder Web Page

```
SoapObject soapRequest = new
SoapObject("http://www.myBusinessDomain.com/", "generalFormBuilder");

String formName;

if ( // Vacation Form should be displayed )
    formName = "Vacation";
else if ( // Contact Form should be displayed )
    formName = "Contact";
else //any other needed forms

//Adding the corresponding parameters to the Soap request
PropertyInfo pi = new PropertyInfo();
pi.setName("form");
pi.setValue(formName);
pi.setType(String.class);
soapRequest.addProperty(pi);
…
jsonResponse = envelope.getResponse( );
…
```

Figure 26: General Form Builder Android Activity

Although these upgrades might seem harder to develop and prepare than ordinary web or mobile forms but in the long run this extra effort will enable you to greatly reduce the size of the application and to allow you full control on the form itself and how the client information is viewed, validated and manipulated.

4. SHARED WEB SERVICE ARCHITECTURE ANALYSIS

The proposed shared web service architectural techniques provide developers and programmers with great flexibility to manage multiple business applications in the best possible way. These techniques have offered a lot of benefits but at the same time have had some limitations.

4.1 Advantages of using a Shared Web Service Architecture

Some of the advantages of using a shared web service architecture are:

- After preparing the shared web service architecture for the first time, any new fixes and upgrades are made quite easily and with minimum time required.
- Mobile application will always be up-to-date with the minimum need for clients to update the application for new fixes or upgrades.
- The application will always be consistent either from the web site or the mobile app with no need to worry about forgetting to fix any bugs or make any changes in any one of them.
- Urgent new form input data are made available at the desired time without the need to worry about mobile clients not updating their applications.
- If dynamic form building techniques are implemented in a good manner, the mobile application size will be reduced greatly and would save a lot of storage space on the clients' mobile devices.
- Any new applications with different programming languages or development environments can be easily added to the business with minimum effort and time because all the functionally is already prepared by the web service.

4.2 Disadvantages of using a Shared Web Service Architecture

Here are some of the disadvantages of using a shared web service architecture:

- It is harder to be prepared by application developers and it needs more time initially to set up the needed architecture.
- More time and internet usage is needed on behalf of the client to access and retrieve the appropriate data from the web service.

5. CONCLUSION

Mobile Applications are increasingly demanded and requested by businesses and companies because of the benefits it brings to the work popularity and widespread. The developers of business applications need to support mobile applications in addition to the original business websites and services. In this paper, a shared web service architectural design is represented and discussed in details to suggest the best possible techniques of such a business architecture to provide the consistency and flexibility among the different business applications and reduce as much obstacles that come with maintaining such an architecture. New and improved ideas and techniques need to be researched and invented to provide better architectural designs to try to minimize the few disadvantages that still exist and to fulfill the growing changes and demands in the world of business application developments and programming technologies.

6. REFERENCES

[1] Md. Rashedul Islam, Md. Rofiqul Islam, Tohidul Arafhin Mazumder, "Mobile Application and Its Global Impact", International Journal of Engineering & Technology, 2010.

[2] "Mobile and tablet internet usage exceeds desktop for

first time worldwide", StatCounter Global Stats. Retrieved 27 July 2020.

[3] Newcomer E., Understanding Web Services –XML WSDL SOAP, 1st ed. Addison Wesley Professional. Boston, United Stated of America. (2002).

[4] Wang Hongbing, Huang Joshua, Qu Yuzhong, Xie Junyuan, Web services: Problems and Future Directions 2004, Journal of Web Semantics, NSFC, JSNSF.

[5] PENG Dunlu, CAO Lidong, XU Wenji, Using JSON for Data Exchanging in Web Service Applications, 2011, Journal of Computational Information Systems 7, JOFCIS.

[6] Evjen B., Hanselman S., Rader D., Professional ASP.NET 4 in C# and VB, 1st ed. Wiley Publishing, Inc. Indiana, United Stated of America. (2010).

[7] Ugurlu T., Pro ASP.NET Web API: HTTP Web Services in ASP.NET (Expert's Voice in .NET), 1st ed. Apress. New York, United Stated of America. (2013).

[8] Nagel C., Professional C# 7 and .NET Core 2.0, 1st ed. Wiley Publishing, Inc. Indiana, United Stated of America. (2018).

[9] Meier R., Professional Android Application Development, 1st ed. Wiley Publishing, Inc. Indiana, United Stated of America. (2009).

[10] "kSOAP2" Kilobyte Objects, [Online]. Available: http://kobjects.org/. [Accessed 03 July 2020].