



# SOFTWARE DESIGN SPECIFICATION

8/23/2013

Graduate Capstone



## Table of Contents

|  |    |
|--|----|
| 1 Introduction .....                   | 4  |
| 1.1 Intended Audience .....            | 4  |
| 1.2 References .....                   | 4  |
| 1.3 Revision History .....             | 4  |
| 2 System Overview .....                | 5  |
| 2.1 Section Overview .....             | 5  |
| 2.2 General Constraints .....          | 5  |
| 2.3 Data Design.....                   | 5  |
| 2.4 Actors.....                        | 5  |
| 2.4.1 Phone .....                      | 5  |
| 2.4.2 Admin Website.....               | 5  |
| 2.4.3 Database .....                   | 6  |
| 2.4.4 Web Services .....               | 6  |
| 2.4.5 Survey Website.....              | 6  |
| 2.5 Domain Model.....                  | 7  |
| 2.6 Goals and Guidelines .....         | 7  |
| 3 Architecture Design.....             | 8  |
| 3.1 Section Overview .....             | 8  |
| 3.2 Architectural Strategies .....     | 8  |
| 3.3 System Architecture.....           | 9  |
| 3.3.1 Web Services .....               | 9  |
| 3.3.2 Automation Process.....          | 14 |
| 3.3.3 Admin CMS .....                  | 14 |
| 3.3.4 Phone Application .....          | 15 |
| 3.3.5 Reverse Proxy .....              | 22 |
| 3.4 Detailed System Design .....       | 23 |
| 4 User Interface Design .....          | 24 |
| 4.1 Section Overview .....             | 24 |
| 4.2 GUI Components .....               | 24 |
| 4.3 Prototype .....                    | 24 |
| 5 Design decisions and tradeoffs ..... | 25 |
| 5.1 Web Services.....                  | 25 |
| 5.1.1 SOAP .....                       | 25 |
| 5.1.2 RESTful.....                     | 26 |





|  |    |
|--|----|
| 5.1.3 Decision .....                             | 26 |
| 5.2 Mobile Cross-Platform Development .....      | 27 |
| 5.2.1 Hybrid App.....                            | 27 |
| 5.2.2 Xamarin .....                              | 28 |
| 5.2.3 Choice .....                               | 28 |
| 5.3 Mobile Cross-Platform Design Pattern .....   | 29 |
| 5.3.1 Model View View-Model (MVVM).....          | 29 |
| 5.3.2 Portable Class Library (PCL).....          | 29 |
| 5.4 Caching.....                                 | 30 |
| 5.4.1 Reverse Proxy .....                        | 30 |
| 5.4.2 Phone Caching.....                         | 30 |
| 5.5 Database Access .....                        | 31 |
| 5.1 Data Access Layer through Web Services ..... | 31 |
| 6 Pseudocode for System .....                    | 32 |
| 6.1 Admin CMS .....                              | 32 |
| 6.1.1 Element Manager Page (Back-End) .....      | 32 |
| 6.1.2 Element Implementation.....                | 34 |
| 6.2 Web Services.....                            | 37 |
| 6.2.1 Service.....                               | 37 |
| 6.2.2 Implementation .....                       | 38 |



# 1 Introduction

The purpose of this document is to provide description regarding the architecture and other design aspects of the system. The document will be used by the development team to develop the functionality. Clients can also use this to verify the agreed upon functionality.

## 1.1 Intended Audience

The document is intended for individuals of a medium-high technical or business background.

## 1.2 References

- [http://web.cs.dal.ca/~hawkey/3130/SDS\\_outline.doc](http://web.cs.dal.ca/~hawkey/3130/SDS_outline.doc)
- [http://www.cs.utah.edu/~jamesj/ayb2005/docs/SDS\\_v2.htm](http://www.cs.utah.edu/~jamesj/ayb2005/docs/SDS_v2.htm)
- [http://www.se.rit.edu/~vdkrit/design/VDK-RIT\\_SDS.doc](http://www.se.rit.edu/~vdkrit/design/VDK-RIT_SDS.doc)

## 1.3 Revision History

| Name        | Date       | Reason For Change   | Version |
|-------------|------------|---|---------|
| Andy Bottom | 04/28/2013 | Started compiling all architecture diagrams into this document          | 0.1     |
| Andy Bottom | 05/07/2013 | Completed the finishing touches on this document                        | 1.0     |
| Andy Bottom | 05/21/2013 | Updated and added additional design decision including PCLs and caching | 1.1     |



## 2 System Overview

### 2.1 Section Overview

---

This section is used for explanations on generically high-level overview of the entire system. This includes information regarding arts of this system.

### 2.2 General Constraints

---

The [Software Requirement Specification](#) for information about general constraints of the system.

### 2.3 Data Design

---

Please refer to the [Database Design Document](#) for a further analysis of the Data and Database Design.

### 2.4 Actors

---

The actors are the main endpoints that utilize the functionality of the system.

#### 2.4.1 Phone

---

The phones are the phone applications which the Users interact with to use the systems features.

#### 2.4.2 Admin Website

---

The admin website is what the administrators interact with to manage the data in the system.





### 2.4.3 Database

---

The database holds the information for the system and gets used by the other actors. All interactions with the database happen via the Web Services.

### 2.4.4 Web Services

---

The Web Services is the way that both the Phone and Admin Website performs common logic and obtains the data. The Web Services acts as our data access layer for both the client applications to be able to get data from the database. The Web Services also acts as the kickoff for the Automation Web Driver process.

### 2.4.5 Survey Website

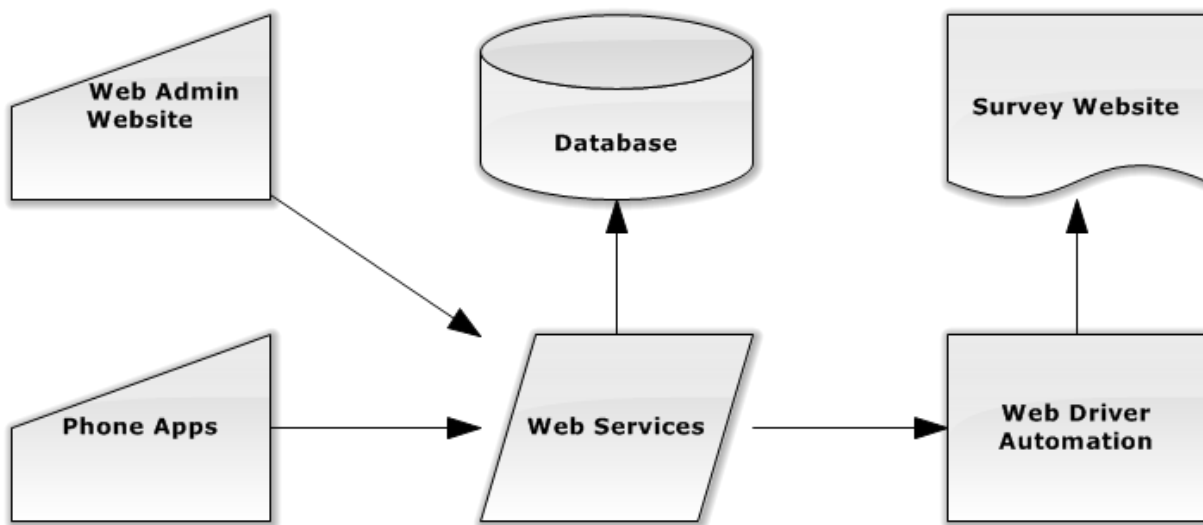
---

The Survey website is a pre-existing website that the company uses for the Satisfaction Surveys. Typically these websites are created by a company that offers to create these types of surveys and the origin company uses their services for this reason. It is important to note that the Company Survey Website is completely independent from our system and code base. The only interaction we have with the website is that it is the endpoint that is our Web Driver uses to perform the form automation on.



## 2.5 Domain Model

---



## 2.6 Goals and Guidelines

---

- A large emphasis is put on convenience of the entire project. The goal is to provide an easier and more convenient way to take the surveys.
- Convenience will be done via very good Usability of the Phone Application.



## 3 Architecture Design

### 3.1 Section Overview

---

The following section explains the architectural designs of the major system in the project.

### 3.2 Architectural Strategies

---

- Object Oriented Design is always the forefront of any application that I develop for. The idea of being able to get the most flexibility and reuse out of my code is always a goal that I strive for.
- The use of a DAL is implemented with the web services so that I can keep separation between the user devices and the rest of the system. It also provides reuse of all data calls.
- The automated process is also located inside the web services layer to keep abstraction between the devices and the process.
- The Web Admin site is a very object oriented approach and follows very closely along the lines of the database. The Web Admin is a Content Management System for the project.
- In the phone applications, we will create a PCL that will be referenced by the apps to incorporate a large reuse of the backend page logic.





## 3.3 System Architecture

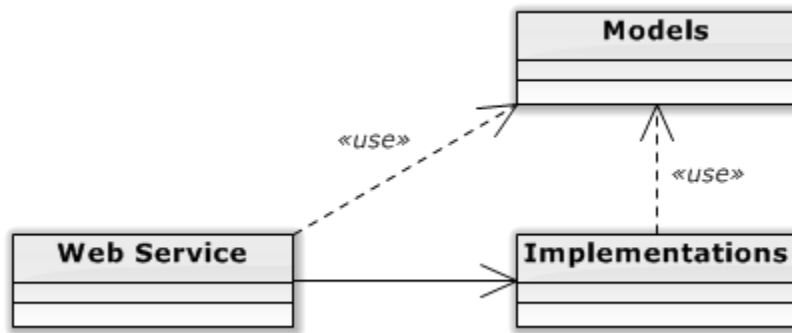
---

### 3.3.1 Web Services

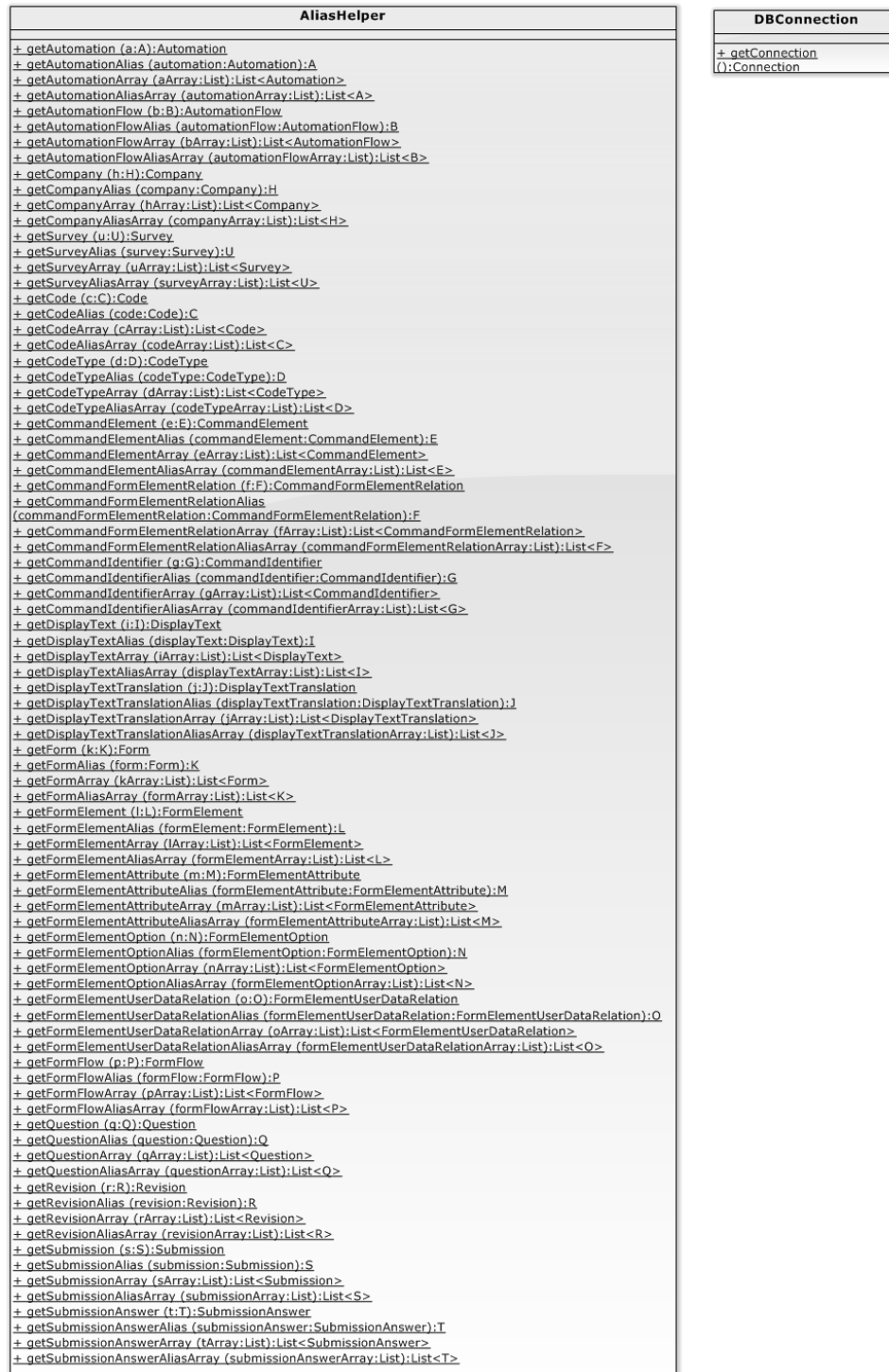
---

#### 3.3.1.1 High-Level Class Diagram

---

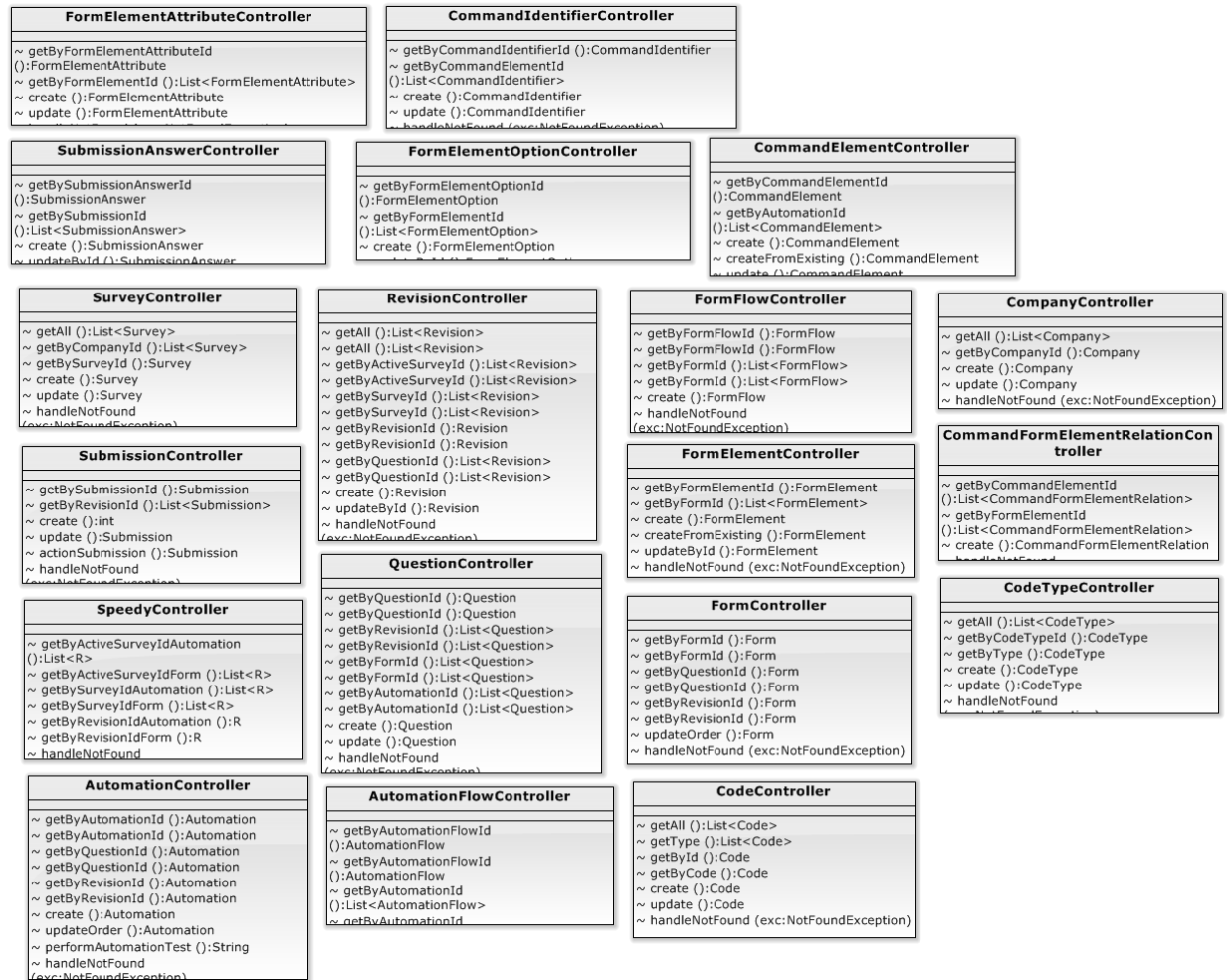


### 3.3.1.2 Class Diagram



*Class Diagram of the Common Package*





*Class Diagram of the Models Package*





```

- f1 : int
- f2 : int
- f3 : int

+ F ()
+ F (f1:int, f2:int, f3:int)
+ getF1 () :int
+ setF1 (f1:int)
+ getF2 () :int
+ setF2 (f2:int)
+ getF3 () :int
+ setF3 (f3:int)
+ toString () :String

```

```

- u1 : int
- u2 : int
- u3 : String
- u4 : List<R>

+ U ()
+ U (u1:int, u2:int, u3:String, u4:List<R>)
+ getU1 () :int
+ setU1 (u1:int)
+ getU2 () :int
+ setU2 (u2:int)
+ getU3 () :String
+ setU3 (u3:String)
+ getU4 () :List<R>
+ setU4 (u4:List)
+ toString () :String

```

```

- t1 : int
- t2 : int
- t3 : int
- t4 : int
- t5 : String

+ T ()
+ T (t1:int, t2:int, t3:int, t4:int, t5:String)
+ getT1 () :int
+ setT1 (t1:int)
+ getT2 () :int
+ setT2 (t2:int)
+ getT3 () :int
+ setT3 (t3:int)
+ getT4 () :int
+ setT4 (t4:int)
+ getT5 () :String
+ setT5 (t5:String)
+ toString () :String

```

```

- s1 : int
- s2 : int
- s3 : Date
- s4 : Date
- s5 : List<T>

+ S ()
+ S (s1:int, s2:int, s3:Date, s4:Date, s5:List<T>)
+ getS1 () :int
+ setS1 (s1:int)
+ getS2 () :int
+ setS2 (s2:int)
+ getS3 () :Date
+ setS3 (s3:Date)
+ getS4 () :Date
+ setS4 (s4:Date)
+ getS5 () :List<T>
+ setS5 (s5:List)
+ toString () :String

```

```

- r1 : int
- r2 : int
- r3 : boolean
- r4 : Date
- r5 : String
- r6 : Q

+ R ()
+ R (r1:int, r2:int, r3:boolean, r4:Date, r5:String, r6:Q)
+ getR1 () :int
+ setR1 (r1:int)
+ getR2 () :int
+ setR2 (r2:int)
+ isR3 () :boolean
+ setR3 (r3:boolean)
+ getR4 () :Date
+ setR4 (r4:Date)
+ getR5 () :String
+ setR5 (r5:String)
+ getR6 () :Q
+ setR6 (r6:Q)
+ toString () :String

```

```

- e1 : int
- e2 : C
- e3 : List<G>
- e4 : List<F>
- e5 : String

+ E ()
+ E (e1:int, e2:C, e3:List, e4:List, e5:String) :<G>
+ getE1 () :int
+ setE1 (e1:int)
+ getE2 () :C
+ setE2 (e2:C)
+ getE3 () :List<G>
+ setE3 (e3:List)
+ getE4 () :List<F>
+ setE4 (e4:List)
+ getE5 () :String
+ setE5 (e5:String)
+ toString () :String

```

```

- j1 : int
- j2 : int
- j3 : C
- j4 : String

+ J ()
+ J (j1:int, j2:int, j3:C, j4:String)
+ getJ1 () :int
+ setJ1 (j1:int)
+ getJ2 () :int
+ setJ2 (j2:int)
+ getJ3 () :C
+ setJ3 (j3:C)
+ getJ4 () :String
+ setJ4 (j4:String)
+ toString () :String

```

```

- n1 : int
- n2 : int
- n3 : I
- n4 : String

+ N ()
+ N (n1:int, n2:int, n3:I, n4:String)
+ getN1 () :int
+ setN1 (n1:int)
+ getN2 () :int
+ setN2 (n2:int)
+ getN3 () :I
+ setN3 (n3:I)
+ getN4 () :String
+ setN4 (n4:String)
+ toString () :String

```

```

- o1 : int
- o2 : int
- o3 : int
- o4 : String

+ O ()
+ O (o1:int, o2:int, o3:int, o4:String)
+ getO1 () :int
+ setO1 (o1:int)
+ getO2 () :int
+ setO2 (o2:int)
+ getO3 () :int
+ setO3 (o3:int)
+ toString () :String

```

```

- p1 : int
- p2 : int
- p3 : Q
- p4 : String

+ P ()
+ P (p1:int, p2:int, p3:Q, p4:String)
+ getP1 () :int
+ setP1 (p1:int)
+ getP2 () :int
+ setP2 (p2:int)
+ getP3 () :Q
+ setP3 (p3:Q)
+ getP4 () :String
+ setP4 (p4:String)
+ toString () :String

```

```

- q1 : int
- q2 : int
- q3 : String
- q4 : K
- q5 : A

+ Q ()
+ Q (q1:int, q2:int, q3:String, q4:K, q5:A)
+ getQ1 () :int
+ setQ1 (q1:int)
+ getQ2 () :int
+ setQ2 (q2:int)
+ getQ3 () :String
+ setQ3 (q3:String)
+ getQ4 () :K
+ setQ4 (q4:K)
+ getQ5 () :A
+ setQ5 (q5:A)
+ toString () :String

```

```

- d1 : int
- d2 : String
- d3 : String

+ D ()
+ D (d1:int, d2:String, d3:String)
+ getD1 () :int
+ setD1 (d1:int)
+ getD2 () :String
+ setD2 (d2:String)
+ getD3 () :String
+ setD3 (d3:String)
+ toString () :String

```

```

- g1 : int
- g2 : int
- g3 : C
- g4 : String

+ G ()
+ G (g1:int)
+ G (g1:int, g2:int, g3:C, g4:String)
+ getG1 () :int
+ setG1 (g1:int)
+ getG2 () :int
+ setG2 (g2:int)
+ getG3 () :C
+ setG3 (g3:C)
+ getG4 () :String
+ setG4 (g4:String)
+ toString () :String

```

```

- h1 : int
- h2 : String
- h3 : List<U>

+ H ()
+ H (h1:int, h2:String, h3:List<U>)
+ getH1 () :int
+ setH1 (h1:int)
+ getH2 () :String
+ setH2 (h2:String)
+ getH3 () :List<U>
+ setH3 (h3:List)
+ toString () :String

```

```

- i1 : int
- i2 : String
- i3 : List<J>

+ I ()
+ I (i1:int, i2:String, i3:List<J>)
+ getI1 () :int
+ setI1 (i1:int)
+ getI2 () :String
+ setI2 (i2:String)
+ getI3 () :List<J>
+ setI3 (i3:List)
+ toString () :String

```

```

- l1 : int
- l2 : C
- l3 : List<N>
- l4 : List<M>

+ L ()
+ L (l1:int, l2:C, l3:List, l4:List) :<N>
+ getL1 () :int
+ setL1 (l1:int)
+ getL2 () :C
+ setL2 (l2:C)
+ getL3 () :List<N>
+ setL3 (l3:List)
+ getL4 () :List<M>
+ setL4 (l4:List)
+ toString () :String

```

```

- m1 : int
- m2 : int
- m3 : C
- m4 : String

+ M ()
+ M (m1:int, m2:int, m3:C, m4:String)
+ getM1 () :int
+ setM1 (m1:int)
+ getM2 () :int
+ setM2 (m2:int)
+ getM3 () :C
+ setM3 (m3:C)
+ getM4 () :String
+ setM4 (m4:String)
+ toString () :String

```

```

- c1 : int
- c2 : String
- c3 : String
- c4 : String
- c5 : D

+ C ()
+ C (c1:int, c2:String, c3:String, c4:String, c5:D)
+ getC1 () :int
+ setC1 (c1:int)
+ getC2 () :String
+ setC2 (c2:String)
+ getC3 () :String
+ setC3 (c3:String)
+ getC4 () :String
+ setC4 (c4:String)
+ getC5 () :D
+ setC5 (c5:D)
+ toString () :String

```

```

- b1 : int
- b2 : int
- b3 : Q
- b4 : int

+ B ()
+ B (b1:int, b2:int, b3:Q, b4:int)
+ getB1 () :int
+ setB1 (b1:int)
+ getB2 () :int
+ setB2 (b2:int)
+ getB3 () :Q
+ setB3 (b3:Q)
+ getB4 () :int
+ setB4 (b4:int)
+ toString () :String

```

```

- a1 : int
- a2 : List<E>
- a3 : List<B>

+ A ()
+ A (a1:int, a2:List, a3:List) :<E>
+ getA1 () :int
+ setA1 (a1:int)
+ getA2 () :List<E>
+ setA2 (a2:List)
+ getA3 () :List<B>
+ setA3 (a3:List)
+ toString () :String

```

```

- k1 : int
- k2 : List<L>
- k3 : List<P>

+ K ()
+ K (k1:int, k2:List, k3:List) :<L>
+ getK1 () :int
+ setK1 (k1:int)
+ getK2 () :List<L>
+ setK2 (k2:List)
+ getK3 () :List<P>
+ setK3 (k3:List)
+ toString () :String

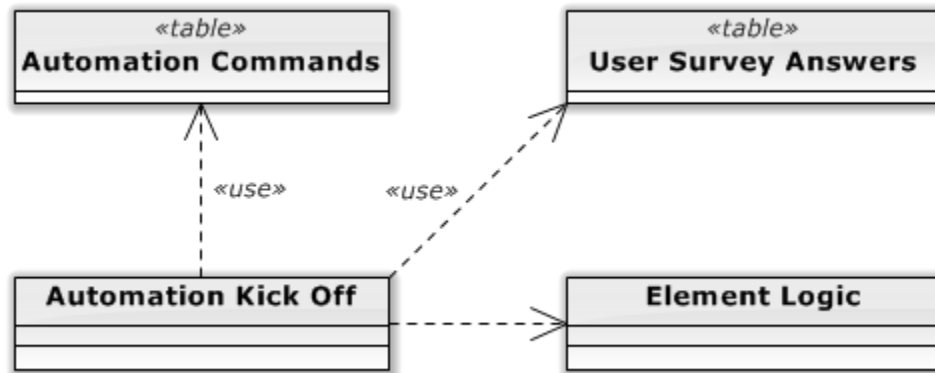
```

## Class Diagram of the Alias Package



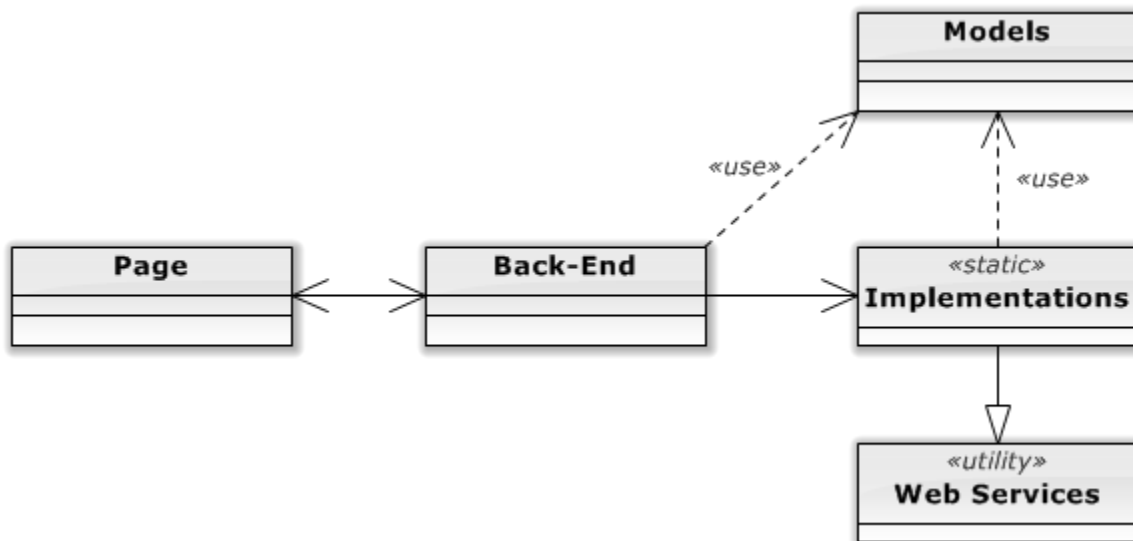
### 3.3.2 Automation Process

#### 3.3.2.1 High-Level Class Diagram



### 3.3.3 Admin CMS

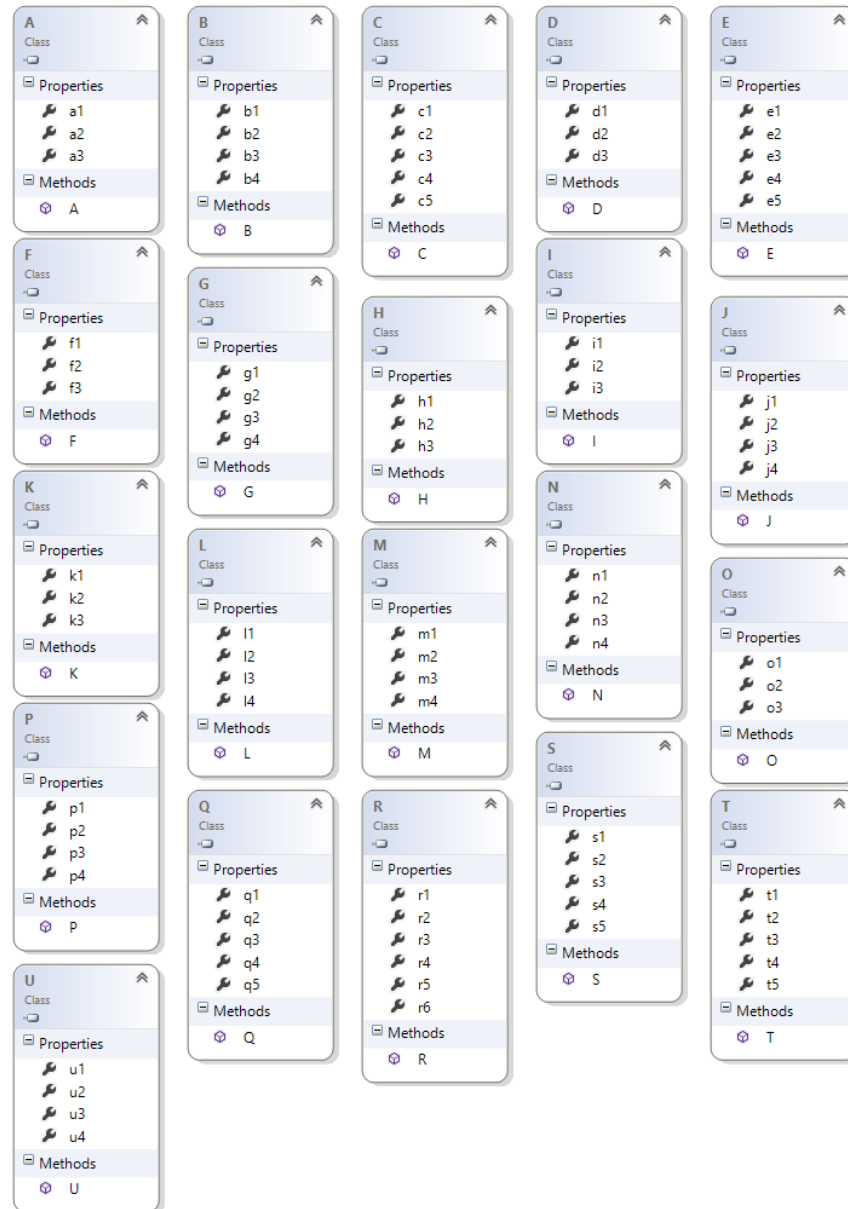
#### 3.3.3.1 High-Level Diagram



### 3.3.4 Phone Application

#### 3.3.4.1 Portable Class Library (Class Diagram)

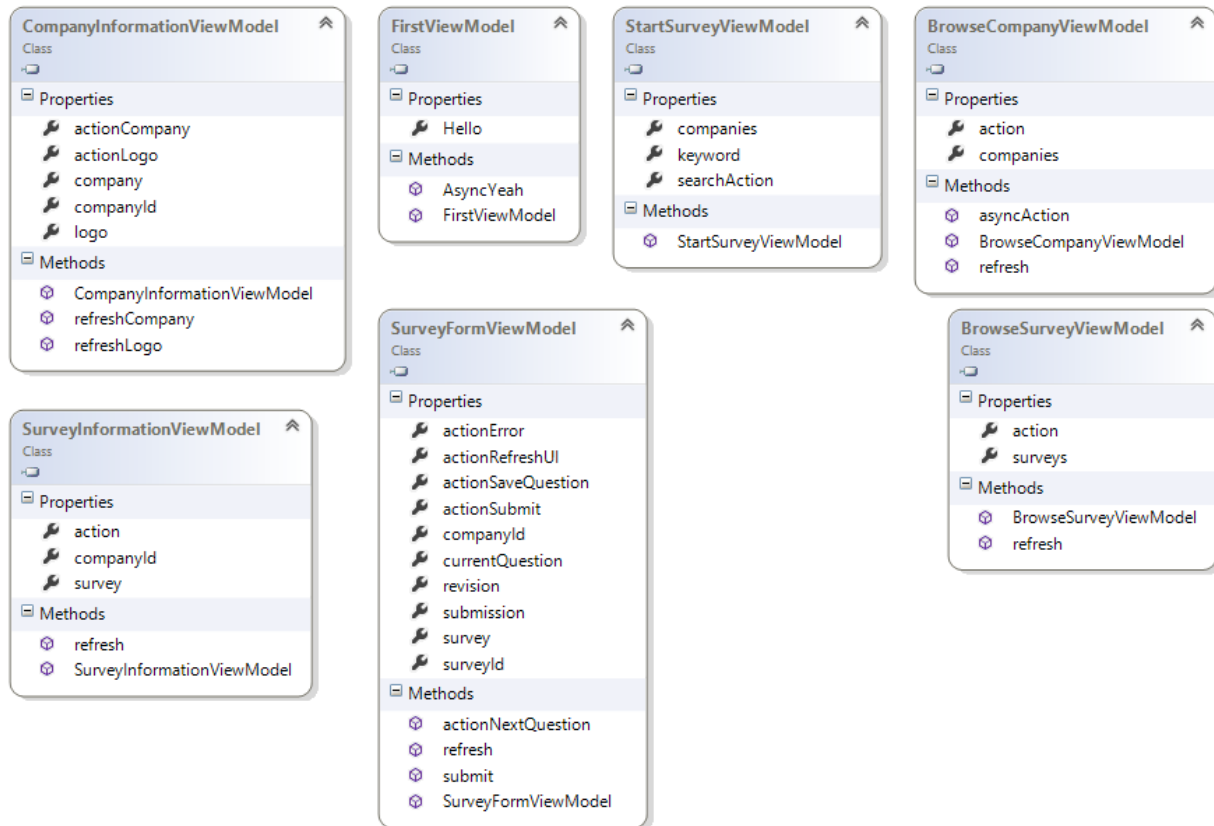
The following diagrams are the class diagrams of the classes located in the Portable Class Library.



*Class Diagram of the Alias Package*



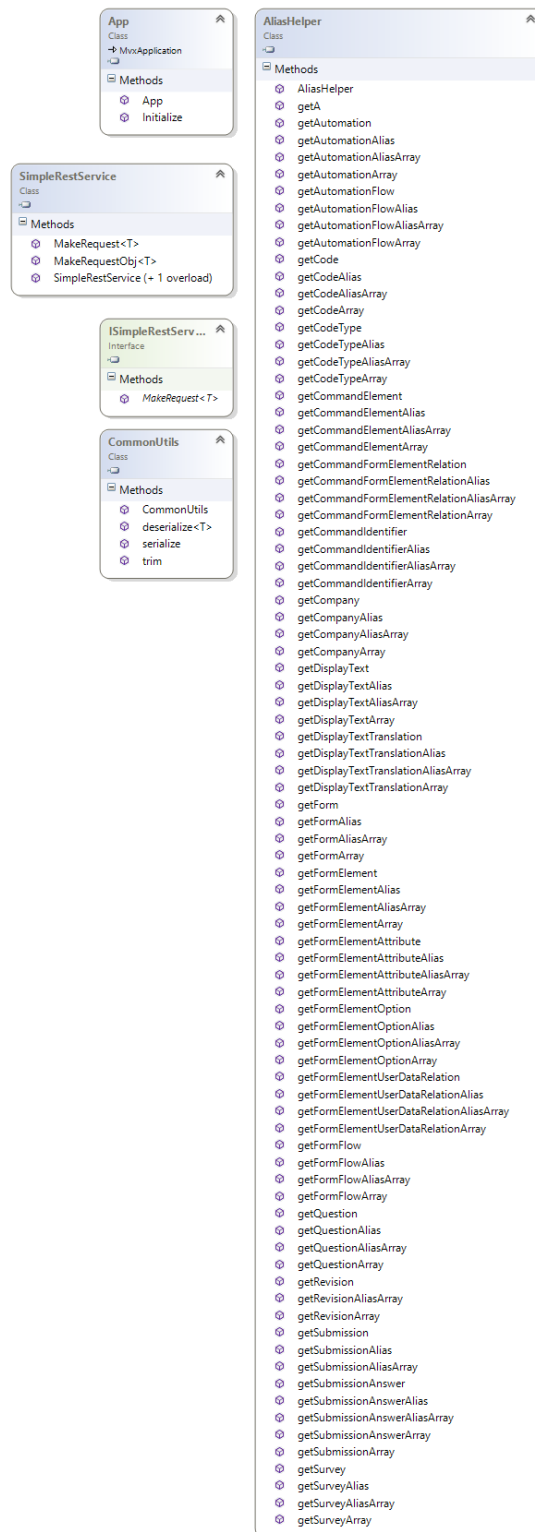




Class Diagram of the View Model Package

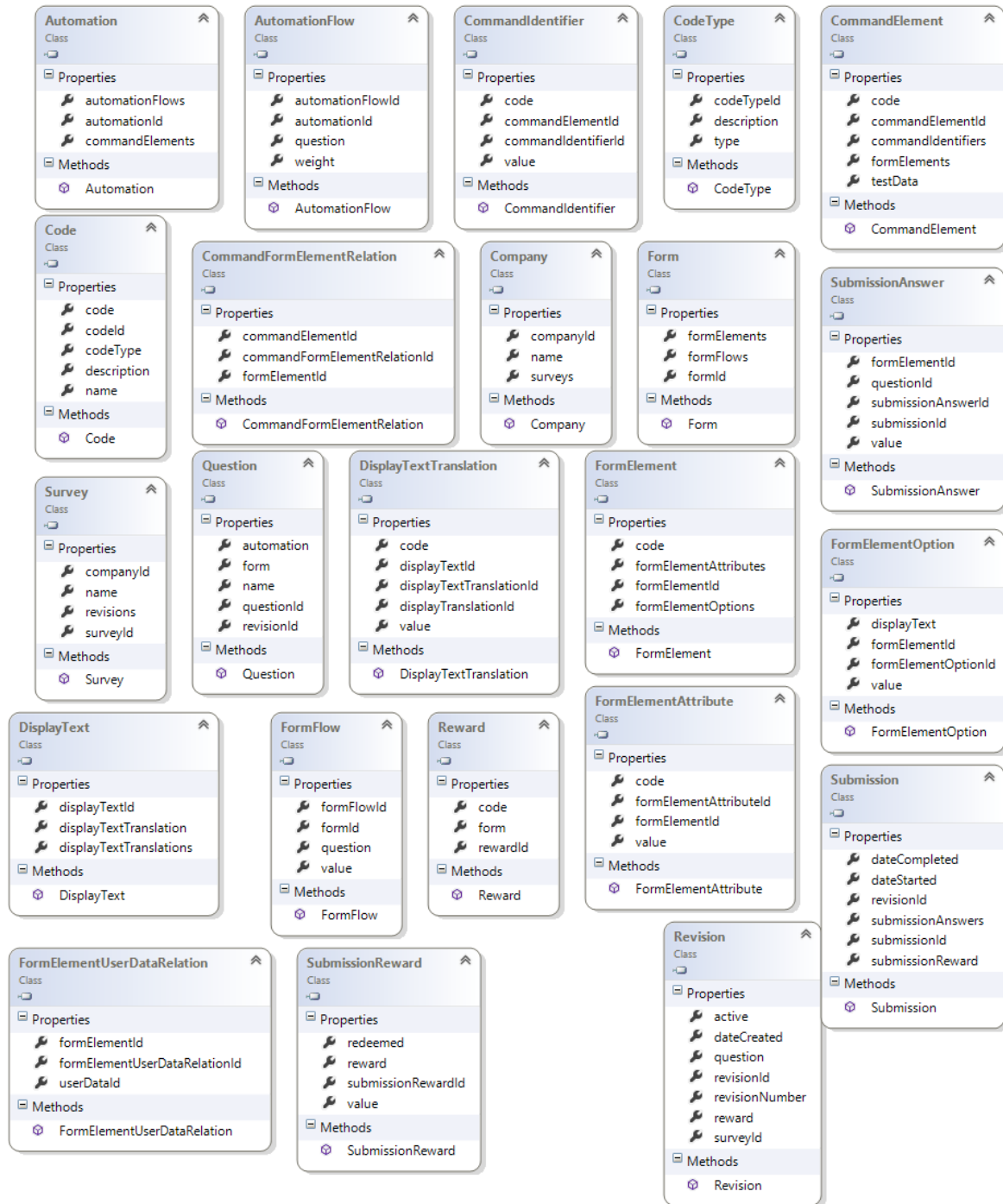






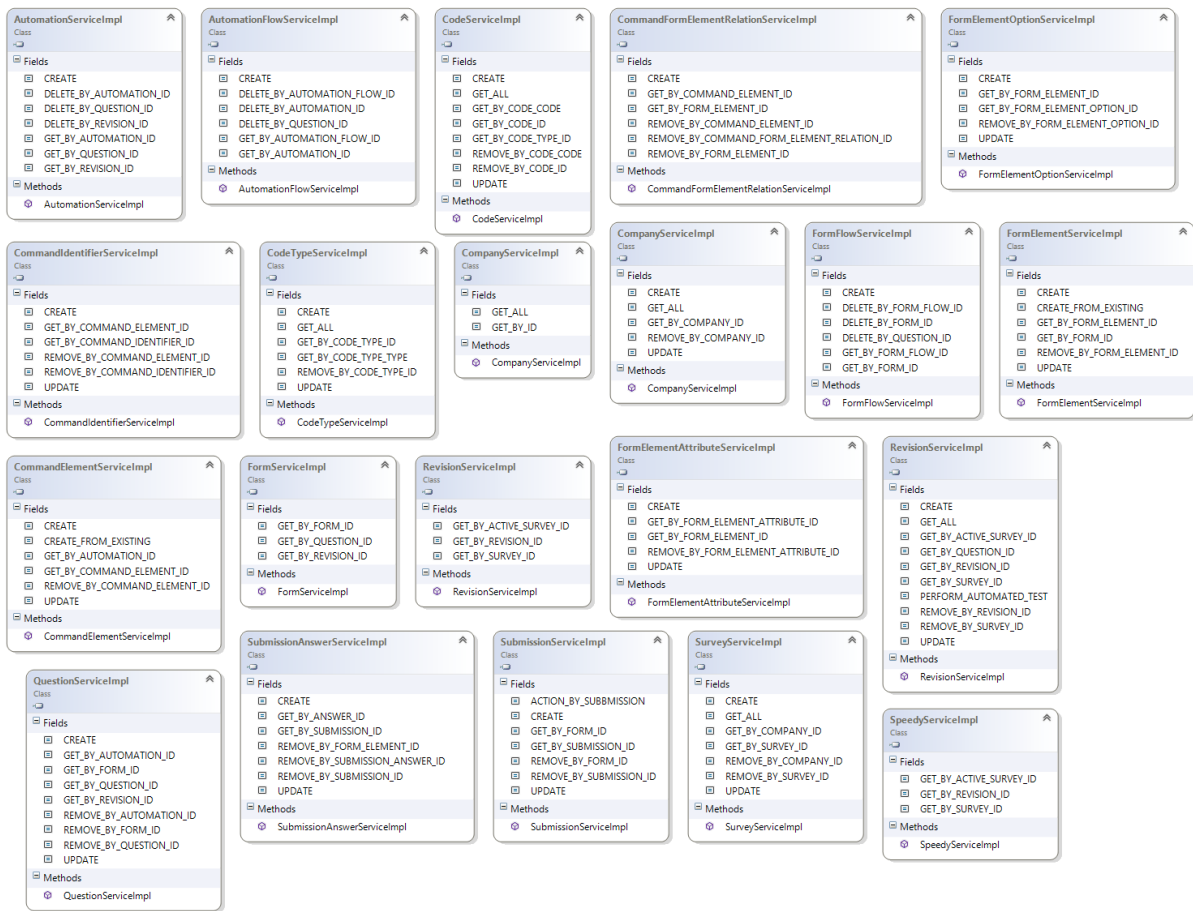
*Class Diagram of the Common Package*





*Class Diagram of the Model Package*

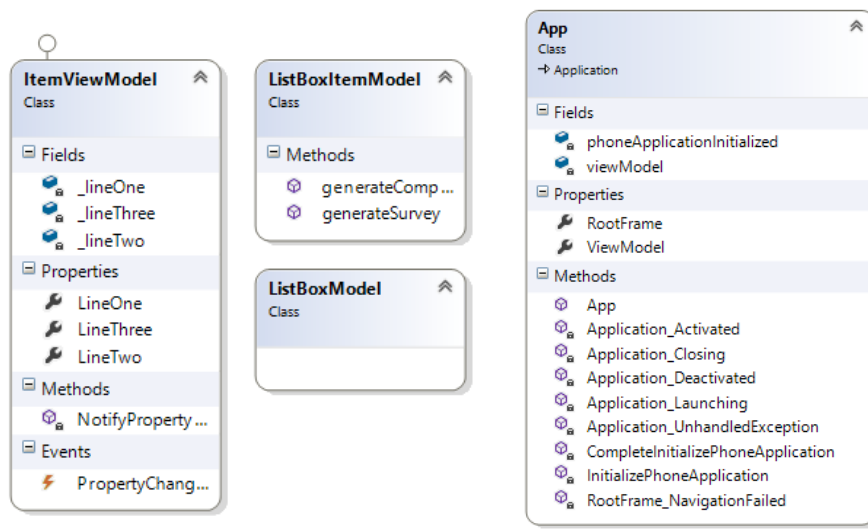




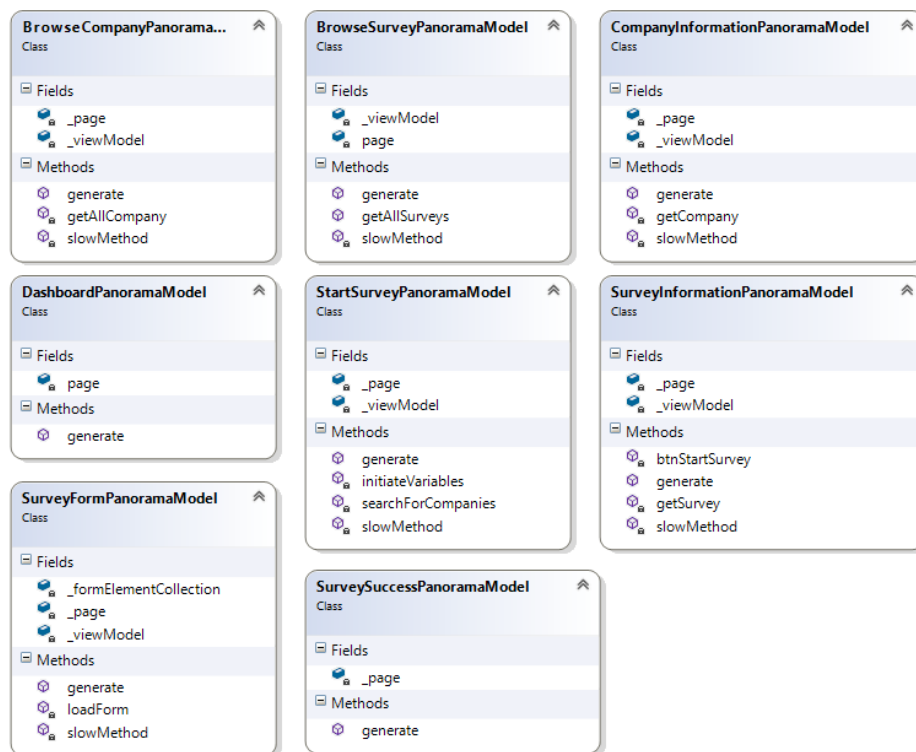
*Class Diagram of the Service Access Layer Package*



### 3.3.4.2 Windows Phone Application

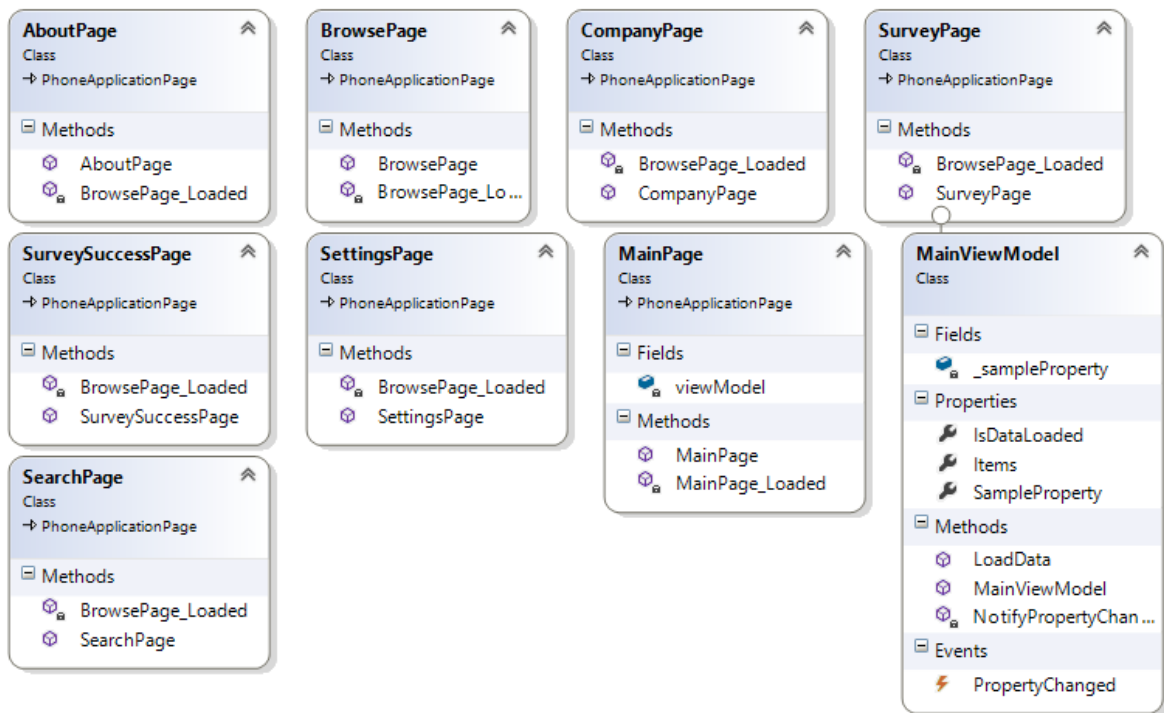


*Class Diagram of the Miscellaneous Package*



*Class Diagram of the Panorama Model Package*



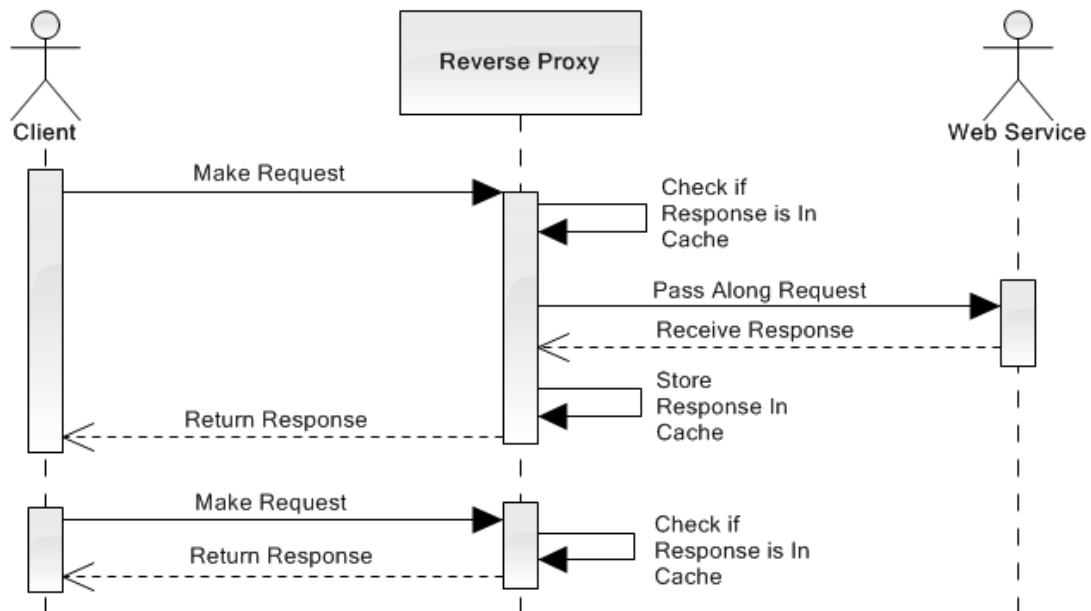


*Class Diagram of the Page Package*

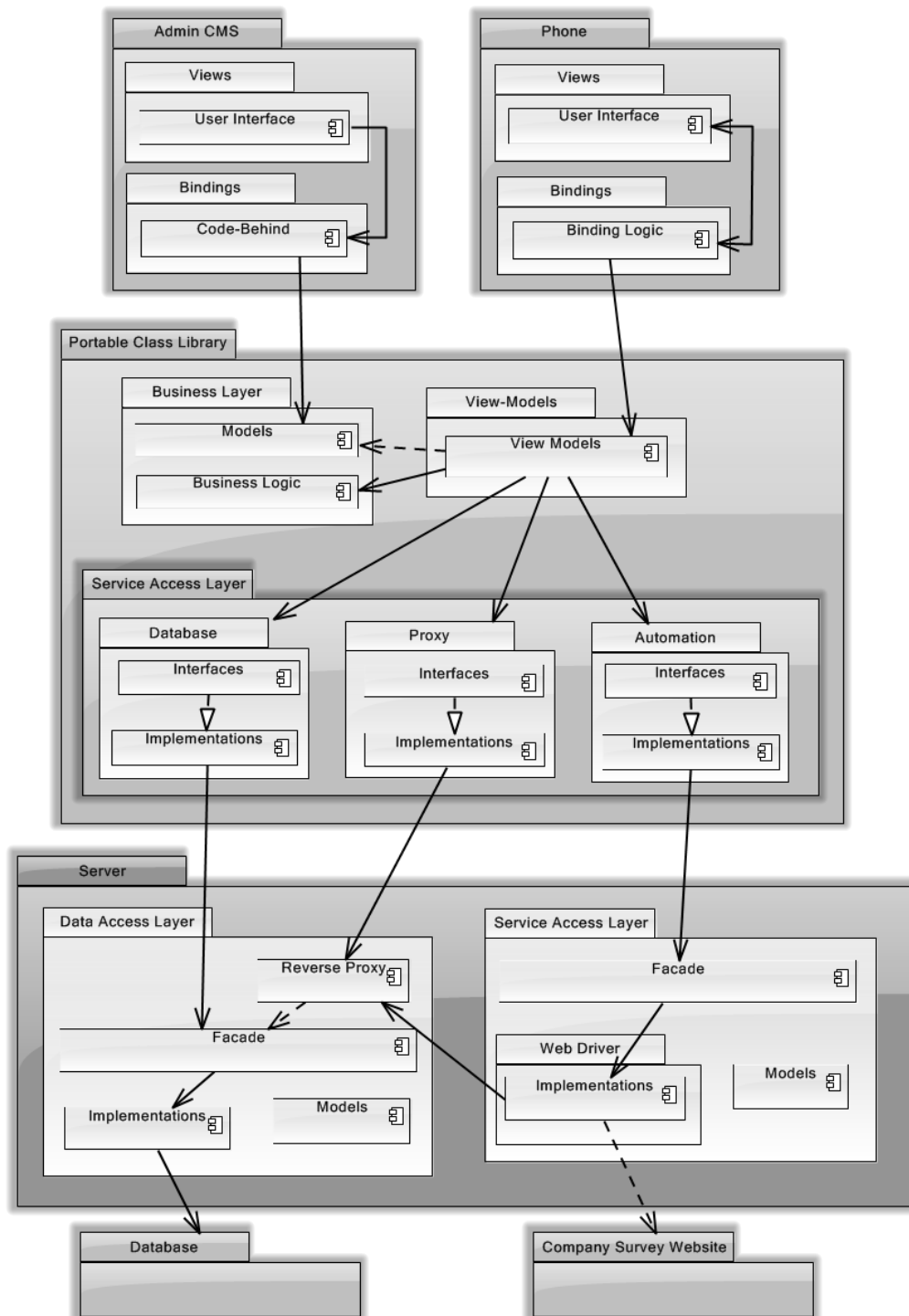


### 3.3.5 Reverse Proxy

#### 3.3.5.1 Sequence Diagrams



### 3.4 Detailed System Design





## 4 User Interface Design

### 4.1 Section Overview

---

This section is used as a guide for designing the user interface for both the Admin CMS and the Phone Application. For more detailed information about the User Interface, refer to the [User Experience Guide](#).

### 4.2 GUI Components

---

Refer to the [User Experience Guide](#) about the components of the user interface.

### 4.3 Prototype

---

Please refer to the [User Experience Guide](#) for the wireframe of the phone application.







## 5 Design decisions and tradeoffs

In this section, in-depth explanations containing reasoning for choices made in regard to the implementation and design of the system are explained. The options, pros and cons and overall choice and reason will be explained.

### 5.1 Web Services

---

Since web services are a vital part of our application, it was needed to be decided as to which time of web services would be implemented in the system. The choices were SOAP and RESTful. Below are a brief description of each and the choice made.

#### 5.1.1 SOAP

---

Soap was the first type of web service created. I also have had experience with SOAP through an internship, so I already know the concept and how to implement them. The main part of SOAP is that it uses XML formatting and is relatively stricter in handling data and formatting. SOAP also tends to be more secure and is a better choice when using very highly sensitive data. Unfortunately, the data in our system will not be immensely sensitive.





### 5.1.2 RESTful

---

After researching more about Web Services, I came upon RESTful. It seemed to me that currently, that RESTful web services seemed to be the more trendy design choice for Web Services. I thought it would be a good idea to gain experience and learn how to implement them. The conceptual flow of RESTful is extremely similar to the SOAP Web Services. The advantages for RESTful are that the data can be sent in as XML or as JSON. Also, the formatting of the system of RESTful tends to be simpler and more flexible. This causes the downside of it being less secure and not a good choice for handling highly sensitive data. Fortunately there are still other ways several designs to help make RESTful web services.. Also, one last note, which was more of a deciding factor, is that the ability to handle the data as JSON would be advantageous for mobile development because JSON is less labor intensive in compiling and parsing than the XML.

### 5.1.3 Decision

---

After weighing the pros and cons, it was decided that RESTful Web Services would the best fit for the less labor intensive fashion of dealing with JSON and that RESTful seems to be the go to choice in the future.





## 5.2 Mobile Cross-Platform Development

---

There was a need to develop the phone application to be cross-platform. When proposed with this problem there are relatively two directions you can take. You can implement a Web Wrapper Application or use a framework to develop the application, such as Xamarin. Below I discuss the advantages of each and the choice made.

### 5.2.1 Hybrid App

---

A hybrid app is essentially a fake app. By this I mean is that it is an application that is written with HTML, CSS and JavaScript. Essentially it is a mobile website that is then “wrapped” inside a phone application so it can be “installed” on a phone. But it isn’t interpretive like a native app, but instead uses the browser rendering technology to display. The advantage of this is that Hybrid Apps are highly multi-platform friendly as it uses Web Standards and not the phone to render. The disadvantage of this is that it doesn’t utilize the true functionality of a phone to the extent that native apps can. Also, another major disadvantage is that the UI of the app doesn’t look native. To explain further, each smartphone OS has its own unique look. Apple has the glossy rounded look where as Windows Phone has its Metro look. A hybrid app that is developed for both of the platforms will function properly, but the User Experience of the app is very different than what the user is used to. Due to the nature of how “Web Apps” look and feel, it is evident that the application was not created for that OS. Thus, they basically don’t fit in.





### 5.2.2 Xamarin

---

The other option is to create a Native Application. Unfortunately, all the OS use a different programming language to develop a application. This realization makes it impossible to create an application that is cross-platform. There would be no reuse of code and the developers would be stuck maintaining 3 separate code bases. This is why a use of a third-party framework can be used to create an application for all three OS but in the same language. The standout option that I selected was Xamarin.

Xamarin is a Framework that allows developers to create a .Net Native Application. The framework will then compile the application to a Native App of all the OS. The advantage of having the native app is that you can utilize a very large potential of the phone's functionality and also, the applications will be displayed using Native UI elements of the OS, thus the applications will feel very natural to the user. By using Xamarin, the UI and the Back-end Logic can be separated. By implementing in this fashion, the only customized code that needs to be developed in the UI for each operating system. The Logic on the backend will can all be reused between all OS. The disadvantage is that you will need to purchase a license to utilize Xamarin.

### 5.2.3 Choice

---

After weighing all the options, I decided to go with Xamarin to develop cross-platform Native Applications. The ability to have a native experience for the user is important and the ability to achieve this with a high percentage of code-reuse is undeniably beneficial.



## 5.3 Mobile Cross-Platform Design Pattern

---

### 5.3.1 Model View View-Model (MVVM)

---

I initially was going to develop the apps following the MVVM pattern. To implement it this way, I found a framework which is very similar to Xamarin called MVVM Cross (MVX) which allows the user to use the MVX libraries and have a core library which could very very easily have large code reuse for all mobile operating systems. At the beginning of my phone development process, I looked very strongly into the possibility of using MVX as my framework and it looked like it would be possible. However, tricky aspect of my app would be the need to have a dynamic form displayed for the surveys. I looked at it at all angles and determined that where MVX is currently with their version 3 release, it does not have the correct capabilities yet to successfully implement all aspects of my application, thus the design decision to continue with the Xamarin Framework was made. However, I have immensely high hopes and expectations that MVX will be a great choice for app development in the future.

### 5.3.2 Portable Class Library (PCL)

---

A portable class library is essentially a package of classes that will be used by all OS versions of the project and will contain all the unified functionality of the system. Unfortunately, at this moment, PCLs aren't supported by Xamarin directly, (but will be in the near future), so instead, I will be making the PCL package, and just copy the package into each project. Inside of the PCL, will be business logic, data access object for accessing the database (phone, and the remote) and a service access layer (for automation and remote database.) Models will also be defined in the PCL.





## 5.4 Caching

---

Since there will be a very heavy dependence on the web services in my application, the need to keep the web services performing very quickly and effectively will be vital. So we will help to lessen the load that actually be hitting the web services by implementing caching layers for content that is requested often.

### 5.4.1 Reverse Proxy

---

The purpose of a reverse proxy is to essentially have this proxy on our server side and have all the requests feed through it. By doing this, the one client will request from the web services and cache the content. Thus if 5 people request to see a list of companies, the first person will go through the reverse proxy and then hit the web services, the remaining four will call to the reverse proxy, but since the content would be cached after the first time, the web services will not be hit, and the cached content will be returned. Again, this won't be used for Admin Content, or for Updated of information. But mostly will be implemented for the most utilized calls to the system.

### 5.4.2 Phone Caching

---

There is no phone caching done with our phone application. The only caching that is done is the default caching of each operating system.





## 5.5 Database Access

---

The next decision was how the phone applications will access the database on the server. After researching, it has been advised by many that for security and performance reasons, having the application directly make a connection to a database is not a good idea.

### 5.1 Data Access Layer through Web Services

---

We will be implementing a Data Access Layer through which all database calls are made. This way the database side will be separated from the rest of the code bases. To access the data layer, it will be hooked up to web services that can be called and depending on the request parameters, can perform any CRUD functionality. This way the Admin CMS and the phone applications can all get to the database in a way that is secure and reused.



## 6 Pseudocode for System

Below are template files to be used to view the syntax and design to be followed in files of a similar nature.

### 6.1 Admin CMS

#### 6.1.1 Element Manager Page (Back-End)

```
1  using ReceiptReward;
2  using System;
3  using System.Collections.Generic;
4  using System.Diagnostics;
5  using System.Web;
6  using System.Web.UI;
7  using System.Web.UI.WebControls;
8
9  public partial class Entity_Manager : System.Web.UI.Page {
10     // Global Variables
11     private int surveyId = 0;
12     private Survey survey = null;
13     private Company company = null;
14     private string type = "survey";
15
16     protected void Page_Load(object sender, EventArgs e) {
17
18         // Call Methods
19         getVariables();
20         if (!this.IsPostBack) {
21             getTitle();
22             getList();
23         }
24     }
25
26
27     /**
28     * Get The Data for the page
29     */
30     private void getVariables() {
31         if (Request.QueryString["surveyId"] != null) { surveyId = int.Parse(Request.QueryString["surveyId"]); }
32         survey = SurveyImpl.getBySurveyId(surveyId);
33         company = CompanyImpl.getById(survey.companyId);
34         lblScripts.Text = "<script type='text/javascript'> var bigId = " + surveyId + "; bigType = \"" + type + "\";</script>";
35     }
36
37
38     /**
39     * Get the title of the page
40     */
41     private void getTitle() {
42         lblTitle.Text = company.name + " : " + survey.name;
43         txtEditName.Text = survey.name;
44     }
45 }
```





```

45
46
47  /**
48   * Creates a new element
49   */
50  protected void Add_Click(Object sender, EventArgs e) {
51      Revision revision = new Revision();
52      revision.revisionId = 0;
53      revision.active = chkAddActive.Checked;
54      revision.automation = null;
55      revision.dateCreated = DateTime.Now;
56      revision.form = null;
57      revision.revisionNumber = txtAddRevisionNumber.Text;
58      revision.surveyId = survey.surveyId;
59      RevisionImpl.create(revision);
60  }
61
62
63  /**
64   * Edit element
65   */
66  protected void Edit_Click(Object sender, EventArgs e) {
67      Survey surveyEdit = new Survey();
68      surveyEdit.surveyId = int.Parse(hdnSaveId.Value);
69      surveyEdit.name = txtEditName.Text;
70      SurveyImpl.update(surveyEdit);
71  }
72
73
74  /**
75   * Delete element
76   */
77  protected void Delete_Click(Object sender, EventArgs e) {
78      if (hdnDeleteType.Value == "survey") {
79          SurveyImpl.removeBySurveyId(int.Parse(hdnDeleteId.Value));
80      } else if (hdnDeleteType.Value == "revision") {
81          RevisionImpl.removeByRevisionId(int.Parse(hdnDeleteId.Value));
82      }
83  }
84
85
86  public void getList() {
87      // Instantiate Variables
88      string html = "";
89      int counter = 0, childrenCount = 0;
90      List<Revision> list = RevisionImpl.getBySurveyId(survey.surveyId);
91
92      // Iterate through list
93      foreach (Revision item in list) {
94
95          // Instantiate Variables
96          childrenCount = 0; //Survey.containsNumOfQuestionGroups(surveyID);
97
98          html += " " +
99              "<div style='position: absolute; top: " + (counter * 25) + "px; left: 0px; height: 50px; width: 960px;'>" +
100                  "<div class='TableElementDiv' style='width: 200px;'>" + CommonUtils.trim(item.revisionNumber, 20) + "</div>" +
101                  "<div class='TableElementDiv' style='width: 100px;'>" + item.active + "</div>" +
102                  "<div class='TableElementDiv' style='width: 200px;'>" + item.dateCreated.ToShortDateString() + "</div>" +
103                  "<div class='TableElementDiv' style='width: 200px;'>" +
104                      "<a href='Revision.aspx?revisionId=" + item.revisionId + "'>Edit</a> | " +
105                      "<a class='isLink' onclick='{\"$dlgDelete\"}.dialog(\"open\"); $(\"#hdnDeleteId\").val(\"" + item.revisionId + "\"); \" +
106                          $(\"#hdnDeleteType\").val(\"" + "revision" + "\");'>Delete</a>" +
107                  "</div>" +
108                  "</div>";
109          counter++;
110      }
111
112      // Add More to the text
113      html = "<div id='movingObjectsHolder' style='position: relative; width: 960px; height: " + ((counter - 1) * 25) + "px;'>" + html + "</div>";
114      html = " " +
115          "<div class='TableElementDiv' style='width: 200px;'><strong>Revision Number</strong></div>" +
116          "<div class='TableElementDiv' style='width: 100px;'><strong>Active</strong></div>" +
117          "<div class='TableElementDiv' style='width: 200px;'><strong>Date Created</strong></div>" +
118          "<div class='TableElementDiv' style='width: 200px;'><strong>Options</strong></div>" + html;
119
120      // Display Text
121      lblList.Text = html;
122  }

```



## 6.1.2 Element Implementation

```
1 using System;
2 using System.Collections.Generic;
3 using System.Web;
4 using System.Configuration;
5 using System.Data.SqlClient;
6 using System.Net;
7 using System.IO;
8 using Newtonsoft.Json.Linq;
9 using System.Diagnostics;
10
11 namespace ReceiptReward
12 {
13     [
14
15     public class SurveyImpl
16     {
17         // Base URL
18         private const string BASE_URL = "http://140.104.69.94:8080/RestfulWebService/html";
19         private const string DIRECTORY = SurveyImpl.BASE_URL + "/Survey";
20
21         // GET Methods
22         private const string GET_ALL = SurveyImpl.DIRECTORY + "/all";
23         private const string GET_BY_COMPANY_ID = SurveyImpl.DIRECTORY + "/ById/Company";
24         private const string GET_BY_SURVEY_ID = SurveyImpl.DIRECTORY + "/ById/Survey";
25
26         // POST Methods
27         private const string CREATE = SurveyImpl.DIRECTORY + "/Create";
28
29         // UPDATE Methods
30         private const string UPDATE = SurveyImpl.DIRECTORY + "/Update";
31
32         // DELETE Methods
33         private const string REMOVE_BY_COMPANY_ID = SurveyImpl.DIRECTORY + "/ById/Company";
34         private const string REMOVE_BY_SURVEY_ID = SurveyImpl.DIRECTORY + "/ById/Survey";
35
36
37         /**
38          * Obtains a list of all surveys in the database
39          * Calls webservice
40          * Return A list of surveys objects
41          */
42         public static List<Survey> getAll()
43         {
44             // Instantiate Variables
45             string json = "";
46             List<Survey> surveys = new List<Survey>();
47
48             // Set up Request
49             HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_ALL);
50             httpWebRequest.ContentType = "text/json";
51             httpWebRequest.Method = "GET";
52
53             // Get the Response
54             HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
55             using (StreamReader streamReader = new StreamReader(httpWebResponse.GetResponseStream()))
56             {
57                 json = "[" + streamReader.ReadToEnd() + "]";
58             }
59
60             // Convert JSON to object
61             JObject jobject = JObject.Parse(json);
62             JToken jToken = jobject["data"];
63             foreach (JToken jCode in jToken.Children())
64             {
65                 Survey survey = new Survey();
66                 survey.surveyId = (int)jCode["surveyId"];
67                 survey.companyId = (int)jCode["companyId"];
68                 survey.name = (string)jCode["name"];
69                 surveys.Add(survey);
70             }
71
72             // Return
73             return surveys;
74         }
75     }
76
77     /**
78      * Obtains a Survey by Company Id from the database
79      * Calls webservice
80      * Return A Survey object
81      */
82     public static List<Survey> getbyCompanyId(int companyId)
83     {
84         // Instantiate Variables
85         string json = "";
86         List<Survey> surveys = new List<Survey>();
87
88         // Set up Request
89         HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_BY_COMPANY_ID + "/" + companyId);
```



```

89     HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_BY_COMPANY_ID + "/" + companyId);
90     httpWebRequest.ContentType = "text/json";
91     httpWebRequest.Method = "GET";
92
93     // Get the Response
94     HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
95     using (StreamReader streamReader = new StreamReader(httpWebResponse.GetResponseStream()))
96     {
97         json = "{ \"data\": \" " + streamReader.ReadToEnd() + " \"}";
98     }
99
100    // Convert JSON to object
101    JObject jsonObject = JObject.Parse(json);
102    JToken jCodes = jsonObject["data"];
103    foreach (JToken jCode in jCodes.Children())
104    {
105        Survey survey = new Survey();
106        survey.surveyId = (int)jCode["surveyId"];
107        survey.companyId = (int)jCode["companyId"];
108        survey.name = (string)jCode["name"];
109        surveys.Add(survey);
110    }
111
112    // Return
113    return surveys;
114 }
115
116
117 /**
118  * Obtains a Survey by Id from the database
119  * Calls webservice
120  * @return A Survey object
121  */
122 public static Survey getById(int surveyId)
123 {
124     // Instantiate Variables
125     string json = "";
126     Survey survey = new Survey();
127
128     // Set up Request
129     HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_BY_SURVEY_ID + "/" + surveyId);
130     httpWebRequest.ContentType = "text/json";
131     httpWebRequest.Method = "GET";
132
133     // Get the Response
134     HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
135     using (StreamReader streamReader = new StreamReader(httpWebResponse.GetResponseStream()))
136     {
137         json = "{ \"data\": \" " + streamReader.ReadToEnd() + " \"}";
138     }
139
140    // Convert JSON to object
141    JObject jsonObject = JObject.Parse(json);
142    JToken jCodes = jsonObject["data"];
143    foreach (JToken jCode in jCodes.Children())
144    {
145        survey.surveyId = (int)jCode["surveyId"];
146        survey.companyId = (int)jCode["companyId"];
147        survey.name = (string)jCode["name"];
148    }
149
150    // Return
151    return survey;
152 }
153
154
155 /**
156  * Create a new entry in the Database
157  * Calls webservice
158  */
159 public static void create(Survey survey)
160 {
161     // Set up Request
162     HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.CREATE);
163     httpWebRequest.ContentType = "application/json";
164     httpWebRequest.Method = "POST";
165
166     // Send Request Data
167     using (StreamWriter streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
168     {
169         string json = "{ \"surveyId\": \" " + survey.surveyId + "\", \"companyId\": \" " + survey.companyId + "\", \"name\": \" " + survey.name + " \"}";
170         Debug.WriteLine(json);
171         streamWriter.Write(json);
172         streamWriter.Flush();
173         streamWriter.Close();
174     }
175
176     // Get the Response
177     HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();

```



```

177     HttpResponseMessage httpResponse = (HttpResponseMessage)httpWebRequest.GetResponse();
178 }
179
180
181 /**
182  * Update entry in the Database
183  * Calls webservice
184  */
185 public static void update(Survey survey)
186 {
187     // Set up Request
188     HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.UPDATE);
189     httpWebRequest.ContentType = "application/json";
190     httpWebRequest.Method = "PUT";
191
192     // Send Request Data
193     using (StreamWriter streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
194     {
195         string json = "{\"surveyId\":\"\" + survey.surveyId + "\",\"name\":\"\" + survey.name + "\"}";
196         Debug.WriteLine(json);
197         streamWriter.Write(json);
198         streamWriter.Flush();
199         streamWriter.Close();
200     }
201
202     // Get the Response
203     HttpResponseMessage httpResponse = (HttpResponseMessage)httpWebRequest.GetResponse();
204 }
205
206
207 /**
208  * Removes an entry in the Database by Company Id
209  * Calls webservice
210  */
211 public static void removeByCompanyId(int companyId)
212 {
213     // Set up Request
214     HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.REMOVE_BY_COMPANY_ID + "/" + companyId);
215     httpWebRequest.ContentType = "application/json";
216     httpWebRequest.Method = "DELETE";
217
218     // Get the Response
219     HttpResponseMessage httpResponse = (HttpResponseMessage)httpWebRequest.GetResponse();
220 }
221
222
223 /**
224  * Removes an entry in the Database by Survey Id
225  * Calls webservice
226  */
227 public static void removeBySurveyId(int surveyId)
228 {
229     // Set up Request
230     HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.REMOVE_BY_SURVEY_ID + "/" + surveyId);
231     httpWebRequest.ContentType = "application/json";
232     httpWebRequest.Method = "DELETE";
233
234     // Get the Response
235     HttpResponseMessage httpResponse = (HttpResponseMessage)httpWebRequest.GetResponse();
236 }
237
238
239 /**
240  * Get the breadcrumbs
241  */
242 static public string getBreadcrumbs(int id)
243 {
244     // Instantiate Variables
245     string tempStr = "";
246     Survey survey = SurveyImpl.getBySurveyId(id);
247     Company company = CompanyImpl.getId(survey.companyId);
248
249     // Survey
250     tempStr = "<a class='breadcrumbCurrent' href='Survey.aspx?surveyId=" + survey.surveyId + "'> " + survey.name + "</a>" + tempStr;
251
252     // Company
253     tempStr = "<a class='breadcrumbNotCurrent' href='Company.aspx?companyId=" + company.companyId + "'> " + company.name + "</a> >> " + tempStr;
254
255     // Homepage
256     tempStr = "<a class='breadcrumbNotCurrent' href='Index.aspx'>Home</a> >> " + tempStr;
257
258     // Output Links
259     return tempStr;
260 }
261
262
263 }

```



## 6.2 Web Services

### 6.2.1 Service

```
1 package com.bti.ws.controller;
2
3 import com.bti.ws.implementations.SurveyImpl;
4 import com.bti.ws.model.Survey;
5 import java.util.List;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.stereotype.Controller;
8 import org.springframework.web.bind.annotation.ExceptionHandler;
9 import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.RequestBody;
11 import org.springframework.web.bind.annotation.RequestMapping;
12 import org.springframework.web.bind.annotation.RequestMethod;
13 import org.springframework.web.bind.annotation.ResponseBody;
14 import org.springframework.web.bind.annotation.ResponseStatus;
15
16 @Controller
17 @RequestMapping("/Survey")
18 public class SurveyController {
19
20     @RequestMapping(value = "/All", method = RequestMethod.GET)
21     @ResponseBody
22     List<Survey> getAll() {
23         return SurveyImpl.findAll();
24     }
25
26     @RequestMapping(value = "/ById/Company/{companyId}", method = RequestMethod.GET)
27     @ResponseBody
28     List<Survey> getByCompanyId(@PathVariable("companyId") int companyId) {
29         return SurveyImpl.findByCompanyId(companyId);
30     }
31
32     @RequestMapping(value = "/ById/Survey/{surveyId}", method = RequestMethod.GET)
33     @ResponseBody
34     List<Survey> getBySurveyId(@PathVariable("surveyId") int surveyId) {
35         return SurveyImpl.findBySurveyId(surveyId);
36     }
37
38     @RequestMapping(value = "/Create", method = RequestMethod.POST)
39     @ResponseStatus(HttpStatus.OK)
40     void create(@RequestBody Survey survey) {
41         SurveyImpl.create(survey);
42     }
43
44     @RequestMapping(value = "/Update", method = RequestMethod.PUT)
45     @ResponseStatus(HttpStatus.OK)
46     void update(@RequestBody Survey survey) {
47         SurveyImpl.update(survey);
48     }
49
50     @RequestMapping(value = "/ById/Survey/{surveyId}", method = RequestMethod.DELETE)
51     @ResponseStatus(HttpStatus.OK)
52     void removeBySurveyId(@PathVariable("surveyId") int surveyId) {
53         SurveyImpl.deleteBySurveyId(surveyId);
54     }
55
56     @RequestMapping(value = "/ById/Company/{companyId}", method = RequestMethod.DELETE)
57     @ResponseStatus(HttpStatus.OK)
58     void removeByCompanyId(@PathVariable("companyId") int companyId) {
59         SurveyImpl.deleteByCompanyId(companyId);
60     }
61
62     @ExceptionHandler(NotFoundException.class)
63     @ResponseStatus(value = HttpStatus.NOT_FOUND, reason="Code Type not found")
64     void handleNotFound(NotFoundException exc) {}
65
66     class NotFoundException extends RuntimeException {
67         private static final long serialVersionUID = 1L;
68     }
69 }
70
71
```



## 6.2.2 Implementation

```
1 package com.bti.ws.implementations;
2
3 import com.bti.ws.common.DBConnection;
4 import com.bti.ws.model.Survey;
5 import java.sql.Connection;
6 import java.sql.PreparedStatement;
7 import java.sql.ResultSet;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class SurveyImpl {
12
13     private static final String FIND_ALL = "SELECT * FROM survey;";
14     private static final String FIND_BY_SURVEY_ID = "SELECT * FROM survey WHERE survey_id = ?;";
15     private static final String FIND_BY_COMPANY_ID = "SELECT * FROM survey WHERE company_id = ?;";
16     private static final String FIND_NEXT_ID = "SELECT TOP 1 survey_id FROM survey ORDER BY survey_id DESC;";
17     private static final String CREATE = "INSERT INTO survey (survey_id, company_id, name) VALUES (?, ?, ?);";
18     private static final String UPDATE = "UPDATE survey SET name = ? WHERE survey_id = ?;";
19     private static final String DELETE_BY_SURVEY_ID = "DELETE FROM survey WHERE survey_id = ?;";
20     private static final String DELETE_BY_COMPANY_ID = "DELETE FROM survey WHERE company_id = ?;";
21
22
23     /***** Find Methods *****/
24     public static List<Survey> findAll() {
25         // Instantiate Variables
26         Connection con = null;
27         PreparedStatement ps = null;
28         ResultSet rs = null;
29         List<Survey> surveys = new ArrayList<Survey>();
30
31         try { // Get List of All
32             con = DBConnection.getConnection();
33             ps = con.prepareStatement(SurveyImpl.FIND_ALL);
34             rs = ps.executeQuery();
35             while(rs.next()) {
36                 surveys.add(new Survey(rs.getInt("survey_id"), rs.getInt("company_id"), rs.getString("name")));
37             }
38             rs.close();
39             ps.close();
40         } catch(Exception ex) {
41             System.out.println("Error: " + ex.getMessage());
42             // @TODO: Handle Exception
43         } finally {
44             try {
45                 if(con != null) {
46                     con.close();
47                     con = null;
48                 }
49             } catch(Exception ex){}
50         }
51
52         // Return
53         return surveys;
54     }
55
56     public static List<Survey> findByCompanyId(int companyId) {
57         // Instantiate Variables
58         Connection con = null;
59         PreparedStatement ps = null;
60         ResultSet rs = null;
61         List<Survey> surveys = new ArrayList<Survey>();
```



```

63     try { // Get List By Id
64         con = DBConnection.getConnection();
65         ps = con.prepareStatement(SurveyImpl.FIND_BY_COMPANY_ID);
66         ps.setInt(1, companyId);
67         rs = ps.executeQuery();
68         while(rs.next()) {
69             surveys.add(new Survey(rs.getInt("survey_id"), rs.getInt("company_id"), rs.getString("name")));
70         }
71         rs.close();
72         ps.close();
73     } catch(Exception ex) {
74         System.out.println("Error: " + ex.getMessage());
75         // TODO: Handle Exception
76     } finally {
77         try {
78             if(con != null) {
79                 con.close();
80                 con = null;
81             }
82         } catch(Exception ex) {}
83     }
84
85     // Return
86     return surveys;
87 }
88
89 public static List<Survey> findBySurveyId(int surveyId) {
90     // Instantiate Variables
91     Connection con = null;
92     PreparedStatement ps = null;
93     ResultSet rs = null;
94     List<Survey> surveys = new ArrayList<Survey>();
95
96     try { // Get List By Id
97         con = DBConnection.getConnection();
98         ps = con.prepareStatement(SurveyImpl.FIND_BY_SURVEY_ID);
99         ps.setInt(1, surveyId);
100        rs = ps.executeQuery();
101        while(rs.next()) {
102            surveys.add(new Survey(rs.getInt("survey_id"), rs.getInt("company_id"), rs.getString("name")));
103        }
104        rs.close();
105        ps.close();
106    } catch(Exception ex) {
107        System.out.println("Error: " + ex.getMessage());
108        // TODO: Handle Exception
109    } finally {
110        try {
111            if(con != null) {
112                con.close();
113                con = null;
114            }
115        } catch(Exception ex) {}
116    }
117
118    // Return
119    return surveys;
120 }
121

```



```

121
122 public static int findNextId() {
123     // Instantiate Variables
124     Connection con = null;
125     PreparedStatement ps = null;
126     ResultSet rs = null;
127     int surveyId = 1;
128
129     try { // Get All the Code Types
130         con = DBConnection.getConnection();
131         ps = con.prepareStatement(SurveyImpl.FIND_NEXT_ID);
132         rs = ps.executeQuery();
133         while(rs.next()) {
134             surveyId = rs.getInt("survey_id") + 1;
135         }
136         rs.close();
137         ps.close();
138     } catch(Exception ex) {
139         System.out.println("Error");
140         // @TODO: Handle Exception
141     } finally {
142         try {
143             if(con != null) {
144                 con.close();
145                 con = null;
146             }
147         } catch(Exception ex){}
148     }
149
150     // Return
151     return surveyId;
152 }
153
154
155 /***** Create Methods *****/
156 public static void create(Survey survey) {
157     // Instantiate Variables
158     Connection con = null;
159     PreparedStatement ps = null;
160
161     try { // Set ID
162         survey.setSurveyId(SurveyImpl.findNextId());
163
164         // Create New Entry
165         con = DBConnection.getConnection();
166         ps = con.prepareStatement(SurveyImpl.CREATE);
167         ps.setInt(1, survey.getSurveyId());
168         ps.setInt(2, survey.getCompanyId());
169         ps.setString(3, survey.getName());
170         ps.executeUpdate();
171         ps.close();
172     } catch(Exception ex) {
173         System.out.println("Error: " + ex.getMessage());
174         // @TODO: Handle Exception
175     } finally {
176         try {
177             if(con != null) {
178                 con.close();
179                 con = null;
180             }
181         } catch(Exception ex){}

```





```

181         } catch(Exception ex){}
182     }
183 }
184
185
186 /***** Update Methods *****/
187 public static void update(Survey survey) {
188     // Instantiate Variables
189     Connection con = null;
190     PreparedStatement ps = null;
191
192     try { // Update Existing Entry
193         con = DBConnection.getConnection();
194         ps = con.prepareStatement(SurveyImpl.UPDATE);
195         ps.setString(1, survey.getName());
196         ps.setInt(2, survey.getSurveyId());
197         ps.executeUpdate();
198         ps.close();
199     } catch(Exception ex) {
200         System.out.println("Error: " + ex.getMessage());
201         // @TODO: Handle Exception
202     } finally {
203         try {
204             if(con != null) {
205                 con.close();
206                 con = null;
207             }
208         } catch(Exception ex){}
209     }
210 }
211
212
213 /***** Delete Methods *****/
214 public static void deleteBySurveyId(int surveyId) {
215     // Instantiate Variables
216     Connection con = null;
217     PreparedStatement ps = null;
218
219     try { // Delete Existing Entry
220         con = DBConnection.getConnection();
221         ps = con.prepareStatement(SurveyImpl.DELETE_BY_SURVEY_ID);
222         ps.setInt(1, surveyId);
223         ps.executeUpdate();
224         ps.close();
225     } catch(Exception ex) {
226         System.out.println("Error: " + ex.getMessage());
227         // @TODO: Handle Exception
228     } finally {
229         try {
230             if(con != null) {
231                 con.close();
232                 con = null;
233             }
234         } catch(Exception ex){}
235     }
236 }
237
238 public static void deleteByCompanyId(int companyId) {
239     // Instantiate Variables
240     Connection con = null;
241     PreparedStatement ps = null;
242
243     try { // Delete Existing Entry
244         con = DBConnection.getConnection();
245         ps = con.prepareStatement(SurveyImpl.DELETE_BY_COMPANY_ID);
246         ps.setInt(1, companyId);
247         ps.executeUpdate();
248         ps.close();
249     } catch(Exception ex) {
250         System.out.println("Error: " + ex.getMessage());
251         // @TODO: Handle Exception
252     } finally {
253         try {
254             if(con != null) {
255                 con.close();
256                 con = null;
257             }
258         } catch(Exception ex){}
259     }
260 }
261 }

```

