Aurangzeb Lodhi

E-Acquisition System For Images in Mobile Environment

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

18 April 2017

Author(s) Title Number of Pages Date	Aurangzeb Lodhi E-Acquisition System For Images In Mobile Environment 50 pages 18 April 2017
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Principal Lecturer

The main goal of the project was to create a new mobile application for the employees and sub-contractors of the case company. It allows storing of files and images directly into the company's database from a project location.

The case company provides planning and consulting services for constructions projects of bridges and concrete structures, railways, urban planning and land use. The rapid expansion of the company's construction business has caused the need for a mobile software application that allows its subcontractors to communicate with the company's database on-site and remotely. This thesis focuses on a mobile software design to facilitate the management of useful data to and from the company's construction project at particular sites as well as a tool for users to interact with the company's database.

Previously, the company used its own website as a way to communicate with their contractors and sub-contractors and to inform and update its customers. With the latest trends on the market, some customers are active users of interactive websites that may not be user friendly for mobile phones whereas others need a mobile solution for interaction when they are on the field. Ignoring a group that does not use interactive websites or do not have the time to use laptops and PCs that use interactive websites would result in a huge loss for the growing business.

This thesis addresses and tackles these issues in three steps. First, the purpose was to create a mobile software application to interact remotely with the employees and clients of the company. This included a careful selection of the elements of the company's website that can actually be utilized by mobile customers and employees. This directly provides an alternative to the website the company has long relied on for interaction with its employees and clients. The next phase included the design and development of a custom Windows Phone application for the company's extranet.

In the last phase of the project, Windows phone built-in features were utilized to show their interaction with the application and the results were analyzed.

Keywords	API, JSON, REST, C#, Windows Phone 7/8

Contents

1 Introduction

Technology has transformed human lives. As enterprises strive to make their business more approachable by allowing their customers to manage user data and information seamlessly – business apps in a way are helping them to work smarter and emphasize on what actually matters. In fact, in many ways, the mobile application today is what the website was ten years ago — one of those tools that has transferred from being a luxury into a necessity for businesses of all sizes. Some of the most preferred mobile apps for business include Evernote (helps people to store everything virtually), Google Drive (lets users port and edit files from PC to mobile devices - seamlessly), and ProtoPrompt (the database program that supports uploaded photographs - to provide visual means of searching through personnel photos in order to identify people quickly in a business setting).

1.1 Background

The case company operates in the field of infrastructure, traffic solutions, logistics, land use, environment and digital services. It has got projects all over Finland with sub-contractors working in various places and projects. Project information has been shared between sub-contractors and employees on the company's webpage and stored in the company's database. Using the company software, a user/sub-contractor has the option to add a project, join an existing project, to see the documents and pages attached to a project and also the articles related to a particular project.

The company was looking for a mobile software application that is user friendly and allows sharing of information between employees and sub-contractors directly from the project site hence eliminating the need to use a web-based application while on the move.

The aim of the project was to develop an application for e-acquisition of images in the mobile environment. The developed application helps the end user interact with the company's extranet to extract data, articles and documents related to a project on-site and off-site. It allows getting images related to a project and read documents of importance. A user also gets an option to capture an image on-site and store it directly on to the company’s database. It is a useful hands-on tool for company’s projects spread across various locations.

1.2 Technology Problem

The company software has a number of users, which include employees and external users that use mobile phones on the go. Ignoring a group that does not have the time or access to use laptops and PC’s is reducing the efficiency of the business.

The company’s internal database can handle huge amounts of data at a time. The company needs a mobile software application that can interact with the database in a form of an intermediate connection such that not all the elements of the actual web application are utilized. In fact, the company required a custom built mobile application that utilizes only the main elements of the actual web based application and information associated to it. By focusing on an application that includes only its core functionality helps to minimize the time the user needs to spend using the mobile application, thus, increasing the value of the application to the user.

1.3 Research Plan and Scope

This thesis addresses and tackles the targets of the development project in order to provide a cost effective solution. The users are involved in one or more projects such as construction or railway projects. Each individual’s information, the number of projects and resource material are available in the company’s dashboard. The company has expanded in recent years and needs a solution through which a customer or company sub-contractor goes to a site or a location of interest, takes information from on-site in the form of a photo or a file and uploads it in their project space to be viewed as a single picture or a collection of pictures. Thus, it saves time and energy as a user can interact with certain database features anywhere by using their mobile phone. The solution serves as a data acquisition tool in the form of a mobile application that is easy to use, reliable and efficient.

The application is targeted for company employees and also the guest users with login credentials. It is a native application and the company plans to design and publish it for windows platform in the first phase. Native applications are faster and more efficient, as they work in tandem with the mobile device they are developed for. Also, they have guaranteed quality, as users can access them only via an online store available for a particular platform.

The project included following steps:

1. Researching different design implementation strategies for the company's internal software application, 'Kaiku Mobiili'.
2. Analysing the API access points and their role in application design
3. Identifying the resources that are going to be accessible through the API
4. Performance analysis of test cases according to previously established requirements.
5. Analyzing measures of Mobile Security throughout the entire software development life cycle (SDLC) so that vulnerabilities may be addressed in a timely and thorough manner.

The project was scoped with a high-level system architecture design and mobile software implementation as an alternative tool to the company's website. The purpose of the study was to make a first version (prototype) of the application and get initial feedback with it. The prototype was developed for Windows Platform as a native application. Although, the application is not designed for other platforms such as iOS and Android platform, it does provide a base to expand the application further to other platforms.

This thesis consists of six chapters.

Chapter one includes a brief introduction of the project, its background, challenges and scope of the study. Chapter two gives a deep insight on the technical background of the project discussing API concepts and their use in commercial projects. Chapter three focuses on the Windows platform architecture and features utilized in mobile software development. Chapter four describes the application development plan. Chapter five focuses on the testing of the application and the results of the project. Chapter six gives the final conclusion of the project and introduces some ideas about potential future steps.

2 Theoretical Background

This chapter describes the application-programming interface (API), different types of APIs, their usage in commercial applications, different types of languages and their approach in programming. It also provides a deep insight into the REST architectures, its different forms and its use in the API development.

2.1 Technical Background

The need to efficiently share vast amounts of data across various departments and with clients and employees is an issue that most organizations face today.
A key tool to tackle this challenge is the Application Programming Interface (API), which at its most basic acts as a door or window into a software program, allowing other programs to interact with it without the need for a developer to share its entire code.
For example, an API would allow a mobile app, set top box or other connected device in a home to communicate with a service. The company exposes an API that tells a programmer how they will interact with the service. The API can be opened to customers or just people inside their own company.

The recent development of apps for mobile devices means organizations need to allow users to access information through apps and not just through the Internet.
Within the public sector, APIs are used to allow agencies to easily share information and also let the public interact with government as well.

In order to understand the concept of APIs, it is important to look back at the period when this concept was introduced sometime before the year 2000, slightly before the .COM bubble. Software developers were in a need of standard ways to create a bridge between applications and allowing them to communicate with each other lead to the developments of APIs.

2.2 API Theory

An application-programming interface (API) is a set of routines, protocols, and tools for building software applications. In its simplest form, API is defined as an interface that enables a software program to interact with other programs. A software company releases its API to the public so that other software developers can design products that are powered by its service.

For example, Amazon.com released its API so that Web site developers could more easily access Amazon's product information. Using the Amazon API, a third party Web site can post direct links to Amazon products with updated prices and an option to "buy now." [1]

An API is a software-to-software interface, not a user interface. With APIs, applications talk to each other without any user knowledge or intervention. When one buys movie tickets online and enters the credit card information, the movie ticket Web site uses an API to send the credit card information to a remote application that verifies whether the information is correct. Once the payment is confirmed, the remote application sends a response back to the movie ticket Web site saying it is alright to issue the tickets.

As a user, one only sees one interface -- the movie ticket Web site -- but behind the scenes, many applications are working together using APIs. This type of integration is called ’seamless’, since the user never notices when software functions are handed from one application to another. An API resembles Software as a Service (SaaS). Instead of building one core application that tries to do everything, such as e-mail, billing, tracking, etcetera, the same application can contract out certain responsibilities to remote software that does it better.

2.3 Types of Application Programmable Interface (API)

There are several classifications of APIs based on how they are used:

- Internal APIs are used exclusively within an organization or company.
- External APIs are primarily available externally to consumers. At this stage of maturity, the growing trend for external APIs are written based on REST/JSON technologies. They provide access and integration capabilities that are easier to use than the more industrial-strength capabilities leveraging web services (for example, WSDLs).

- Partner APIs are specifically designed for partners to be able to access business functions in relation to the business relationship of the partnership. Examples include online catalogue, ordering, and reconciliation. Over time, other categories of APIs might come available. [2]

2.4 API documentation

API documentation provides information about services an API offers and knowledge of how to use those services. It aims to cover everything a client would need to know to use the API. Documentation is an integral part of development and maintenance of applications that use the API. Traditionally, API documentation is found in documentation files, however it can also be found in social media sites such as blogs, forums, and other Q&A websites. Documentation files are often presented via a documentation system, such as Javadoc pr Pydoc that has a consistent structure and appearance. The type of content included in the documentation however differs from API to API [3]. API documentation can include description of classes and methods used in the API as well as typical usage scenarios, code snippets, design rationales, contracts and performance discussions. Details of implementation of the API services are usually omitted by the case company to keep the process confidential. There are also restrictions and limitations on how the API can be used. Since API documentation is quite comprehensive, it may be difficult for the writers to keep the documentation updated and for the users to follow it. This can potentially result in bugs which may be difficult to fix.

2.5 REST and Other Open Source Application Programming Interfaces

According to ProgrammableWeb.com [4], out of 30861 APIs, 73% are implemented with the REST architectural style. The percentages in Figure 1 below need to be handled with care. On one hand they just represent the percentages of APIs listed on ProgrammableWeb.com. Therefore only the APIs which are indexed by ProgrammableWeb.com and which are also publicly available are listed. On the other hand it could well be that some of these REST APIs are GET-only APIs, which are of course implemented according to REST principles but are not real APIs. They may lack data operations such as updating or deleting resources. Also, many of the private APIs, which are used in internal business environments, are implemented in a SOAP style. Nevertheless, the figure below emphasizes the importance of REST in the context of public web services.

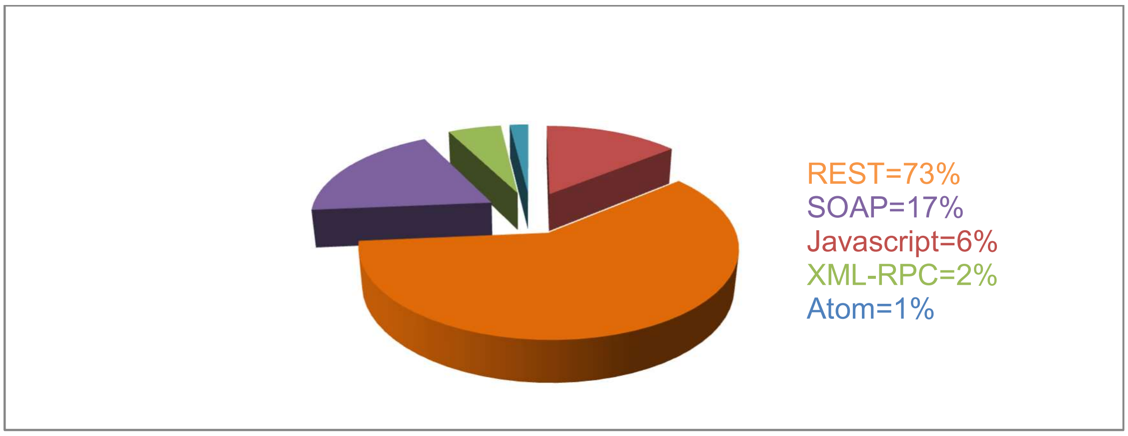


Figure1. Protocol Usage by API's. [5]

REST in the context of web services defines concrete design principles which describe a resource-oriented web service. A resource-oriented architecture serves the purpose of making resources accessible. In the context of REST: resources are data entities. Some architecture might fulfill the REST criteria better than another but there is no such thing as one true REST architecture.

2.6 A REST Design

REST is a software design pattern typically used for web applications. In layman's terms this means that it is a commonly used idea in many different projects. It stands for REpresentational State Transfer. The basic idea of REST is treating objects on the server-side (as in rows in a database table) as resources than can be created or destroyed.

The most basic way of thinking about REST is as a way of formatting the URL's of the web applications in use. For example, if the resource was called "posts", then:

- /posts is how a user would access ALL the posts, for displaying.
- /posts/ :id is how a user would access and view an individual post, retrieved based on their unique id.

- /posts/new is how a user would display a form for creating a new post.Sending a POST request to /users is how a user would actually *create* a new post on the database level.
- Sending a PUT request to /users/ :id is how a user would update the attributes of a given post, again identified by a unique id.
- Sending a DELETE request to /users/ :id is how a user would delete a given post, again identified by a unique id.

When the term "RESTful API," is used, it generally means an API that uses RESTful URL'S for retrieving data.

2.7 Features of REST

REST is typically used over HTTP, primarily due to the simplicity of HTTP and its very natural mapping to RESTful principles. REST, however, is not tied to any specific protocol. It includes the following features:

Statelessness

Statelessness means that the REST server can fulfill the client request in complete isolation. As a result the client sends all the information the server needs with the request. The server does not need to maintain a session. This is one of the reasons why the architectural style of REST can build highly scalable solutions. The server can process information without storing sessions, which need a lot of memory and calculations.

Addressability

This principle says that every resource should be made available through a unique address. In a real world this address is described by the use of Unique Resource Identifiers (URI). This principle is also crucial for the web. Resources, for example an image, should be available through an URI. On the web this is not always the case, due to Asynchronous Java script and XML (AJAX) applications, which can load content dynamically without changing the URI. To get a resource of a REST service, one can

request a URI. In a real world, APIs try to keep the URI design as logical as possible. Therefore the URI's are often hierarchically designed. Hierarchies increase the readability and make it easier for developers to address the resources. Also, it is possible to guess resources, which make it easier for clients to discover resources deductively.
Uniform Interface

A uniform interface only offers a set of operations. But it still offers enough operations to retrieve, change or create data. The simplicity of the uniform interface is important to REST, because it keeps the interaction between client and server as simple as possible. In practice HTTP is used as the uniform interface of REST services. But REST does not dictate a particular protocol. Technically, REST can use any uniformed protocol. But because the HTTP offers all the necessary operations, is well known and widely spread, it has become the de-facto standard for REST services. All the interaction between clients and resources are based on a few basic HTTP methods.

GET
The GET method of HTTP describes a request for information about a resource. To a GET request, the server will respond with a set of headers and a representation containing the requested resource.

POST
Like a PUT request, a POST request can create a resource. The difference is that it is not bound to a specified URI. Normally, POST is used when the client sends data to the server and the server then tells the client where it put the data. But the server can do anything with the POST request. As mentioned, it can store it under the given URI (like PUT) but it can also send back a HTTP header or do nothing at all [6].

Client-server architecture

The client-server style is the most frequently encountered of the architectural styles for network-based applications. A server component, offering a set of services, listens for requests upon those services. A client component, desiring that a service be performed, sends a request to the server via a connector. The server either rejects or performs the request and sends a response back to the client. A client is a triggering process; a server is a reactive process. Clients make requests that trigger reactions from servers. Thus, a client initiates activity at times of its choosing. It often then delays until its request has been serviced. On the other hand, a server waits for requests to be made and then reacts to them. A server is usually a non-terminating process and often provides service to more than one client. Separation of concerns is the principle behind the client-server constraints. By separating the user interface concerns from the data storage concerns, one improves the portability of the user interface across multiple platforms and improves scalability by simplifying the server components [7].

Representation Oriented Architectural Style

Representations describe the format in which data is being exchanged between server and client. Common are XML, JSON and HTML. But because there is a strict difference between the resource and its representation, the resource can be converted to any representation. This is one of the reasons why REST architectures are said to be loosely coupled. The representation can also handle the language of the content. Practically speaking, the representation oriented architectural style allows servers to give the client exactly the format it demands. It is easy to extend the REST API with a new representation.

2.8 JSON Approach in Programming

JSON stands for JavaScript Object Notation and it is a type of format, which stores various types of information, and allows this information to be shared between client and server applications. Because JSON is a lightweight data-interchange format, it is easy to read and write. It is also easy for computers to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. A JSON object is a collection of key and value pairs. The keys are strings and the types of values presented in JSON can be strings, numbers, Booleans, object, arrays, or even NULL. A colon separates the keys from the values, and a comma separates the pairs. The pairs of keys and values are wrapped in curly braces and the four arrays are wrapped in square brackets [8].

As a simple example, information about the author might be written in JSON as follows

```
var jason = {
                "age" : "24",
                "hometown" : "Missoula, MT",
                "gender" : "male"
};
```

This creates an object that can be accessed using the variable jason. By enclosing the variable's value in curly braces indicate that the value is an object. Inside the object, any number of properties can be declared using a “name”: “value” pairing, separated by commas. To access the information stored in jason, name of the property can be referred. For instance, the following snippets can be used:

```
document.write('Jason is ' jason.age); // Output: Jason is 24
document.write('Jason is a ' jason.gender); // Output: Jason is a male
```

The JSON format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML. JSON libraries or built-in JSON support exists for many programming languages and systems [9].

2.9 Structure of XML

Extensible Markup Language (XML) describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is in a format that is both human readable and machine-readable. XML is are created to store and transport data. The design goals of XML emphasize simplicity, generality, and usability over the Internet. Similarly to JSON, the XML has a basic format of key and value pairs and they are designed to be self-descriptive with information wrapped in tags. However, XML is designed to store data rather than displaying data.
By definition, an XML document is a string of characters. The characters making up an XML document are divided into markup and content, which may be distinguished by the application of simple syntactic rules. Generally, strings that constitute markup begin with the character \<" and end with a \>”. These strings are called tags. There are start-tags and end-tags. The strings in the tags are the \keys", i.e., they can be referred to as

the names of the variables. The end-tags start with \/" and the strings in the end-tags must match those in the start tags. Between the start-tag and the end-tag, there can be one value or other pairs of keys and values. If an XML object consists of key/value pairs within a start-tag, the contents within the start-tag are describing the attributes of the XML object.

XML has many advantages, for instance, it is highly versatile making it adaptable to every kind of situation; one can use xml by creating new tags and nesting them regardless of the type of a project one is working on. A large number of developers know this language and in almost all cases it is quite easy to manipulate. Sometimes, it’s possible that too many tags can make a document hard to read, especially for developers. JSON has a very simple structure, which a developer can read easily even upon first sight; this is not something one want to underestimate, especially when one’s working on large projects involving a lot of people [10].

2.10 XAML

Extensible Application Markup Language, or XAML (pronounced "zammel"), is an XML-based markup language developed by Microsoft. XAML is the language behind the visual presentation of an application that one develops in Microsoft Expression Blend, just as HTML is the language behind the visual presentation of a Web page. Creating an application in Expression Blend means writing XAML code either by hand or visually by working in the design view of Expression Blend. XAML is a declarative markup language. It simplifies creating a UI for a .NET Framework application. One can create visible UI elements in the declarative XAML markup, and then separate the UI definition from the run-time logic by using code-behind files, joined to the markup through partial class definitions. XAML directly represents the instantiation of objects in a specific set of backing types defined in assemblies. This is unlike most other markup languages, which are typically an interpreted language without such a direct tie to a backing type system. XAML enables a workflow where separate parties can work on the UI and the logic of an application, using potentially different tools [11].

When represented as text, XAML files are XML files that generally have the .xaml extension. The files can be encoded by any XML encoding, but encoding as UTF-8 is typical.

2.11 Security Considerations

Security when building client-web services covers a wide range of issues and different aspects. Authenticating users to access a resource, confidentiality of information, preventing unauthorized agents form accessing resources and following the laws of the country must be considered carefully. There is no single conventional method to follow but each application requires a specific analysis in the architecture and design pattern. When a client accesses a protected resource, the sever uses an authenticate header to challenge the client to provide the expected information. The client uses authorization header to provide the proper answer. This kind of authentication schema can be used when a client accesses a protected resource on behalf of itself or user. Developed in 2007 OAuth is an authorization protocol that allows user to give permission for clients to access their data on the server without revealing their identity. OAuth protocol can be categorized as two-legged or three-legged authentication solution depending on the number of parties involved in the authentication process. The name three-legged authentication is given to OAuth because there are three parties involved in the authentication process: the service provider (the server), the OAuth consumer (the client) and the user. For two-legged OAuth, the parties involved in the authentication process are only the service provider (the server) and the OAuth consumer (the client).

The following list shows the steps in the authentication of three-legged OAuth:

- Client request authorization to access resource from the user
- The user grant authorization for the client
- The client request access token from the server
- The server issues access token to the client
- The client request resource from the server
- The server provides the resource for the client

A token based authentication is modelled like the Facebook or twitter API. It means that authorization data is also encrypted. If one can afford the performance cost, a HTTPS connection is used to perform authentication and authorization.

3 Introduction to Windows Phone Platform

Microsoft developed a series of mobile operating systems such as Windows Phone platform. It was a sequence of the previous Windows Mobile platform. Mobile phone industry was revolutionized in 2008 when Microsoft discarded the old Symbian operating system and started developing Windows Phone platform. It was to cope up with a huge demand of touchscreen phones and use of social media applications and resources over the internet.

The old windows phone interface consisted of icons and was replaced by an interface that was based upon a fresh and new design style called ‘Metro’. The metro design consisted of new coloured icons/tiles and instructions in the form of text to navigate between different screens in the Windows Phone user interface. The design was focused more on the typography and there was less emphasis on the graphical design. Products like Zune, Windows Phone, Xbox and the soon to be released Windows 8 follow the Metro design style. The design style's main objective is to remove any chrome and extra decoration and allow users to interact directly with content [12].

“Windows Phone 7 is the first generation of the Windows Phone mobile operating system released on October 2012. Windows Mobile 7 development is done using the .NET framework. The .NET framework is a software framework created by Microsoft for use in creating Windows applications. Programmers write applications using one of the several languages supported by the .NET framework, like C#, and the applications then execute inside of a runtime environment called the Common Language Runtime. For Windows Phone 7, there are two distinct development approaches when creating an application.

The first approach is to use Silverlight for Windows Phone. Silverlight was originally envisioned as a way for developers to create rich internet applications. It has seen a sharp increase in market adoption in recent years, driven mostly by the fact that Netflix uses Silverlight to stream videos and NBC used Silverlight for its online broadcast of the Olympic games. A Silverlight application combines declarative markup (called XAML) to construct the user interface and code written in a .NET framework language to control an application’s behavior. Silverlight is often used for developing a data driven application for Windows Phone 7” [21].

Alternatively, the XNA framework is an alternative to Silverlight to develop a Windows Phone 7 app. XNA is Microsoft’s game development framework and has been used in recent years to create both Windows and Xbox 360 applications. When creating a game for Windows Phone 7, the XNA framework is utilized. The XNA framework is quite powerful, but that power comes with a considerable learning curve and longer development cycles [13].

3.1 Standard Hardware

In order to provide a more consistent experience between devices, Windows Phone devices are required to meet a certain set of hardware requirements. These are:

- A common set of hardware controls and buttons that include “Start”, “Search”, and “Back” buttons.
- A large WVGA (800 x 480) format display capable of rendering most web content in full-page width and displaying movies in widescreen.
- At least a 1 GHz ARMv7 Cortex/Scorpion or better processor.
- Capacitive multi-point multi-touch screens for a quick touch response and use of phone and its features
- Data connectivity support using mobile networks and Wi-Fi to connect to a fixed network.
- At least 256 MB of RAM to run phone applications and hardware.
- A-GPS system to support current maps.
- Accelerometer.

3.2 Optional Hardware

Here are some features that Microsoft does not explicitly require its phone makers to include with a Windows Phone:

- Compass.
- Gyro.
- Removable storage

- Camera with video recording capabilities.
- Ultra high resolution screens and custom resolutions.
- Front-facing Camera [14].

“The first major update to Windows Phone 7, called “Mango”, was released in May 2011. Although the operating system internally identifies itself as version 7.1, it was marketed as version 7.5 in all published materials. Mango update introduced back-ground agents along with multi-tasking of third-party apps.

Windows Phone 8 is the second generation of the Windows Phone mobile operating system released on October 29, 2012. It introduced major changes in architecture, new features and many existing ones were improved. Windows Phone 8 is designed to run existing Windows Phone apps unchanged. Unfortunately, current Windows Phone 7 platform devices are not updatable to Windows Phone 8.” [21]

3.3 Platform Architecture

This chapter discusses the major changes that have occurred in Windows Platform Architecture since the previous generation of architecture and the opportunities it provides from a developer’s perspective.

Shared code

“In Windows Phone 8 Windows CE-based architecture is replaced with Windows NT kernel. Moving to a common Windows core meant that every major underlying subsystem had to change. Windows Phone 8 now shares the same file system (NTFS), networking stack, security elements, graphics engine (DirectX), device driver framework and hardware abstraction layer (HAL) as Windows 8. The shared basis of the two platforms means that an application can be ported between these two platforms with much less effort. This change also brought support for multi-core processors.”[21]

CoreCLR engine and garbage collector

“Windows Phone 8 includes the CoreCLR engine previously maintained by .NET Compact Framework. The CoreCLR includes many of the same features and optimizations as the CLR in the .NET Framework 4.5. The CoreCLR includes an auto-tuning garbage collector. These changes resulted in reduced startup time and higher responsiveness in apps."[21]

Async programming model

"Windows Phone 8 introduced the new task-based async programming model across the CoreCLR and the .NET Framework libraries and enabled asynchronous code without much effort. Using the new async and await language keywords, it is now much easier to provide a highly responsive UI experience."[21]

Native code support

"Windows Phone 8 has full C and C++ support, making it easier to write apps for multiple platforms more quickly. It means support for gaming middleware such as Havok Vision Engine as well as native DirectX-based game development."[21]

3.4 Platform Features

This chapter describes the most important features of Windows Phone platform from the developer's point of view. Also, attention to the features that have been either added or improved along with Windows Phone 8 is discussed.

Resolutions and scaling

"Windows Phone 8 supports three different screen resolutions WVGA (800 x 480), WXVGA (1280 x 768), and 720p (1280 x 720). One problem faced by mobile developers is the multitude of different screen resolutions that are available. Therefore, applications must be customized for each resolution. The Windows Phone screen hardware is able to scale the screen of an application to fit whatever screen size the device supports. This makes it possible to create games or applications that will work on any screen size, including ones that have not been made yet. Existing Windows Phone 7 based applications run without changes and scale automatically with crisper text and vector art on the higher-resolution displays. But in order to truly get the best of this feature, developers should use high resolution graphics" [14] [21].

“The phone contains an accelerometer that detects how the phone is being held. The Windows Phone operating system can then adjust the display to match the orientation. A Developer may design an application to work in both landscape and portrait mode or just either one of them.”[21]

Multi-tasking

“Multitasking is a controversial and debatable topic. Applications running in the background often consume resources and drain the battery. Balancing resources and multitasking has been a difficult issue for any mobile operating system. In situations where the application is deactivated, the application is put into a dormant state. The application’s state is completely preserved, but the main thread is paused. However, if the operating system determines that it is running low on RAM, it may take some dormant applications and tombstone them (see below).”[21]

Tombstoning

“It is similar to the hibernate-state on a desktop environment, but it applies for every background-app. When a user changes to other applications, the current app status will save the necessary data to the memory and the application is “put into a grave”. The app will not be running and it will not consume any resources. When the user comes back, it will be loaded from scratch. But it also uses the previous data and the application “returns from the grave”. Developers must handle tombstoning in the code” [15] [21].

Localization

Localization refers making an app to adapt a new market. To localize an app, one has to remove all the hard coded strings and images and instead mark such elements with a unique identifier.
In a data acquisition app for this thesis, a lot of text data is used. When building the app, text is put directly into the XAML and C# code. But when there is a need to translate app to another language, one has to move all the strings to separate files called resource files. A resource file has the extension .resw and is placed in a special directory called Strings inside the project. This folder must contain a subfolder for every

supported language. The subfolders follow the naming convention for the language/region pattern. For example, en-US means English language in the United States, and fi-FI means Finnish language in Finland. This pattern is called the BCP-47 language tag. One can also use just the language to force the runtime to use localized resources for every region. For example, if a folder named en is used, the English language will be used for United States as well as Great Britain (and Australia, and so on). In this subfolder, one creates a file called Resources.resw. This is the new naming convention for Windows Store apps. If there is a .resx file from a .NET project, then that file is copied into the new folder structure and is renamed for the file extension to .resw. The format is compatible because the defined schema is same [16].

Scheduled Tasks and background agents

“Scheduled Tasks and background agents allow an application to execute code in a separated thread even if the application is paused. The application must first register the agent on the Scheduled Action Service. Background agents can only be scheduled to run in two ways:
Periodic Tasks: Periodic agents can run for a very short period of time and perform lightweight tasks on a regular recurring interval. Typical scenarios for this type of a task include uploading the device’s location and performing small amounts of data synchronization.

Resource Intensive Task: Resource-intensive agents can run for a relatively long period of time when the phone meets a set of requirements. A typical scenario for this type of a task is synchronizing large amounts of data to the phone while it is not being actively used by the user.

Windows Phone 8 introduced two new background agents, giving both VOIP and location-based applications the option of working with background services. The VOIP agent handled incoming voice, video and chat sessions, while the background location agent worked with location data.

Microsoft recommends using a mutex for synchronizing access to resources that are shared between the foreground application and the background agent, such as files in isolated storage. A file in isolated storage can be used for one-directional communication, where the foreground app writes and the agent only reads.”[17] [21]

Live Tiles and Live Apps

“Windows Phone uses Live Tiles to display interactive content on the start screen. The user does not need to navigate in and out of apps manually to find out what’s going on. The user can choose which tiles to see and arrange them to any order. Windows Phone 8 adds support for small tile size, previously supporting only medium and large tile size. Live Tiles can now be resized by the user (as shown in Figure 2 below).

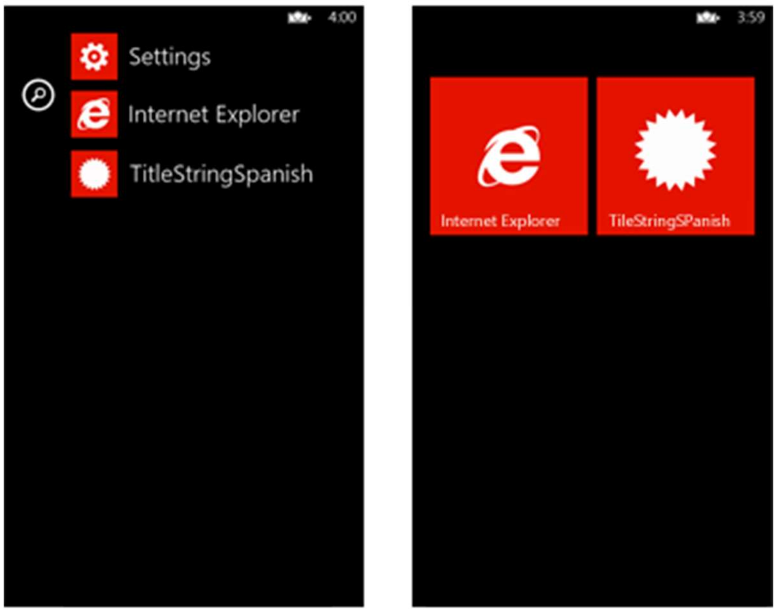


Figure 2.The title in the app list and title on the app Tile pinned to the Start screen.

Windows Phone 8 brought a new feature called Live Apps. It includes Live Apps for many built-in apps such as email, messaging and calendar. Live Apps are basically just versions of Live Tiles for apps to display interactive content (Figure 2). They can be integrated with the lock screen. Third-party developers are able to create Live Apps.”[21]

Lock screen

“Windows Phone 8 lock screen supports Live Apps and Windows 8-like basic and detailed notification options. Third party applications can now register as the lock screen wallpaper provider, and can be included in the lock screen notification area. These notifications can include a 24 x 24 applications icon and an information text which are taken from the app’s tile.”[21]

Camera

“Windows Phone 8 introduced a new Camera API that can be easily used by developers. It offers following functionalities: – Camera parameter configuration: ISO speed and exposure.

- Real-time access to the phone’s video stream.
- Multi-frame capture for creating new types of camera experiences and imagery.
- Custom lenses that integrate with the built-in camera app offering effects, filters and computational photography.”[21]
- Images can be captured using phone feature and can be saved into company’s database on-site.

Windows Phone Maps

“Nokia’s Maps solution replaced Bing Maps in Windows Phone 8. It offers more complete and accurate map data, a new 3D mode, and hardware-accelerated rendering. A new WinPRT -based location API also accompanies the new Nokia Maps control and provides functionality new to the Windows Phone platform such as generating driving directions with an API call so that they can be included in an app where geo location is required. Existing apps that include the Bing Maps control will continue to be in use for business applications.”[21]

4 Application Development Plan

The development of the application consisted of six steps:

1. Creating a REST API Interface for company’s database (managed by the company developers)
2. Creating access points to be implemented (managed by the company developers)
3. Designing a UI appropriate for mobile devices, taking into account platform constraints
4. Developing a Client that uses the REST API
5. Creating user interface for the relevant access points
6. Implementing the access points for the user interface

As a summary, the development included the design and development of a custom mobile Windows Phone application for the company’s extranet. The mobile application used a REST API with end-points to connect to extranet server. The goal was to implement the client-side end points within the API. In the mobile application, all of these assets are on the mobile phone as they are not pulled in from the website. The only thing that is extracted from the website is the data through the REST API.

The aim of this project was to develop an application for e-acquisition of images in the mobile environment. It allowed sharing of information between the company employees and clients in a fast, reliable and efficient manner as shown in Figure 3 below. The application should show how an application would respond to the company’s extranet. The development plan included creating a login page with a username and password unique to a user. Upon login, user would connect to company’s internal data base using a REST Interface. The first page of application would include list of projects and articles related to the company’s projects. It would also show projects assigned to the user or related to the user. Upon selecting a project, it would navigate a user to subpages in the application which includes documents and articles and images related to that project.

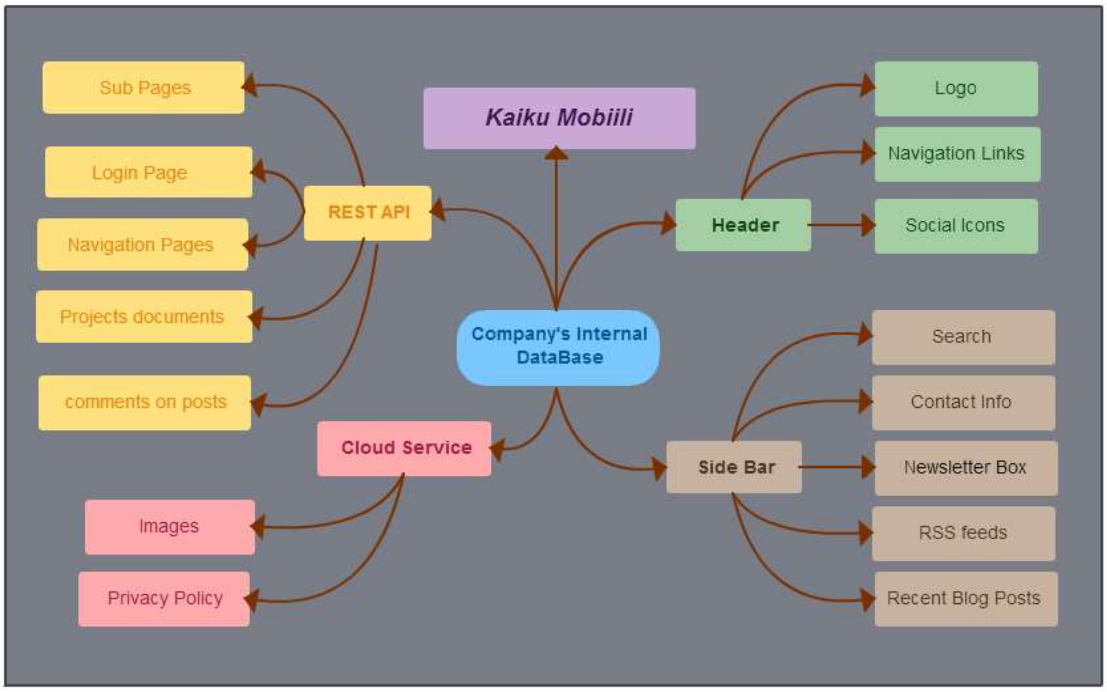


Figure3. Types of Contents (RESOURCES) in the Software Application

The user also had an option to capture an image from his windows phone camera and save it as a file attached to his projects. The company intended to develop the project further to include cloud services for data storage and add security and social media features to application. Social media features included navigation to different social media forums, search engine, company information, news feeds from various internet resources and individual blogs.

4.1 Choosing Implementation Tools and Technology

“The Windows Phone SDK 8.0 provides the tools that are needed to develop applications and games for Windows Phone 8 and Windows Phone 7.1. The most important tools used were:

- Visual Studio Express 2012 edition for Windows Phone
- Windows Phone Developer Registration tool
- Windows Phone Connect tool
- Emulators for Windows Phone 7.1 and 8.0
- Windows Phone Application Analysis tool

- Simulation Dashboard for Windows Phone

Microsoft is slowly deprecating Silverlight and XNA in Windows Phone platform. The SDK still allows creating Silverlight and XNA based apps, but only if the build target is set to Windows Phone 7.1."[21]

Visual Studio Express 2012 edition for Windows Phone

Visual Studio Express 2012 is a revised version of the Integrated Development Environment (IDE) for Visual Studios that provides features specific to Windows Phone development. The Software Development Kit (SDK) provides an add-in tool with other versions of Visual Studios 2012.

Blend for Visual Studio 2012

"Blend for Visual Studio 2012 is a user interface design tool for creating graphical interfaces. One of the key ideas behind Blend is that it allows animators and UI designers to create the interface while developers write the code-behind. Blend offers dynamic flow and elements layout and positioning that are based on relevance to its parent. One can still specify width and height, but in most cases this will be a minimum width and a minimum height value. It handles data bindings giving an accurate depiction of the XAML live in the design-time environment."[21]

Connect phone to development environment

"In order to deploy an application from the development environment directly to a device, it must first be registered using Windows Phone Developer Registration tool. Registration requires the following things:

- Installation of the Zune software.
- A Microsoft account (formerly known as a Windows Live ID).
- A valid and current Windows Phone Dev Center account.

After successful registration, the phone can be connected to the development environment using either the Windows Phone Connect tool or Zune software."[21]

The case company's custom Application Programmable Interface

There are dozens of programming or scripting languages capable of implementing a REST API. If a programming or scripting language can support the key principles of REST mentioned in the Chapter two, it is technically possible to implement the API with that particular language. PHP and Ruby language were selected in this case. The REST API was designed by the case company's hired developers. Microsoft software development platforms and libraries are then extensively utilized for the entire coding process of application

Windows Phone Software Development Toolkit (SDK 8.0)

The Windows Phone SDK 8.0 is a windows development environment for creating applications for Windows Phone 8.0 and Windows Phone 7.5. The Windows Phone SDK provides a stand-alone Visual Studio Express 2012 edition for Windows Phone or works as an add-in to Visual Studio 2012 various editions (Professional, Premium or Ultimate). Developers can use their existing programming skills using SDK and code to build managed applications or applications in native code. SDK also features multiple emulators and testing tools for debugging and testing coded apps for the real application scenario. [18]

C# and .NET Framework

C# is an object-oriented programming language developed by Microsoft as a part of .NET platform. It inherits many features of C, C++, Visual Basic and Java. The most recent version is C# 7.0, which was released in March, 2017. Windows Phone 8 supports the new C# 5.0 language features. Programs in C# run on the .NET framework. For managing the executions of these programs is responsible Common Language Runtime (CLR). CLR allows managing the execution of .NET programs. The CLR is created by Microsoft and provides type safety and memory management. C# code is compiled into an intermediate language (IL), hence the code is stored in a file with an extension .exe or .dll.

Silverlight

Silverlight is a software plugin based on the .NET framework. It brings advanced functions for multimedia and better interaction features. The plugin is compatible with many internet browsers and operating systems. Windows Phone 7 Operating System is compatible with Silverlight 3 and supports Silverlight 4's features and some other performance improvements in storyboard animations and interfacing. [19].

Windows Phone Application Analysis tool

"Windows Phone apps must meet a set of certain performance criteria to be published in the Windows Phone Store. There are certification requirements regarding:

- App launch time.
- App responsiveness.
- Maximum memory usage by the app.

The Windows Phone Application Analysis tool provides monitoring and profiling options to evaluate and improve the quality and performance."[21]. Profiling option in the tool allows one to evaluate either execution-related or memory usage aspects such as:

- Application memory consumption.
- App monitoring generates a detailed analysis page displaying graphs and monitoring warnings [20].

"The app monitoring option helps one to identify problems such as:

- Slow startup time.
- Slow response time to input, such as scrolling or zooming.
- High battery drain.
- Network latency.
- High cost of network data
- Poor performance as the quality of the network signal changes.
- Out of memory errors caused by high resource usage." [21]

One can use the Windows Phone Store Test Kit to identify some of these issues. However the Application Analysis tool helps one to identify, understand, and troubleshoot the source of these issues in the actual application.

4.2 Simulation Dashboard for Windows Phone

“Often a developer tests an application under optimal conditions. The Simulation Dashboard offers simulation options to ensure that the app performs well under unexpected scenarios that might occur in real life. Currently simulation dashboard offers the following settings:

- Network simulation for network speed and signal strength.
- Lock screen simulation.
- Reminder simulation.”[21]

4.3 Test Phase

The test phase of app testing is an essential part of Application Development to make the app work as efficiently as possible. It includes the following phases:

1. Concretizing the app concept before it goes into the design process. Once the design process starts, it is much harder to change things around, so clearer the prototype from the start, the better.
1. Polishing the app to make it work excellently, fixing the bugs in windows phone application development environment through simulation testing using Microsoft Visual Studio
2. Performing unit tests which are automated tests that verify functionality at the component, class, method, or property level. Unit tests are the foundation of automated and regression testing, which provides a long-term stability and a future maintainability of the project.
3. Integration testing which was performed as soon as access to back-end services, web services and APIs were available. Integration testing ensured that the API worked as expected, that all areas of the system communicated with

each other correctly and that there were no gaps in the data flow. The final integration test proved that the system works as an integrated unit when all the fixes were complete.

4. Functional testing assures that each element of the service meets the functional requirements of the business as outlined in the requirements document/functional brief, use cases, system design specification, other functional documents produced and user interface testing. Functional testing also includes device and OS specific testing.
5. Enabling and configuring security for the application that typically requires configuring the login server and testing it on a real device to ensure and make sure that user identity is established each time the application allows data excess to the user.

4.4 Implementation

This chapter introduces examples of programming codes used that utilize some of the most popular and familiar features of mobile devices. The examples were created bearing in mind their purpose as introduction material. They are hands on examples, which were kept user-friendly and easily understandable for a reader. All the programming was done by using programming languages C#, XAML and the Visual Studio 2012 Express for Windows Phone 8 as a development environment.

The KaikuMobiili application was made for Windows phone 7.0/7.5 (Mango) and Windows Phone 8 in Visual studio Express 2012. At first, radio buttons were created for the user interface and to interact with other elements in the application. The source code was written in C# language and XAML. This client application uses the company’s custom designed REST architecture and JSON API. For de-serializing, The JSON objects used were Json.NET framework. For a better user experience, there are also control elements from Windows phone toolkit in use.

The application utilizes windows phone camera features and implements the CameraCaptureTask class for taking pictures. CameraCaptureTask class handles all camera related work and uses the built-in camera software. When a user taps on windows phone camera icon, a request is sent from the user to start the application and would wait for the user to capture an image.

4.5 Basic HTTP Authentication

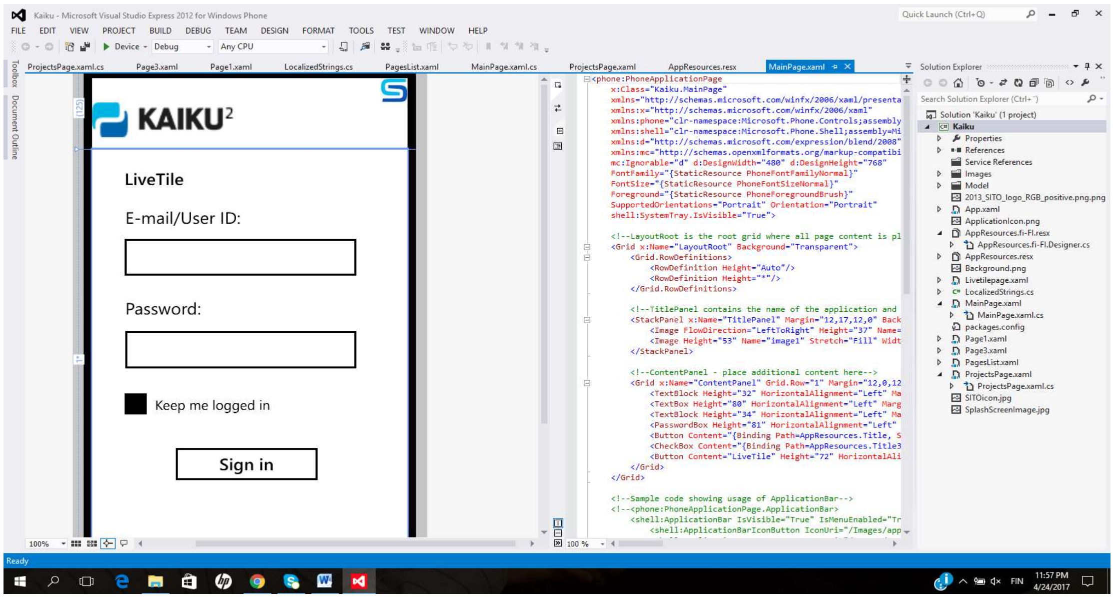
Authentication is the verification of credentials of the connection attempt that is part of the login process. This process consists of sending the credentials from the remote access client to the remote access server in an either plaintext or encrypted form by using an authentication protocol. Here is an example of adding basic HTTP access authorization to a REST endpoint that was used in the implementation. In the "configure" method, one can see that the application shows a basic auth module and passing in the "realm" (part of the basic authentication standard) and the path to the Apache-style credential file.

Later on in the code, one uses the "onAuthenticate" point in the endpoint request lifecycle. This method is called between the "preProcess" and the "handleRequest" point cuts. One thing to note is that in the REST endpoint, one can mark REST methods with rest:authenticate and use that annotation in our "onAuthenticate" method to determine if authentication is needed. Some REST APIs have public methods without authentication and some methods with authentication. One can also define rest:authenticate="true" on thecfcomponentto to indicate a global default which is available by callinggetAutenticationDefault().

The REST endpoint will programmatically call onAuthenticate() if one or more of the following is true:

- The REST method has rest:authenticate="true" metadata on the method definition.
- The REST CFC has rest:authenticate="true" metadata on the CFC. This is a global flag and sets that all REST methods must be authenticated unless otherwise stated differently on the CFC REST method directly.

Firstly, a user interface is created with Radio buttons that allows users to select one option from two or more choices of buttons. Each option is represented by one radio button as shown in Figure 4. To select multiple options, a checkbox or a list box control is added.



```
<phone:PhoneApplicationPage
    x:Class="Kaiku.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presenta
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Mi
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibi
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is pl
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and
        <StackPanel x:Name="TitlePanel" Margin="12,17,12,0" Back
            <Image FlowDirection="LeftToRight" Height="37" Name=
            <Image Height="53" Name="image1" Stretch="Fill" Widt
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12
            <TextBlock Height="32" HorizontalAlignment="Left" Ma
            <TextBox Height="80" HorizontalAlignment="Left" Marg
            <TextBlock Height="34" HorizontalAlignment="Left" Ma
            <PasswordBox Height="81" HorizontalAlignment="Left"
            <Button Content="{Binding Path=AppResources.Title, S
            <CheckBox Content="{Binding Path=AppResources.Title3
            <Button Content="LiveTile" Height="72" HorizontalAli
        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="Tr
            <shell:ApplicationBarIconButton IconUri="/Images/app
```

Figure 4. UI Kaiku Mobilli User Login Page showing Radio Buttons.

In the end, the basic HTTP access authentication module takes care of checking the headers, decoding the auth, checking the credential file and ultimately setting other header if the authentication request fails. This module returns a Boolean so one can customize the exception output if the authentication fails. Getting back to the code below if the authentication fails, an exception is thrown out. By default, the base REST endpoint handles all exceptions except in this case one defined special logic in the onException() point-cut to provide the correct data back to the client. The application authorization coding is shown below in Listing 1 as follows:

```
using System;
using System.IO;
using System.Net;
using System.Linq;
using System.Windows;
using Microsoft.Phone.Controls;
using System.Device.Location;
using System.Text.RegularExpressions;
using System.Runtime.Serialization.Json;
using System.Json;
using System.IO.IsolatedStorage;
using Microsoft.Phone.Shell;

namespace Kaiku
{
    public partial class MainPage : PhoneApplicationPage
    {
        String strUserToken;
        string strUsername;
        string strPasswd;
        static string strAcceptHeader = "application/json";

        //Ugly Code: Change this to use resource file or a  UI setting
        string strUrl = "https://kaikutest.sito.fi";

        // Constructor
        GeoCoordinateWatcher watcher;
        public MainPage()
        {
            strUserToken = "";
            InitializeComponent();
            // The watcher variable was previously declared as type GeoCoordi-
nateWatcher.
            if (watcher == null)
            {
                watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High); //
Use high accuracy.
                watcher.MovementThreshold = 20; // Use MovementThreshold to ig-
nore noise in the signal.
                watcher.StatusChanged += new
EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
            }
            watcher.Start();

        }

        private void radioButton1_Checked(object sender, RoutedEventArgs e)
        {

        }
        void watcher_StatusChanged(object sender, GeoPositionStatusChangedEven-
tArgs e)
        {
            if (e.Status == GeoPositionStatus.Ready)
            {
                // Use the Position property of the GeoCoordinateWatcher object
to get the current location.
                GeoCoordinate co = watcher.Position.Location;
                //textBox1.Text =
co.Latitude.ToString("0.000")+co.Longitude.ToString("0.000");
                //textBox1.Text = ;
                //Stop the Location Service to conserve battery power.
                watcher.Stop();
            }
        }


        private void btnLogin_Click(object sender, RoutedEventArgs e)
        {
            /*
             * UGLY FIX: Replace these static creds after testing.
             * */
            strUsername = txtUserName.Text;
            strPasswd = txtPassword.Password;
            strUsername = HttpUtility.UrlEncode("aurangzeb.lodhi@sito.fi");
            strPasswd = HttpUtility.UrlEncode("cnn420AA111");

            string strAuthEndPoint =
"/api/authentication/token_exchange?user_name="+strUsername+"&password="+strPassw
d;
```

```
            // Get token
            HttpWebRequest reqAuthToken = (HttpWebRe-
quest)WebRequest.Create(strUrl + strAuthEndPoint);
            reqAuthToken.Accept = strAcceptHeader;
            reqAuthToken.BeginGetResponse(new Async-
Callback(AuthResponseReceivedCallBack), reqAuthToken);

        }



        void AuthResponseReceivedCallBack(IAsyncResult result)
        {
            HttpWebRequest request = result.AsyncState as HttpWebRequest;
            if (request != null && result.IsCompleted)
            {
                try
                {
                    HttpWebResponse response = (HttpWebRe-
sponse)request.EndGetResponse(result);
                    if (response.StatusCode == HttpStatusCode.OK)
                    {

                        using (var reader = new StreamRead-
er(response.GetResponseStream()))
                        {
                            String respData = reader.ReadToEnd();
                            Match match = Regex.Match(respData,
@"authentication_token\""\:\""([A-Za-z0-9]+)\""\,");
                            if (match.Success)
                            {
                                strUserToken = match.Groups[1].Value;

                                /*
                                 * TODO
                                 * 1. Use Isolated Storage AppSettings to save
strUserToken and strUserName.
                                 *    We need these values in other pages.
                                 * 2. Use Navigation Service to Navigate to the
ProjectsPage.
                                 * */

                                IsolatedStorageSettings settings = IsolatedStor-
ageSettings.ApplicationSettings;
                                try
                                {
                                    settings.Add("username", strUsername);
                                    settings.Add("token", strUserToken);

                                }
                                catch (Exception ex) {
                                    settings["username"] = strUsername;
                                    settings["token"] = strUserToken;
                                }
                                settings.Save();
                                Dispatcher.BeginInvoke(() =>
                                {
                                    NavigationService.Navigate(new
Uri("/ProjectsPage.xaml", UriKind.Relative));
                                });
```

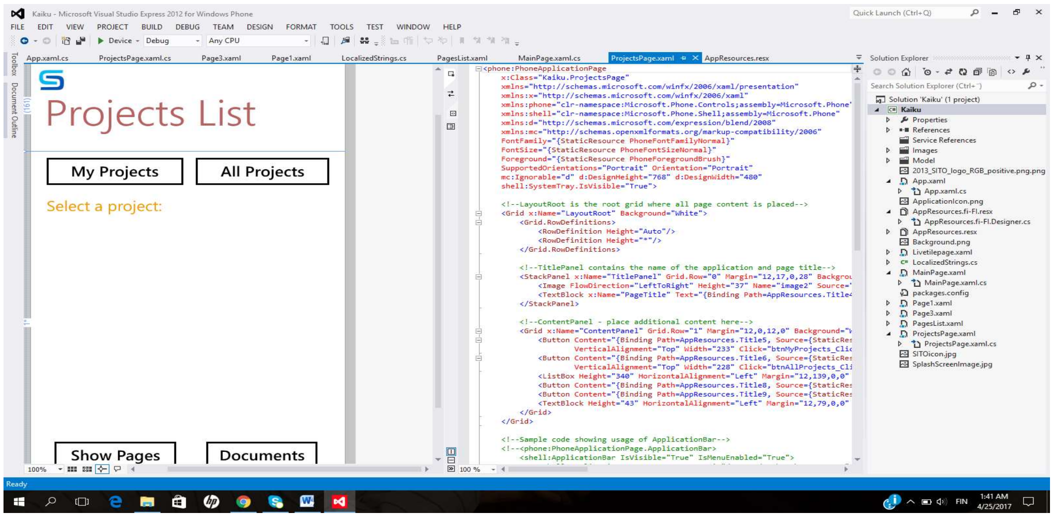
```
                              }
                              else
                                  Dispatcher.BeginInvoke(() => Message-
Box.Show("Failed to get user token"));
                           }
                       }

                   }
                   catch (WebException ex)
                   {
                       if (ex.Response.Headers["Status"] == "403")
                           Dispatcher.BeginInvoke(() => MessageBox.Show("Login 
Failed"));
                       else
                           Dispatcher.BeginInvoke(() => Message-
Box.Show(ex.Response.ToString()));
                   }

               }
           }

           private void checkBox1_Checked(object sender, RoutedEventArgs e)
           {

           }

           private void button1_Click(object sender, RoutedEventArgs e)
             {

           }


           private void button1_Click_1(object sender, RoutedEventArgs e)
           {
               CreateTile();
           }

           private void CreateTile()
           {
               /**
                * http://msdn.microsoft.com/en-
us/library/windowsphone/develop/hh202979(v=vs.105).aspx
                */
               // Look to see whether the Tile already exists and if so, don't try 
to create it again.
               // need to use the reference Microsoft.Phone.Shell and System.Linq
               ShellTile TileToFind = ShellTile.ActiveTiles.FirstOrDefault(x => 
x.NavigationUri.ToString().Contains("pinned=MyImage.jpg"));

               // Create the Tile if we didn't find that it already exists.
               if (TileToFind == null)
               {

                   // Create the Tile object and set some initial properties for the 
Tile.
                   // A Count value of 0 indicates that the Count should not be dis-
played.
                   StandardTileData NewTileData = new StandardTileData
                   {
                       BackgroundImage = new
Uri("http://d1045bjs8msx9s.cloudfront.net/assets/work/kaiku-
96ea8c0dd1551fe88380324908de582e.png", UriKind.Absolute),
                       //Title = "  SITO OY",
                       //BackTitle = "KAIKU MOBIILI",
                       BackBackgroundImage = new
Uri("http://sito.fi/media/sito_logo_large1.jpg", UriKind.Absolute),
                       Count = 0
                   };

                   // Create the Tile and pin it to Start. This will cause a naviga-
tion to Start and a deactivation of our application.
                   ShellTile.Create(new Uri("/MainPage.xaml?pinned=MyImage.jpg",
UriKind.Relative), NewTileData);
               }
               else
               {
                   System.Diagnostics.Debug.WriteLine("--- A tile is already exist-
ed.---");
                   MessageBox.Show("A tile already exists!");
               }
           }

           private void txtUserName_TextChanged(object sender, Sys-
tem.Windows.Controls.TextChangedEventArgs e)
           {

           }

       }
}
```

Listing 1.Step by Step coding for Authentication

Once the authentication process is completed, the projects list is loaded through the API. A user can then tap or click on the project list or list of own projects to view pages related to it.

Get Projects List from Company’s Extranet

After pressing the login button and following the authentication, the application sends a web request to get the user data for projects. When projects are downloaded, three more requests are sent: get pages, documents and articles as shown in Figure 5. As soon as the user authentication process is completed and the user is logged in, he/she is able to see all the current projects found on company’s data base as well as the projects assigned to the user. The user can then select a particular project from the app and is able to see the relevant pages, documents and articles related to the project.



```
<phone:PhoneApplicationPage
    x:Class="Kaiku.ProjectsPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28" Backgrou
            <Image FlowDirection="LeftToRight" Height="37" Name="image2" Source="
            <TextBlock x:Name="PageTitle" Text="{Binding Path=AppResources.Title4
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0" Background="W
            <Button Content="{Binding Path=AppResources.Title5, Source={StaticRes
                    VerticalAlignment="Top" Width="233" Click="btnMyProjects_Clic
            <Button Content="{Binding Path=AppResources.Title6, Source={StaticRes
                    VerticalAlignment="Top" Width="228" Click="btnAllProjects_Cli
            <ListBox Height="340" HorizontalAlignment="Left" Margin="12,139,0,0"
            <Button Content="{Binding Path=AppResources.Title8, Source={StaticRes
            <Button Content="{Binding Path=AppResources.Title9, Source={StaticRes
            <TextBlock Height="43" HorizontalAlignment="Left" Margin="12,79,0,0"
        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
```

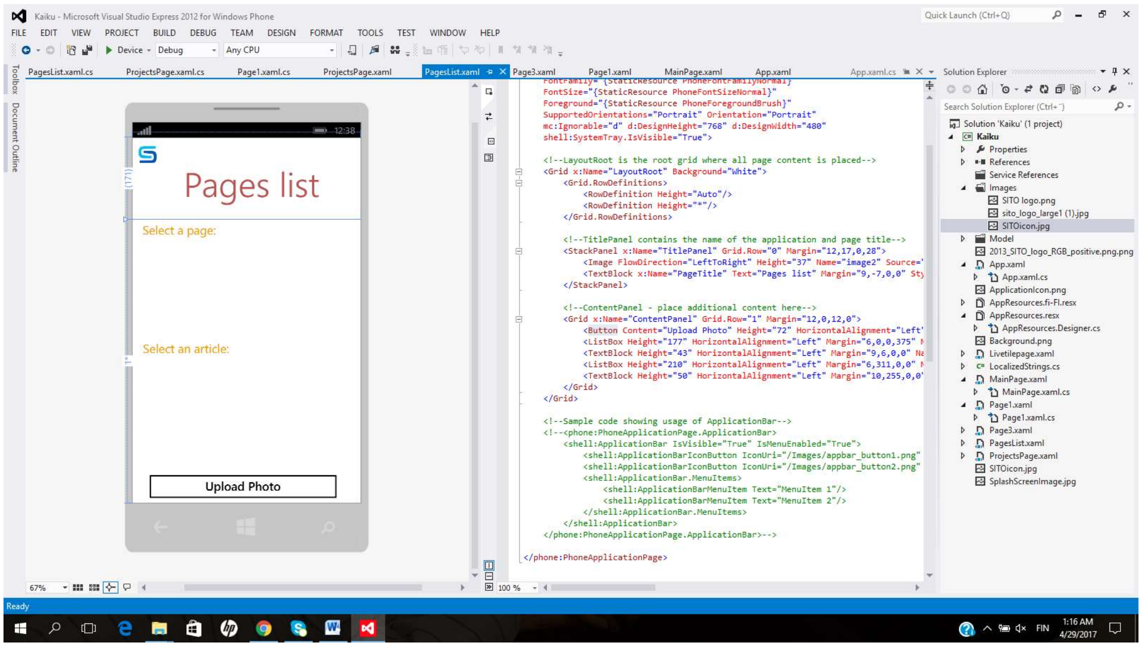
Figure 5. Get Projects Page that displays all projects as well as user's own projects.

Coding to get the projects involves getting projects data from company's API once the authentication process is completed. The list of projects related to 'my projects' or 'all projects' would be loaded in the projects list box as shown in the Listing 2 below:

```
namespace Kaiku
{
    public partial class ProjectsPage : PhoneApplicationPage
    {
        String strUserToken;
        string strUsername;
        static string strAcceptHeader = "application/json";

        //Ugly Code: Change this to use resource file or a UI setting
        string strUrl = "https://kaikutest.sito.fi";

        // Constructor GeoCoordinateWatcher watcher;

        public ProjectsPage()
        {
            InitializeComponent();
            IsolatedStorageSettings settings = IsolatedStorageSet-
tings.ApplicationSettings;
            strUsername = settings["username"].ToString();
            strUserToken = settings["token"].ToString();
        }


        private void GetProjects(bool all = false)
        {
            string strGetProjects = "/api/projects?user_name=" + strUsername +
"&user_token=" + strUserToken;
            if(all)
                strGetProjects += "&all=true";
            // Get list of projects for this user
            HttpWebRequest reqGetProjects = (HttpWebRe-
quest)WebRequest.Create(strUrl + strGetProjects);
            reqGetProjects.Accept = strAcceptHeader;

            reqGetProjects.BeginGetResponse(new Async-
Callback(GetProjectsRespReceivedCallback), reqGetProjects);
        }

        void GetProjectsRespReceivedCallback(IAsyncResult result)
        {
            HttpWebRequest request = result.AsyncState as HttpWebRequest;
            if (request != null && result.IsCompleted)
            {
                try
                {
                    HttpWebResponse response = (HttpWebRe-
sponse)request.EndGetResponse(result);
                    if (response.StatusCode == HttpStatusCode.OK)
                    {

                        DataContractJsonSerializer serializer = new DataCon-
tractJsonSerializer(typeof(Projects));
                        Projects returnedProjects = (Pro-
jects)serializer.ReadObject(response.GetResponseStream());
                        Dispatcher.BeginInvoke(() => lstProjects.ItemsSource =
returnedProjects.projects);
                    }

                }
                catch (WebException ex)
                {
                    if (ex.Response.Headers["Status"] == "403")
                        Dispatcher.BeginInvoke(() => MessageBox.Show("Login
Failed"));
                    else
                        Dispatcher.BeginInvoke(() => Message-
Box.Show(ex.Response.ToString()));
                }

            }

        }

        private void btnMyProjects_Click(object sender, RoutedEventArgs e)
        {
            GetProjects();
        }

        private void btnAllProjects_Click(object sender, RoutedEventArgs e)
        {
            GetProjects(true);
        }

        private void btnShowPages_Click(object sender, RoutedEventArgs e)
        {
            // TODO Navigate to ArticlesPage
            // In the articles pages button click, load the articles from API and
show in list box.
            // ERR handle if no itemks are selected
            Project selectedProject = (Project)lstProjects.SelectedItem;
            // pass the selected Project ID as a parameter to the PagesPage.

            NavigationService.Navigate(new
Uri("/PagesList.xaml?id="+selectedProject.id+"&name="+HttpUtility.UrlEncode(selec
tedProject.name), UriKind.Relative));
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {

        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {

        }

        private void image2_ImageFailed(object sender, ExceptionRoutedEventArgs
e)
        {

        }

        private void lstProjects_SelectionChanged(object sender, Selection-
ChangedEventArgs e)
        {

        }
        }

    }
```

Listing 2. Step by Step coding to Load Project Pages.

Once a relevant page is chosen related to a particular project, the user interface provides an option to select a page and article/articles related to the page as shown in Figure 6 and uses the windows phone camera feature to capture an image/photo, upload it and attach it to an article. The UI provides a smooth navigation from one page to another. It is possible to jump to the previous page by using the back button on the phone.

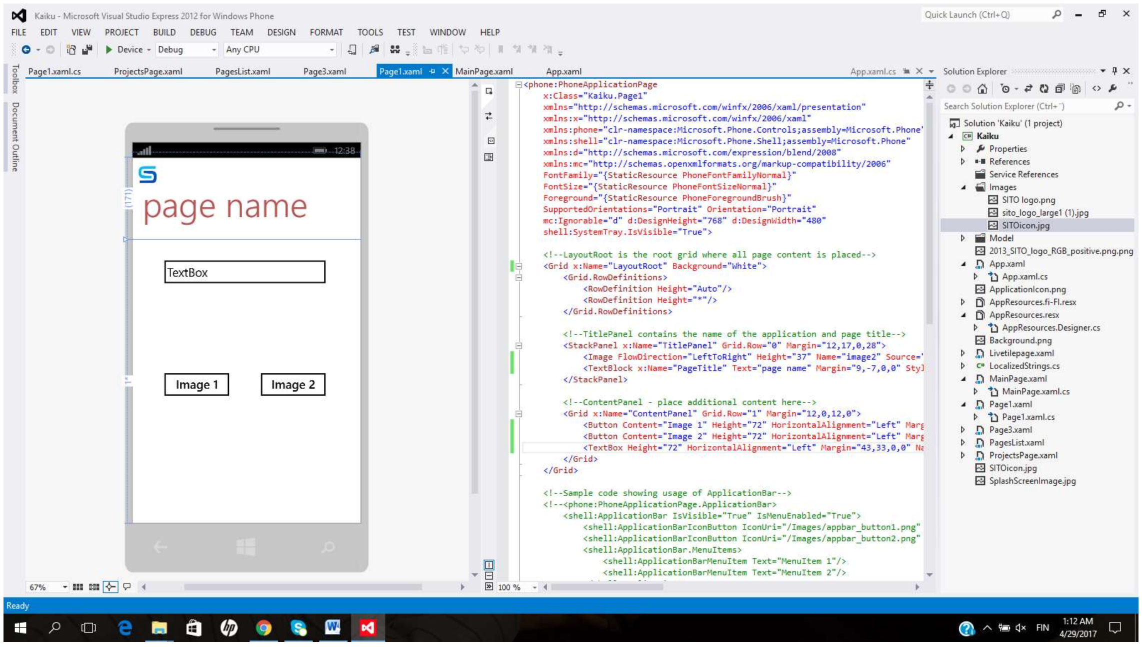


```
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <Image FlowDirection="LeftToRight" Height="37" Name="image2" Source='
            <TextBlock x:Name="PageTitle" Text="Pages list" Margin="9,-7,0,0" Sty
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Button Content="Upload Photo" Height="72" HorizontalAlignment="Left'
            <ListBox Height="177" HorizontalAlignment="Left" Margin="6,0,0,375" N
            <TextBlock Height="43" HorizontalAlignment="Left" Margin="9,6,0,0" Na
            <ListBox Height="210" HorizontalAlignment="Left" Margin="6,311,0,0" N
            <TextBlock Height="50" HorizontalAlignment="Left" Margin="10,255,0,0'
        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png"
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png"
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
                <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>-->

</phone:PhoneApplicationPage>
```

Figure 6. Pages List and Articles related to a project.

Get Images from Company's Database

The user has an option to search for a project in the text box and then click on the image box one or two to view up the pictures related to the project. It means that two images from the project can be viewed at the same time, as shown in Figure 7.



```
<phone:PhoneApplicationPage
    x:Class="Kaiku.Page1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone'
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <Image FlowDirection="LeftToRight" Height="37" Name="image2" Source='
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0" Styl
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Button Content="Image 1" Height="72" HorizontalAlignment="Left" Marg
            <Button Content="Image 2" Height="72" HorizontalAlignment="Left" Marg
            <TextBox Height="72" HorizontalAlignment="Left" Margin="43,33,0,0" Na
        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png"
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png"
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
                <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
```

Figure 7. Get images related to a project search.

Images are displayed in a separate image box in the application and the image resolution depends on the size of image.

5 Testing and Results

Windows Phone Store Test Kit provides tests which help to determine whether or not an app will pass the store certification. The tests are categorized according to the way they run: automatically or manually. Automated tests evaluate the basic criteria of the app. The manual tests require navigating through the app and observing its behaviour in several different conditions, in order to ensure it meets the app certification requirements.

5.1 Unit Testing

In unit testing, individual units of source code for all radio buttons were tested in the windows phone emulator for Visual Studios to see if they performed individual tasks correctly and met the purpose. Unit testing found problems associated with individual buttons early in the development cycle. Each unit of 'Kaiku Mobiili' application was tested by running the emulator on Visual Studios and looked for any errors related to the program.

Errors were highlighted automatically in the coding and they were corrected unit by unit. This included both bugs in the programs implementation and flaws or missing parts of the specification for the unit. The cost of finding a bug before the coding begun or when the code was first written was considerably lower than the cost of detecting, identifying, and correcting the bug later. It would have been difficult to test a poorly written code at a later stage in the testing. Thus, unit testing helped the application to structure functions and objects in better ways.

5.2 Integration Testing

Integration testing was a logical extension of unit testing. Two units from the application that had already been tested were combined into a component and the interface between them was tested by running the windows phone emulator. A component, in this sense, referred to an integrated aggregate of more than one unit. Many units were therefore combined into components, which are in turn aggregated into even larger parts of the program. The idea was to test combinations of pieces and eventually expand the process to test other buttons of the app. All the individual buttons making up a process were tested together.

Integration testing identified problems that occurred when units were combined. By using a test plan that required one to test each unit and ensure the viability of each before combining units, any errors discovered when combining units were likely related to the interface between units. This method of testing reduced the number of possibilities and made the analysis easier.

For all input modules that had been unit tested, the integrated testing phase grouped them in larger aggregates, applied the tests defined in an integration test plan to those aggregates, and delivered the app for system testing [26]. Integration testing verified the functional, performance, and reliability requirements placed on major design items. There was a continuous integration with Visual Studio Application life cycle management to ensure that whenever the code was developed and checked in, it worked correctly with the existing code [41]. After integration testing, the app was delivered to simulation testing and performance testing.

5.3 Simulation Testing

Simulation testing was performed to simulate a slow network speed, a weak signal, or the absence of a network connection while running Kaiku Mobiili app on the emulator and also on an actual windows phone to test how the app handled these conditions.
Simulation Dashboard was used in Visual Studio to test the app for these connection problems, and to prevent users from encountering scenarios such as the following:

- Calls to a web service fail with a timeout.
- The app crashes when no network is available.
- Data transfer does not resume when the network connection is lost and then restored.
- The user’s battery is drained by streaming app that uses the network inefficiently.

Network Simulation that was available in the Simulation Dashboard in Visual Studio for network simulation was enabled to simulate a 3G or 4G connection. To simulate the behavior of mobile network signal, good, average and poor signal strength options were chosen on the simulation dashboard.

5.4 Application Performance Testing

When running the app in Windows Phone Emulator, frame rate counters were used to monitor and test the performance of Kaiku mobiili app. Frame rate counters were first enabled in the application code which tested the following:

- The rate at which the screen is updated.
- The rate at which the UI thread is running.
- The number of pixels per frame used on the screen.
- The video memory and system memory copies of textures being used in the app.

Images in JPG and PNG image formats were used in the app to improve the performance of the app. JPG worked well for continuous tone images, including photographs. However, it did cause ringing and blocking artefacts in images with varying colours and gradients. Hence the logo of the company was added in PNG format to keep up with varying colour pattern. With the ‘app monitoring’ option, one could evaluate and test the most important behaviours of mobile app that contributed to a good user experience, such as start time and responsiveness, maximum memory usage by the app and the approximate rate of battery consumption while the app was running.

5.5 Results

The test scenarios with their respective test steps, expected results and actual results are shown in Table 1 below:

Table 1.Results of Testing

Scenario	Test Step	Expected Result	Test Result
Login request	Provide coding values to the login page using Radio	Login should be successful only for the registered users	Login page test passed and user was authenticated

	buttons	of the application	for logging in to the application
Responsive layout	Run the application on different Windows phone devices and the windows phone emulator.	Display should be responsive and show all the required fields.	Application worked fine with all screen resolutions.
Projects list Page	Provide coding values to the get projects list button and run it on windows emulator.	Projects list should be loaded from the company's database.	List of projects is displayed in a separate list box in the application.
Documents and Articles List Page	Test and display all the values through a simulation on actual windows phone.	Document and Articles related to a particular project should be loaded.	List of Documents and Articles related to it were displayed in a separate list box.
Tombstoning the app	Testing storage options on windows phone that puts application in the background to be retrieved later.	App is not dumped out of a memory or exited upon pressing start button from a phone. It does a last-moment save of state to avoid a re-launch from scratch.	When the user relaunches the application (or returns to it via the Back button), applications state was reloaded.
Localization of UI and a content that is culture and language specific	Application recourse .resx containing name and value of resources in Finnish language in a string format.	Validation of all application resources and verification of linguistic accuracy and resource attributes.	Upon launch, application contents in English language are localized and displayed in Finnish language.

Navigation between Application pages	Navigation class method to navigate to a specific page and come back to a specific page using a Windows phone back button.	The Back button on a phone automatically navigates backward without any special handling.	Smooth navigation between user interface pages with no delay or crash.

The results were achieved from the first version of a Kaiku Mobiili API service that provided an API solution on Windows platform. The work on the application still continues, and with cooperation with professional developers in the company, the application will be further developed.

The project provides room to expand it to other platforms such as the Android and iOS platform. The prototype forms a basis to all other future developments with the application.

6 Conclusion

The subject of this thesis came into existence after the case company decided to develop a mobile application customised for efficient use in Windows Phone Platform. This was a perfect opportunity for the author to familiarize himself with the platform. Despite the fact Windows Phone being the youngest mobile platform, it provides an excellent development environment within the SDK. Visual Studio is maybe the best IDE and forms the basis for many IDE's currently available and Blend introduces a unique way of creating a user interface. One could argue that C# combined with .NET libraries offer easy first steps into programming. As proved in Chapter four discussing the implementation, it takes only a few lines of code to do rather impressive things such as controlling the camera. Moreover, the SDK offers ways for testing and quality assurance that have never been seen before. At the time of working on this thesis, the Windows Phone 10 SDK was just about to get released. After the actual release, there were no published books available, only online documentation. Together with the time restriction, this was maybe the biggest challenge in the process. The new "customized" user interface offered a unique option for a sub-contractor of the company. It can be praised for simplicity as well. Windows Phone runs surprisingly well on the cheapest hardware allowed. One reason for this is the tight conditions set for applications. Unfortunately, this comes with a price; when an application gets paused, there are only a few limited options left for the developer.

The battle between the different mobile platforms is fierce. Since all employees and sub-contractors of the case company were given a Windows phone to access the company data, development of applications on Windows Phone 7 and 8 was a success story.

The case company plans to further develop the Kaiku Mobiili Application by using additional features such as cloud services and also to expand to other mobile platforms such as the Android and the iOS. Future work could focus on the areas that need to be developed further so the mobile application would be more efficient for the case company. Such features, amongst others include graphic and user interface improvements.

A native mobile application can be made in future across all platforms to enable users to use the application easily and more effectively. Improvements can also be made to data backup so it would work properly and on daily basis to make sure that the user will not lose any data in case of system crash.

References

[1] Application Program Interface by Vangie Beal.
http://www.webopedia.com/TERM/A/API.html

[2] IBM Developer Works, API Connect
https://developer.ibm.com/apiconnect/documentation/api-101/types-apis/
Accessed 23.05.2016

[3] Patterns of Knowledge in API Reference by Maalej, Waleed, Robillard, Martin P. Accessed 01.04.2012

[4] Internet-Based Application Programming Interfaces, API's by David Berlind.
www.programmableweb.com Accessed 01.05.2014

[5] Building Flexible API's for web 2.X/Cloud Applications by Raymond Feng.
https://www.slideshare.net/rfeng/building-flexible-apis-for-web-2xcloud-applications-javaone-2011-session-25208 Accessed 05.04.2011

[6] Learn Rest: A Restful Tutorial by Todd Fredrich.
www.restapitutorial.com/lessons/whatisrest.html Accessed 30.05.2012

[7] Architectural Styles and the design of Network based Software Architectures by Roy Thomas Fielding.
https://www.ics.uci.edu/~fielding/pubs/dissertation/net_arch_styles.htm#sec_3_4_1
Accessed 03.01.2000

[8] Introducing JSON.
http://www.json.org/ Accessed 06.10.2013

[9] Copter Labs by Jason Lengstorf
https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/
Accessed 05.06.2009

[10] XML Syntax and Parsing concepts by Kenneth.B.Sall.
http://www.informit.com/articles/article.aspx?p=27006&seqNum=3
Accessed 31.05.2002

[11] Microsoft Developer Network Documentation.
https://msdn.microsoft.com/en-us/library/cc295302.aspx Accessed 03.05.2016

[12] Blinks Guide to metro.
http://blinkux.com/metro/ Accessed 10.01.2014

[13] Introduction to Windows Mobile 7 Development by Nick Ohrn.
https://code.tutsplus.com/tutorials/introduction-to-windows-mobile-7-development--mobile-91 Accessed 19.05.2010

[14] Windows Phone 7 Feature focus: Hardware by Paul Thurrott.
http://winsupersite.com/article/windows-7/windows-phone-7-feature-focus-hardware
Accessed 01.11.2010

[15] Windows Phone 7 tombstoning explained by Justin James.
http://www.techrepublic.com/blog/smartphones/windows-phone-7-tombstoning-explained/ Accessed 23.05.2011

[16] How to build a Localized app for Windows 8 by Microsoft Corporation.
https://msdn.microsoft.com/en-us/library/windows/apps/ff637520(v=vs.105).aspx
Accessed 01.07.2016

[17] Background agents for Windows Phone 8, Microsoft Tutorials.
https://msdn.microsoft.com/en-us/library/windows/apps/hh202942(v=vs.105).aspx
Accessed 03.04.2015

[18] Windows Phone SDK 8.0 by Microsoft.
https://www.microsoft.com/en-GB/download/details.aspx?id=35471
Accessed 05.03.2016

[19] About Silverlight by Microsoft.
https://www.microsoft.com/silverlight/what-is-silverlight/ Accessed 10.07.2014

[20] Windows Phone Application Performance Analysis by Dhananjay Kumar
http://www.c-sharpcorner.com/uploadfile/dhananjaycoder/windows-phone-application-performance-analysis/ Accessed 17.09.2011

[21] Introduction to Windows Phone 8 by Pahkala, Jan
http://www.theseus.fi/bitstream/handle/10024/51260/Pahkala_Jan.pdf?sequence=1
Accessed 2012