Final Thesis

Responsive design in Windows 8 applications

by

Sofi Klockare

LIU-IDA/LITH-EX-A--13/036--SE

2013-08-02

Supervisor:  Anders Fröberg, IDA, Linköping university
            Fredrik Eriksson, Infor

Examiner:    Erik Berglund, IDA, Linköping university
Abstract

Responsive design is a common expression at the web today. This thesis was about learning about responsive web design in order to apply it to Windows 8 applications created in C# and XAML. The purpose of this thesis was to learn if responsive design can be used in such applications. This was done by first learning about responsive design through a literature study about responsive web design and then specifying responsive application design. How responsive design can be used in applications was found through research and creation of an example application. No literature was found about responsive design for Windows 8 applications in C# and XAML and therefore most of the results came from own ideas and partly solutions found on different forums. The thesis was performed at Infor in Kista. The example application was a social application which was connected to a social web service developed by Infor.

Responsive web design is a technique used for making webpages adapt to the size and resolution of the viewport. This is done by creating a structured, fluid layout by using a flexible grid and flexible images. Media queries are used to set up different states for presenting the page in different ways for different sizes and resolutions of the viewport. In the different states parts of the content can for example be hidden or styled. The number of columns of information can also be set to adapt to make the page easier to view.

Responsive design for applications should make the layout of the application adapt to fit the application window of the device used depending on its resolution. One way to specify a responsive design for Windows 8 applications could be to create only one application which could be used on all different Windows 8 platforms, such as phones, tablets and computers. It is probably not possible today to create this one application for different Windows 8 platforms. Another way of specifying responsive design for application is to create a responsive
user interface which could be shared and used in the different Windows 8 applications. The latter was the chosen definition for this thesis.

The scope of this thesis was limited to only research for responsive design in Windows Store apps and Windows Phone 8 applications. Windows Phone 8 applications are applications used on smartphones with Windows Phone 8 as operating system while Windows Store applications are used on computers with Windows 8 as operating system. Other Windows 8 devices were not included in this thesis.

The final conclusion of this thesis was that although it is complicated to use responsive application design it can be used for most of the user interface. A responsive user interface for the applications can be created separately, but some parts of the application pages cannot be shared between the different platforms. This results in a mostly responsive sharable user interface.
Acknowledgements

This essay is the result of my master thesis at the program Master in Computer Science and Engineering at Linköping University. The study was performed at Infor in Kista, Stockholm.

I would like to thank Erik Berglund, my examiner and Anders Fröberg my supervisor at Linköping University. I would also like to thank my supervisor at Infor, Fredrik Eriksson.

A special thanks to Peter Karlsson and Karin Portillo who have helped me when I got stuck in the programming of the applications.

Special thanks also to my father, mother and my aunt for helping me improve this thesis.

At last I want to thank my family and my partner for believing in me, I love you all.

Thank you!

Norrtälje, June 2013
Sofi Klockare
# Table of Contents

GLOSSARY .......................................................................................................................... 1

I.  INTRODUCTION .............................................................................................................. 3

1.  INTRODUCTION ............................................................................................................ 3

   1.1  BACKGROUND OF THE STUDY ................................................................. 4
   1.2  INTENT ................................................................................................................. 4
         1.2.1  Problem description .............................................................................. 5
   1.3  SCOPE AND LIMITATIONS .......................................................................... 5
   1.4  DISPOSITION OF THE THESIS ................................................................. 6
         1.4.1  Typographical convention ...................................................................... 6

II.  METHOD .......................................................................................................................... 7

2.  METHOD ......................................................................................................................... 7

   2.1  PLANNING .............................................................................................................. 7
   2.2  LITERATURE STUDY ......................................................................................... 8
   2.3  RESEARCH ............................................................................................................ 9
   2.4  DEVELOPING THE APPLICATIONS ............................................................... 9
   2.5  ANALYSIS ............................................................................................................ 11

III.  BACKGROUND .............................................................................................................. 13

3.  BACKGROUND .............................................................................................................. 13

   3.1  WINDOWS 8 ....................................................................................................... 13
       3.1.1  Developing for Windows 8 .................................................................... 15
   3.2  INFOR SOCIAL SPACE ...................................................................................... 16

IV.  LITERATURE STUDY .................................................................................................... 19

4.  RESPONSIVE WEB DESIGN .......................................................................................... 19

   4.1  FLUID DESIGN .................................................................................................... 19
   4.2  WHAT IS RESPONSIVE WEB DESIGN .............................................................. 20
   4.3  THE INFINITE GRID .......................................................................................... 21
   4.4  COMPONENTS OF RESPONSIVE WEB DESIGN ............................................ 23
       4.4.1  Flexible Grid ............................................................................................ 23
       4.4.2  Flexible images ......................................................................................... 31
       4.4.3  Media queries ............................................................................................ 32
V. RESULT

5. RESEARCH ABOUT RESPONSIVE APPLICATION DESIGN ........................................... 37

5.1 WHAT IS RESPONSIVE APPLICATION DESIGN? ......................................................... 37
5.2 SHARING CODE BETWEEN WINDOWS 8 AND WINDOWS PHONE 8 ...................... 39
  5.2.1 Using sharable elements in XAML layout ......................................................... 40
5.3 TESTING FOR RESPONSIVENESS ......................................................................... 42
5.4 COMPONENTS IN RESPONSIVE APPLICATION DESIGN .................................. 45
  5.4.1 Flexible Grid ........................................................................................................ 45
  5.4.2 XAML Images ...................................................................................................... 49
  5.4.3 Media queries ...................................................................................................... 55
5.5 CREATING A RESPONSIVE USER INTERFACE ..................................................... 58
  5.5.1 Making elements responsive ............................................................................. 59

6. THE INFOR SOCIAL SPACE APPLICATION ............................................................. 63

VI. DISCUSSION ........................................................................................................... 71

7. DISCUSSION ............................................................................................................. 71

  7.1 ANALYSIS OF RESULT .......................................................................................... 73
  7.2 DISCUSSION ABOUT THE WORK ........................................................................ 76
  7.3 RECOMMENDATIONS FOR FUTURE STUDIES .................................................. 78

VII. CONCLUSION ......................................................................................................... 81

8. CONCLUSION ............................................................................................................. 81

REFERENCES ............................................................................................................... 83

  PUBLISHED SOURCES ............................................................................................... 83
  UNPUBLISHED SOURCES ........................................................................................... 83

APPENDIX 1 .................................................................................................................. 87

1. SHARING TECHNIQUES ........................................................................................... 87

  1.1 MODEL-VIEW-VIEWMODEL ................................................................................. 87
  1.2 PORTABLE CLASS LIBRARIES ........................................................................... 88
  1.3 ADD AS LINK .......................................................................................................... 88
  1.4 WINDOWS RUNTIME COMPONENT ..................................................................... 89
  1.5 XAML USER INTERFACE ...................................................................................... 89
    1.5.1 Why Microsoft doesn’t recommend sharing User Interface ................. 89
  1.6 CONDITIONAL COMPIILATION .......................................................................... 90
1.7  PARTIAL CLASSES .................................................................90
1.8  INHERITANCE .................................................................90
1.9  INTERFACES .................................................................91
Table of Figures

FIGURE 1: Screenshot of start menu in Windows 8 .................................................. 14
FIGURE 2: Grid components ....................................................................................... 25
FIGURE 3: The page Responsive Grid System uses a flexible grid-based layout ........... 26
FIGURE 4: The page Responsive Grid System in a slimmer viewport .......................... 27
FIGURE 5: The layout of www.responsivegridsystem.com in correspondence to a visual grid ..................................................................................................................... 28
FIGURE 6: The layout of responsivegridsystem.com in correspondence to a visual grid with content in three and four columns ............................................................... 29
FIGURE 7: A grid example* ......................................................................................... 31
FIGURE 8: responsivegridsystem.com in slim viewport ........................................... 36
FIGURE 9: Image of simulator for running Windows Store apps ............................... 43
FIGURE 10: Image of resolutions of Windows Phone 8 ............................................ 44
FIGURE 11: Alignment grid on emulator .................................................................... 46
FIGURE 12: The feed page shown on a Windows Phone 8 emulator in landscape mode. ................................................................................................................... 64
FIGURE 13: Feed page shown on Windows Phone 8 emulator in portrait mode ...... 64
FIGURE 14: Feed page for the Windows Store application shown in simulator with smaller resolution in portrait mode ................................................................. 66
FIGURE 15: Feed page for the Windows Store application shown in simulator with higher resolution in portrait mode ................................................................. 67
FIGURE 16: Feed page and colleague page shown at Windows 8 simulator in landscape mode .............................................................. .................................................. 69
FIGURE 17: Several parts of the page are visible on a wide screen without scrolling. ................................................................................................................... 70

Table of Tables

TABLE 1: Description of features which can be used in media queries ...................... 34
TABLE 2: Sharable controls ......................................................................................... 41
Glossary
The following terms will be used in this thesis:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>A program tailored for a specific operating system that performs some task for the user. In this thesis apps are tailored for either Windows Phone 8 or Windows 8.</td>
</tr>
<tr>
<td>Application</td>
<td>See app</td>
</tr>
<tr>
<td>C#</td>
<td>Programming language</td>
</tr>
<tr>
<td>Code behind</td>
<td>In this thesis it’s the C# code which together with the connected XAML document handles the user interface of the Windows 8 application.</td>
</tr>
<tr>
<td>CSS</td>
<td>CSS stands for Cascading Style Sheet and is style sheet language used to attach styles to documents. CSS2.1 is level 2, revision 1 of CSS.</td>
</tr>
<tr>
<td>Device</td>
<td>Refers to a specific device, it can be a smartphone, a computer or tablets.</td>
</tr>
<tr>
<td>Device type</td>
<td>Refers to the different types of devices. Smartphones, computers and tablets are three different device types. It does not refer to the different types of smartphones, computers or tablets; these are instead referred to as different devices.</td>
</tr>
<tr>
<td>Display area</td>
<td>The part of the window or screen that shows the web page or application.</td>
</tr>
<tr>
<td>Fiddler</td>
<td>A web debugging proxy which can be used to track web requests.</td>
</tr>
<tr>
<td>Flexible grid</td>
<td>A fluid visual image which shows a grid used to structure content. See chapter 4.4.1 Flexible Grid.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HTML</td>
<td>A programming language for describing web pages.</td>
</tr>
<tr>
<td>Infor Social Space</td>
<td>A social network for the workplace developed by Infor. See chapter 3.2 <em>Infor Social Space</em></td>
</tr>
<tr>
<td>Media queries</td>
<td>Queries that can be used in cascading style sheets while designing webpages to learn about the device and browser showing the webpage. See chapter 4.4.3 <em>Media queries</em>.</td>
</tr>
<tr>
<td>Phone</td>
<td>In this thesis a phone refers to a smartphone</td>
</tr>
<tr>
<td>Platform</td>
<td>See device type</td>
</tr>
<tr>
<td>User interface</td>
<td>In this thesis the user interface is the pages of an application, which presents the content to the user.</td>
</tr>
<tr>
<td>Viewport</td>
<td>See display area</td>
</tr>
<tr>
<td>Windows Phone 8 application</td>
<td>An application (see app) for smartphones that uses Windows Phone 8 as operating system.</td>
</tr>
<tr>
<td>Windows Store application</td>
<td>An application (see app) for computers using Windows 8 as operating system.</td>
</tr>
<tr>
<td>XAML</td>
<td>Programming language used in Windows 8 to set up the layout</td>
</tr>
</tbody>
</table>
I. Introduction

1. Introduction

Not that many years ago, the only way to access the web was with a desktop computer. Since then technology has advanced a lot and fast. Today we can visit webpages from laptops, smartphones, tablets and TVs as well. Soon webpages might be seen on the inside of glasses or at the windows of buildings. It is impossible to design everything for the future when the future is changing constantly with new technology popping up from around the corner. Some future proof design for the nearer future is preferable and might be possible. Computers, tablet and smartphones all access webpages through a browser and presents it on the screen. One of the biggest differences between these devise is the amount of screen available to present the webpage. A webpage that looks good on a computer screen might not look good on a small phone or a huge TV.

When designing for the web developers use some different methods in order to create a better experience for the visitors of a homepage. One usual approach to handle phone visitors is to add a link on the first page, which when clicked forwards the user to a website specially designed for smartphones. Another solution that now has been around since 2010 is to use responsive web design. Ethan Marcotte, who coined the expression, found a solution for designing the webpage so that it would adapt to the size of the viewport. This is done by moving contents around and formatting the page to look good and make the content readable on the screen whether the visitor watches it from home at the TV or on the bus via a smartphone. [2] It is possible that webpages designed with this technique can also be presented in a good way on the windows of a building.

The idea of creating one page to fit “all” viewing devices is interesting for applications as well. Last year Microsoft released the operating
system Windows 8 for both phones and tablets in addition to the earlier computer version. For Windows 8, applications can be created which makes it possible to continue the work that was done on the computer on the tablet or phone. Although the user can use the applications from any device to do a task the applications are created individually. What if responsive design could be used within applications to create just one sharable application or at least to create a responsive sharable user interface?

1.1 Background of the study
This report is the result of my master thesis at the program Master in Computer Science and Engineering at Linköping University. The study was performed at Infor in Kista, Stockholm.

With the rapid development of new technological devices which can be used to visit the web through browsers and applications it is important to design applications and webpages so that all or as many as possible of the visitors can use and get a good user experience when using them. Since responsive web design has become such a successful approach for creating a common user interface for webpages being presented on different devices, it raises a curiosity to find out if it can also be used for creating a sharable application or a common user interface for applications.

1.2 Intent
The goal with this thesis was to discover if responsive design can be used when developing Windows 8 applications. This work involved learning about responsive web design to understand the concepts of responsive design. Research for how responsive design can be used was performed. An example application was created in order to test the found solutions, to discover difficulties and to come up with new solutions or workarounds. The application that was developed is a social application that is connected to the system Infor Social Space.
1.2.1 Problem description

The result for this thesis is an answer to the question:

- Can responsive design be used when developing Windows 8 applications in C# and XAML?

The question above must be examined. In order to understand responsive design, responsive web design was studied. Responsive web design might not be exactly the same as responsive application design. The study did therefore also include a discussion on what responsive application design could intend before answering if it can be done. The questions that needed to be answered were the following:

- What is responsive web design?
- What is or what could responsive design be in Windows 8?
- Can responsive design be used in Windows 8 applications?

1.3 Scope and limitations

This study had a time limit of 20 weeks which lead to some limitations in order to finish on time. The first limitation is that only Windows Phone 8 and Windows Store applications were researched. Other Windows 8 applications, such as applications for tablets, fell out of scope. Another demarcation is that only applications created in C# and XAML was covered in this thesis. Responsive design was only investigated as a part of web design and application design. Other fields exist were responsive design is applicable, but these fell out of scope.

There was not enough time to implement the entire application. Only some parts of the application were implemented to test the research of responsive design in applications.

It was not enough time to find all possible solutions to responsive application design. The focus of this thesis, responsive application design, is a new area which made it difficult to find solutions since it was hard to find documentation. The last demarcation was that all possibilities were not explored.
1.4 Disposition of the thesis

This thesis is organized into seven parts.

**Introduction:** The first chapter of the thesis is the Introduction. In this chapter the intent and background of the study is presented. There is also a problem description and information about the scope of the project to present the focus and limitations of this thesis.

**Method:** The second chapter presents approach taken during this thesis.

**Background:** The background covers basic information about Windows 8 and Infor Social Space.

**Literature study:** The literature study covers the area of responsive web design. The chapter covers what responsive web design is and what the basic components are.

**Result:** The result chapter presents the research about responsive application design. This involves what responsive application design is and how it can be used for Windows 8 applications. This chapter also presents the developed social applications.

**Discussion:** Covers a summary and analysis of the results of this thesis and a discussion about the work behind the thesis.

**Conclusion:** Presents the conclusions from this thesis.

1.4.1 Typographical convention

In this thesis references to sources is numbered square brackets (e.g [1]), linked to the list of reference in the end of the thesis.

In the chapter research a lot of XAML controls are mentioned. These controls are written with italic font and a capital as first letter (e.g TextBlock). Values for the properties are written in italic font.

Pieces of code are presented in text boxes and are color coded. Code snippets within the paragraphs uses italic font.
II. Method

2. Method

During this thesis several steps have been completed in order to find the answer to the problem description. During the first weeks of the project a lot of time was spent on planning the work and learning about Infor Social Space and Windows 8. After this a literature study of responsive web design was conducted and the development of the applications was started. Around half time the research and testing of responsive application design techniques was started.

2.1 Planning

In the beginning of this thesis the goal with the project was to develop a social application for Windows Phone 8. The application was going to be an extension to the company Infors web service, Infor Social Space. During the first weeks of the project, different focuses were examined for this essay. Much time was spent on planning the work with the thesis and the focus was decided upon. Small studies were also conducted to learn about Windows 8, Windows Phone 8, user experience and to learn about Infor Social Space. To find information about Windows 8 and Windows Phone 8 a smaller search where performed on the web to search for basic knowledge about the subjects. To learn about Infor Social Space, the existing web system was explored by using it.

During the user experience study, some sketches were created to see what this application could look like. These sketches where produced by following Microsoft’s design process “How great apps come alive” [23]. Since both usability and conducting a study in social applications where excluded as focuses for this thesis, the work done regarding these subjects were removed from this thesis.

During this first phase, I and the examiner and mentors of this thesis discussed several possibilities of focuses for this project. The plan was
adjusted several times to suit the project before a focus of responsive design was decided. The concept was to conduct a literature study about responsive web design in order to learn what it is, what it is used for and how to create responsive web pages. This knowledge would later be used to learn how to adapt responsive web design to create responsive application design for Windows 8 applications created in C# and XAML. In addition to the study the assignment was to also use this knowledge to create a responsive application (or two separately) for Windows 8 and Windows Phone 8 to test the findings. To finish the thesis an analysis of the results found would be conducted.

2.2 Literature study
The literature study was performed by searching for material both on the web and also in one book written by Ethan Marcotte[2], the person who coined the expression “responsive web design”. There are several websites with information about this very popular web phenomenon. There are also some books available both on the web and in printed form.

Ethan Marcotte has also written articles about the subject at the webpage alistapart.com where other designers also have made some informative articles on the subject. This page has been one important source of information regarding responsive web design.

Since Ethan Marcotte is the founder of responsive web design, his book and articles were the foundation of the study. Other resources were also used to verify and complement the foundation.

During the literature study, a search for responsive application design was performed. The results closest to the sought results were findings of responsive design in web applications created in html and CSS3. No results were found about responsive design for other applications than web applications for Windows 8.
2.3 Research

The research was performed in parallel with the development of the applications. This was important because the applications were used to test the found solutions to make sure they were valid. The components of responsive web design were used as a steppingstone to search for similar components in C# and XAML code for Windows Phone 8 and Windows Store applications. Knowledge about responsive web design was also used to find requirements which in turn were used to find different possibilities for responsive C# and XAML approaches.

To know what to search for, the intent with responsive application design for Windows 8 applications was defined. This led to an extra search for sharing code between Windows Store apps and Windows Phone applications.

There was no full solution to find about responsive application design. But smaller possible techniques were tested to check if they could be used to form some part of the responsive design and together form a solution to the problem. Many of the possible solutions were discarded because they didn’t work for either Windows Phone applications or Windows Store applications. Own ideas were also explored by changing properties of different XAML controls, creating bindings and writing code in the C# files attached to the XAML application pages. This was done to examine how the properties and controls worked and to test ideas for solutions. Many smaller applications were created in order to test smaller modules alone. Many possibilities were discovered by trying and searching for solutions only when obstacles occurred. The solutions for how to work around the obstacles were found on the forum StackOverflow.com, on the forums and development pages on msdn.microsoft.com [21][14] and also sometimes by using the Microsoft support and asking coworkers at Infor.

2.4 Developing the applications

In the developing area several test applications were developed. Some of the test applications were created to control how different settings
for sizes of objects in XAML affected the fluidness of the page. Others were created in order to learn how code could be shared between Windows Store applications and Windows Phone 8 applications.

Much information was fetched from Microsoft’s development pages for Windows Phone 8 and Windows 8 [21][14]. These pages contains application programming interfaces, example applications and many guides to assist when developing applications.

The developments of the social applications were also started. The first goal was to write the code that manages to fetch and post data from and to Infor Social Space. To start of the work with the applications some code, from another application connected to Infor Social Space, was used. This code was a good basis, but much of the code couldn’t be used when coding a Windows Phone and/or Windows Store application.

The code was debugged on Windows Phone emulators and a Windows 8 simulator and the program Fiddler was used to control the messages sent from the application. A lot of time was spent on searching for correct instructions and trying for getting the emulator to run with Fiddler. Time was also spent on learning how to save the state of the emulators. This was needed because emulators are reset when restarted and the applications needed certificates which needed to be installed on the emulator.

There are a lot of differences between Windows Phone 8 applications and Windows Store apps. During the entire developing phase this has resulted in much testing and searching for answers.

In the end the project had to be cut down to cover only a smaller part of the application in order to have enough time to create a more responsive user interface where more code could be shared between the Windows Store and the Windows Phone 8 application.
2.5 Analysis

At the end of the project the results of the research were summarized and an analysis of the results was performed. The analysis contains difficulties with the different approaches. Own thoughts and experiences were gathered in order to analyze if and in what cases, responsive application design can be useful. A discussion about the work behind the thesis was also conducted by analyzing experience from the entire project.
III. Background

3. Background
This thesis is about responsive design in Windows 8 applications, developed in C# and XAML. In order to understand this subject it is important to know about responsive design and to know some basics about Windows 8. In this thesis the information about responsive design is located under the Literature study. One part of the thesis was to develop a responsive Windows 8 application. This application is connected to a system called Infor Social Space. Windows 8 and Infor Social Space will be explained in this background chapter. The sources to the Windows 8 chapter is the Swedish article *Mobils guide till Windows Phone 8* [8] and a webcast titled *Webcast: Windows Phone 8 Launch* [20].

3.1 Windows 8
Windows 8 is Microsoft’s latest operating system and is available for computers, tablets and smartphones. Microsoft have spent a lot of time making this operating system personal and individualized. Windows 8 have many differences towards its predecessor Windows 7. One of the key differences is that Windows 8 is developed to be used with both mouse and keyboard and touch.

In the computer-version the most visible change in the system is the transformation of the start menu. Earlier the menu was accessed from a start button in the left corner of the screen and the menu was composed of a list of program names. In Windows 8 the start menu has been transformed to be a different screen with program panels, called tiles or live tiles, placed in a grid system. The Live Tiles that forms the start page can be moved and changed in size in accordance with the users wishes. The following image shows the start menu of Windows 8 for computers.
Windows Phone 8 was launched at the end of October in 2012. The Windows Phone 8 menu is also comprised of tiles, which for example can be connected to contacts on the phone, in order to always be updated and receive the latest information from contacts and their Twitter, Facebook and LinkedIn updates.

This is not that much different from its predecessors Windows Phone 7 and Windows Phone 7.5, which also have a menu made up from tiles.

In Windows 8 both “regular” programs and Windows tailored programs called Windows Store apps can be used. Microsoft has spent time on expanding the number of applications available in Windows 8, since this have been one of the shortages in Windows phone 7 and 7.5. One part of the solution is that Microsoft has expanded the support with more developing tools and also added backing for more programming languages than its predecessors. These tools and different language support can be useful when developing responsive applications.

In Windows 8 you can more easily sync data between different devices by using the service SkyDrive. It gives the users the possibility to continue their work on any Windows 8 device no matter of which device.
the work was last saved. This is one of the advantages of “SkyDrive”, which maintains the files that can be reached from all Windows 8 devices. Since sharing of files is possible the users might want to be able to continue working with their files. This can apply to applications needing to work for several platforms. Here a common user interface could be used which is adapted according to the size of the screen by using responsive design.

There is much more new functionality in both Windows 8 for computers and the other devices. For more information about Windows Phone 8 you can read the sources to this background information [8] [20] and articles from Microsoft such as the Swedish articles Upptäck Windows 8 [19]

3.1.1 Developing for Windows 8
During this thesis a lot of information will be presented about developing Windows Phone 8 and Windows Store applications. Windows Phone 8 applications are used on smartphones with Windows Phone 8 as operating system while Windows Store applications are used on computers with Windows 8 as operating system. When developing for Windows 8 there are some ground principles that must be thought off.

Usually when designing Windows client applications that can be used in desktop mode, WPF\(^1\) is used to compose the applications. In Windows Store and Windows Phone 8 applications WPF cannot be used and the developer is limited to use Windows Phone components for Windows Phone 8 and Windows XAML components for Windows Store applications. The Windows Store applications are available at the Windows Store and are only run on Windows 8. Traditional desktop applications run in desktop mode in distinction to Windows Store applications and can also be used on devices running Windows 7. [9]

\(^1\) WPF stands for Windows Presentation Foundation. According to Wikipedia WPF is a computer-software graphical subsystem for rendering user interfaces in Windows-based applications.
When creating a Windows Phone 8 or Windows Store application, the developer should in addition to creating the application spend time on navigation, lock screen integration and create a live tile for the application. If the application is for Windows Store the application must also support snapped view.² [12]

The pages that are created in the applications are specified using a XAML document with an attached C# file (in this thesis referred to as the code behind). In the XAML document the basic layout is set up by using different XAML controls and it is initialized from the code behind. The layout can also be altered from the code behind.

When creating a responsive user interface for applications, which should work in both Windows Store and Windows Phone 8 applications, the controls used in the XAML document must exist as both a Windows Phone component and as a Windows XAML component in order to be applicable for both devices. To share the functionality of the application is easier and there are several possibilities of sharing C#-code between Windows Phone 8 and Windows Store applications. More information about sharing code can be found in chapter 5.2 Sharing code between Windows 8 and Windows Phone 8 and Appendix 1.

### 3.2 Infor Social Space

Infor is developing a website for internal social interactions within the workplace. The applications that are created in this thesis is the beginning of an addition to this system for devices running Windows 8. The information presented in this chapter has been found by using and exploring the system itself.

As a first step in developing the applications the functionality in the web version are checked out, in order to obtain requirements and desires for

² When an application is in snapped view it occupies a narrow region of the screen leaving the rest of the screen available for other content such as another application.
the application. The base functionality of Infor Social Space web page (2012-11-21) is the following:

**Profile pages**
Where the user can add a picture, contact information, personal information and personal settings.

”Feed page”
Shows the latest feed for the users profile such as posts from colleagues and groups.

**Colleagues**
Within Infor Social Space you can add colleagues that you want to have contact with. Colleagues post are shown at the users feed page.

**Groups**
Users can request to join groups, such as for example teams or departments. There are public groups where everyone at the company can read and comment posts that are shown in the groups news feed. There are also private groups, where nonmembers only can see basic information about the group. Users can also create groups of their own.

**Search function**
There is a search bar visible on all pages where the user can search for colleagues, posts and messages.

**Messages**
Personal messages can be sent to one or more persons. Users do not need to be colleagues in order to send each other messages. Received messages are found in the inbox. In messages you can add a heading and attach one file.

**Posts and sharing**
Posts and sharing are the same thing and can be shown for colleagues, in a group or be public. One attachment can be added to the post.
This system is not finished, and all functionality is not available during the time of this thesis. Therefore the applications developed during this project do not contain all these functionalities. The mobile phone also has weaknesses and strengths which will make the final application different from the web version. Many of the weaknesses of Infor Social Space will probably be solved in the future.
IV. Literature study

4. Responsive web design

In the beginning of this thesis a literature study in responsive design was conducted. The study has given knowledge about the field in order to learn how to implement and analyze how responsive design can be used in applications for different Windows 8 devices. Responsive design is used in several different areas such as architecture, software development, training techniques and so forth. This study has been delimited to responsive web design. The reason for this is that when this study was made, there were no literature about responsive design for applications that were not web applications created in html5 and CSS3. Since responsive web design was the area closest to application development this was the focus area chosen for the study. Within responsive web design there are a lot of articles, blog posts and also some printed books which can give great information why and how responsive design is used at the web.

4.1 Fluid design

Web designers and developers have for many years used style sheets to design webpages to adapt to fit screens. This is done by using percentage in the style sheets to set up page sizes, margins and to allow the reader to set the font size in her own browser. This way of creating an adaptable design is in this thesis referred to as fluid design. Fluid design scales the webpage to fit the viewport which made it readable for the visitor. This was when the size of the screen didn’t differ as much as it does today. Fluid design is the first stepping stone towards a responsive design. [5] [10]

For a lot of years only computers were used to access web pages. All these computer screens were big enough to fit most webpages. For the last years some of the screens that show the web pages have become
larger while others have become much smaller than most of the older web pages were adapted for. Today web pages can be reached from mobile devices, which additionally can be used in landscape and portrait mode. The webpage can also be presented on a huge TV screen.

To just scale the webpage to fit the small screen of a mobile phone does not create a great experience for the user. By just using percentage, the webpage would be a mess with objects and texts on top of each other and menus that take up the entire screen.

The basis for responsive design is to use a fluid design. But more than just fluid design is needed to ensure an acceptable user experience for viewports of different sizes.

4.2 What is responsive web design

In order to make the web page readable and look good on all devices, responsive design is needed. Before responsive web design, individual webpages could be created for all different viewport sizes and resolutions. Lately a lot of web developers have created an extra webpage just for visitors using a mobile phone. The problem is that the number of different screen sizes and resolutions used, keeps growing and it is or soon will be too many to handle. Instead responsive design is used. Responsive design is a way of adapting the web page to fit the screen it’s being viewed on, without needing to individually handle all different sizes and devices.

Responsive web design consists of a flexible, grid-based layout, flexible images (and media) and media queries. Flexible webpages is done by using percentage to make the webpage adapt to the size of the screen (see chapter 4.1 Fluid design and 4.4 Components of responsive web design). The second subset, flexible images, can be created using the max-width property of images and is needed to make images follow the flexible layout. (see chapter 4.4 Components of responsive web design).
The last component of responsive web design is media queries, which are queries that check features of the viewport and applies specific style sheet decisions to the viewers that match the query. [2]

Responsive design implies learning about the device and using this knowledge to conform the page to fit that device. There is a lot of literature about responsive design for web pages and this information was used to learn about responsive design for applications. Usually responsive web design means that the web page shall adapt to fit the screen.

The infinite grid is a grid which makes the web page adapt even to extreme sizes. This grid can be created using responsive web design and by following a technique presented in the article “The Infinite Grid” [6]. It gives some good points on how to think when designing webpages. The following chapter will explain the concepts of the infinite grid.

4.3 The infinite grid

To start of this section, is a quote by Chris Armstrong, which summarizes the purpose of the infinite grid [6]

*The goal is to make the most of the space available, relative to your content, to maximize readability and presentation.*

*Quote: Chris Armstrong [6]*

Grid systems have long been used together with fixed size canvases when designing applications. To create an application that will suit a lot of different screen sizes, we cannot use fixed canvas sizes and grid sizes. According to Chris Armstrong [6] we are designing for an infinite canvas and therefore need to develop an infinite grid. The problem with infinite canvases is that when we should set up the grid we cannot use the canvas to define it, since the digital canvas dimensions are unknown. Instead Chris Armstrong purposes to define the grid relative to the content. Which he defines as following:
“Designing from the content out means finding a constant in your content—be it the ideal measure of a paragraph or the dimensions of an ad unit—and building your grid out from there to fill the space available.”

The infinite grid does not just stretch to fit screen sizes and content. It also adapts by rearranging the components, by adapting the number of columns and sizes of them and rearranging content such as moving shifting menu from vertical to horizontal at smaller screens. It might also be favorable to reduce the number of detail and maybe also hide some information on smaller screens. On smartphones and tablets the menus might have to change in order to make them easier to click. In the infinite grid this would mean adopting the menu according to screen sizes that might be tablets and smartphone. This is not optimal. If we have a large tablet we still want to be able to click the menus with our large fingers. The easiest solution for this problem is to always make the menus in a size which is easy to click. Device detections can be used to try to figure out what sort of device is used, and customize the application regarding to this information. From own experience where a tablet is often forwarded to the mobile-site of a homepage, I do not trust device detection completely. The webpages looks really strange when they are being stretched to fit a tablet in landscape mode when it is designed to fit the screen of a smaller smartphone or if a larger phone is used than what the page is designed for. In these cases designing for the size of the screen would have been better.

According to Chris Armstrong the infinite grid has different states, in which components are adapted in different ways. The components can be rearranged, resized or hidden in order to make the page look good on the current screen. These states can be set by using media queries. [6]

In *The Infinite Grid* [6], Chris Armstrong defines the following guidelines for designing an infinite grid system:

1. Use proportional units.
2. Start with the extremes, and then work out the in-betweens. With this he means that we should start by designing how to best present the content for the smallest and largest states\(^3\) independently.

3. Change state where relationships break down. This involves deciding at which screen dimensions a transition to another state should occur.

4. Go beyond the extremes. Which implies to design for even smaller screens than a phone or larger than a computer (maybe the page is shown on a wide TV).

These guidelines are good to keep in mind when designing a responsive web page. One of the advantages with responsive web design is to be able to design the page to fit all viewports and by following these guidelines the risk for overlooking some screen size is reduced.

4.4 Components of responsive web design
This chapter will cover the components needed for creating a responsive web design; a flexible grid, flexible images and media queries.

4.4.1 Flexible Grid
The word grid has several different meanings. If you search in Wikipedia you will receive many different hits such as music groups or human genes. In the area of designing applications we have a functional grid, and a visual design grid. A functional grid can be used when coding applications in C# and XAML to place information in a sort of table. This grid is only visible in code (if no borders are added). The design grid is a visual image of a table used to design where information should be placed in a web page or an application to make it structured and aligned. Functional grids can be used to implement the design of applications, but to create the design and to make sure that the design is fulfilled, this

---

\(^3\) The Infinite grid will have different states (for the different screen sizes) in which the components are adapted in different ways. In for example the smallest state (which probably suits most smartphones) the content might only be shown in one column, while 4 or more columns might be used in the largest state (for example large screens).
visual grid is of better use. In web design it’s the visual grid that is referred when the word grid is used.

A grid is much like a table with columns and rows, but in many designs only the columns are relevant. A module is the intersection of one row and one column. More columns leads to more flexibility but might also be more difficult to work with. Between the columns and rows there is a gutter which is used to create some air between information. Information can take up one or several modules. A group of modules is called a spatial zone. [3] [4] [7]

Figure 2 is an image to give a better understanding of the grid basics.
The grid in the picture Figure 2 has both columns and rows, but often the developer only care about the columns, since the rows adapts to fit the content while the columns adapts to fit the viewport. Figure 3 is an example of a page which uses a flexible grid-based layout.

---

Figure 3: The page Responsive Grid System uses a flexible grid-based layout

The image, Figure 3, is a screenshot of the page “Responsive Grid System”, www.responsivegridsystem.com [24]. At the top there are four columns with information, and when the page width is reduced, so is the width of the columns. A screenshot of the webpage with slimmer viewport are shown in Figure 4.

---

5 Miller, Graham. Responsive Grid System [24]
Figure 4: The page Responsive Grid System in a slimmer viewport

Although the page responsivegridsystem.com is in fact responsive, the examples in Figure 3 and Figure 4 are in the same state and has therefor the same basic layout adapted only with the fluidness of the flexible grid-based layout. If the viewport is shrunk further, the page will eventually change state and the number of columns is reduced to one to better fit smaller devices such as smartphones.

A regular grid is a layout scheme with fixed sized components, where the sizes are described in pixels. A flexible or a fluid grid is a grid where the sizes of the components can be described in percentage. The first component in responsive web design is therefore the fluid layout from chapter 4.1 *Fluid design*, combined with a grid for better structure. [10]
The flexible grid is, as mentioned earlier, a visual aid to structure the components and information on a page. Figure 5 shows a homepage where the grid is visible to show how the page follows the visual grid.

![Responsive Grid System](image)

**Figure 5**: The layout of www.responsivegridsystem.com in correspondence to a visual grid

In Figure 5, the structure of the content columns nicely follows the lines of the visual grid. At this webpage, there are four standard columns which each uses three of the grid columns. But there are also a heading that uses more columns and further down on the page only three content columns which each uses four of the 12 columns. The content columns can be defined in width as the amount of grid columns it uses. With a twelve column grid several different setups can be used within the same page. Figure 6 shows how both three and four columns are used within the page responsivegridsystem.com.

---

6 Image based on screenshot of Miller, Graham. *Responsive Grid System* [24]
The grid columns can be used to align the text and images within a content column and it can be used differently within the same webpage. Figure 6 is a good example of how the 12 grid columns can be used to structure the components of the page.

Ethan Marcotte [2] [10] describes how to develop webpages to conform to the flexible grid. He uses percentage too set up the webpages different content parts, fonts, margins and paddings. First the elements can be designed with a size in pixels so that they look good, then the size is recalculated to percentages so it will scale when the browser scales. The following is the equation for calculating the percentage sizes in correspondence to the pixel sizes.

\[
\text{Percentage} = \left(\frac{\text{Pixel Size}}{\text{Maximum Pixel Size}}\right) \times 100
\]

---

7 Image based on screenshot of Miller, Graham [24]
\[
\text{Element size in percent} = \frac{\text{Element standard size in pixels}}{\text{Containers standard size in pixels}}
\]

By transforming all sizes of the page using this equation the page will become flexible and its components will adapt to fit their container. Using this grid based layout at the webpage will give a structured flexible webpage. Other units that can be used instead or along with percentage are the \textit{vw} and \textit{vh} units which set the size relative to the size of the viewport instead of the size of the container. These units are so far not supported in all browsers, therefore percentages is off better use to ensure that the design is followed. [26]

A flexible grid will, no matter how small or big it is, still show all columns and the page will be shrunken or stretched to fit the size of the screen. This way of zooming out or in on the page containers will often not create a good user experience. It could result in content getting on top of each other and menus taking up the entire screen. In a responsive page these columns will instead be stacked on top of each other if the viewport is too small to show them all in one row. Figure 7 shows an example based on an image from \texttt{www.agencyofrecord.com}[^8]. This image shows how the grid in responsive web design can be used between different devices.

To sum up, a flexible grid (sometimes known as a fluid grid) is used to create a fluid grid-based layout, which stretches or shrinks so that the entire page horizontally is visible on whatever viewport it’s being viewed at. Because of its capacity to adapt, the flexible grid is the first cornerstone of responsive web design.

4.4.2 Flexible images
Flexible images are similarly to the components of a flexible grid, elements that scales according to their environment. If a browser window is scaled down, the images need to follow the rest of the site when it is getting smaller. In Ethan Marcotte’s book about responsive web design [2] there is also information about flexible images.

If an image is simply added to the html file of a webpage, the entire picture will be shown in full size. If the image is bigger than its container it will overflow the container and ruin the design. To solve this problem we can add `img {max-width: 100%}`\(^9\) to the project style sheet. This will set the max-width of all images to their containers widths. By doing this the image will scale with its container as long as it is bigger than the

---

\(^9\) Some of the older browsers do not support max-width, but this will not be discussed in this paper.
container. If the image is smaller than its container it will not fill the
container, which is often preferable since images do not always scale up
well. If it is preferable to always scale the image to fit the container, instead add the `img {width: 100%}` to the project style sheets.

An alternative to scaling the images to fit the container is to cut the
images. One way of doing this is to instead add `{overflow: hidden}` to the
container in the style sheet and add `img {display: block, max-width: auto}`. The image will always be in full size, but the overflow will be
hidden and the image will be cropped to fit the container. This is
generally not a good solution, since only one part of the whole image
will be visible and on small screens the image might not make sense.

A third solution is to create different images to fit different sizes. This
isn’t very responsive, but sometimes it might be better to create some
different sized images and let the server transmit the one best suited for
the screen size. To adapt the images to the screen size, the most suited
image is scaled or cropped to fit the container. This way the page is still
responsive to all different sizes even though not all different sized
images are available. This solution has another benefit (which has
nothing to do with responsive design); when a device with a small screen
requests the image, it gets a smaller image, which in turn requires less
data being transferred. That might be good for visitors on for example
mobile phones, where data download can be expensive. [2]

Images and media are not flexible by default. In order to make them
follow a flexible layout, the width need to be set to follow the container
they are placed within. By setting width or max-width to a percentage
value, images and other media can be used in flexible layouts.

4.4.3 Media queries
The first step towards media queries came with CSS2.1. This version
enabled the use of media types and enabled webpage developers to
adapt the cascading styles for some acceptable media types [11] [1]. For
example you could create different style sheets for screens, printing or
handheld devices [27]. The Idea was that browsers would identify what
media type it was part of. Some of these types were not used that much and they were kind of strict. For example, all handheld devices were treated the same, no matter if they had large or small screens. Here is an example of a style for the media type “print”:

```css
@media print {
  body { font-size: 10pt }
}
```

In CSS3 “Media queries” were introduced. WC3 describe them as following:

“A media query consists of a media type and zero or more expressions that check for the conditions of particular media features. Among the media features that can be used in media queries are ‘width’, ‘height’, and ‘color’. By using media queries, presentations can be tailored to a specific range of output devices without changing the content itself.” [28]

This means that developers have greater control and can independently style for devices with different features. If a media query is true for a viewport then the styles in the body of that query is applied to the webpage. An example:

```css
@media (min-width: 500px) { ... }
```

This media query is true for all viewports whose width is larger or equal to 500px

Several different media features are allowed when using CSS3. Media features are used in the queries to describe the requirements of different features of the viewing device. Table 1 gives a short description of the acceptable media features:
Table 1: Description of features which can be used in media queries

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>Describes the width of the target display area (not necessary screen width)</td>
</tr>
<tr>
<td>Height</td>
<td>Describes the height of the target display area</td>
</tr>
<tr>
<td>Device width</td>
<td>Describes the width of the display area of the device (screen width)</td>
</tr>
<tr>
<td>Device-length</td>
<td>Describes the height of the display area of the device</td>
</tr>
<tr>
<td>Orientation</td>
<td>Portrait (if height &gt; width) or landscape (if width &gt; height)</td>
</tr>
<tr>
<td>Aspect-ratio</td>
<td>Describes the ratio of width to height media feature</td>
</tr>
<tr>
<td>Device-aspect-ratio</td>
<td>Describes the ratio of device-width to device-height media feature</td>
</tr>
<tr>
<td>Color</td>
<td>Describes the number of bits per color component of the device (zero if non-color)</td>
</tr>
<tr>
<td>Color-index</td>
<td>Describes the number of entries in the color lookup table of the device</td>
</tr>
<tr>
<td>Monochrome</td>
<td>Describes the number of bits per pixel in a monochrome frame buffer</td>
</tr>
<tr>
<td>Resolution</td>
<td>Describes the resolution (density of pixels) of the device</td>
</tr>
<tr>
<td>Scan</td>
<td>Describes the scanning process of device (TV)</td>
</tr>
<tr>
<td>Grid</td>
<td>Tells if the device is grid or bitmap</td>
</tr>
</tbody>
</table>

Most media features allow min- and max-prefixes to be used for accepting certain devices. Not all media features are meaningful when creating a responsive web page, or an application that can be viewed from a desktop or a phone running Windows 8. Width, height and resolution are the most important features in this case. Testing for

---

10 Based on information from W3C organization (2012)[28]
several features in the same query is done by joining the queries with the “and” keyword. [28] [2]

When the layout is scaled, media queries can be used as breakpoints to setup different states which can have custom styles that will refine the layout to look better for the specific viewport and shift the layout depending on which queries are fulfilled. With a flexible layout and media queries a responsive webpage can be created. This webpage can stretch and rearrange components to better suit the viewport. The media queries can give knowledge about the size of the current viewport and are therefore needed in responsive web design to tell which layout should be used.

In chapter 4.4.1 Flexible Grid the web page “Responsive Grid System”, www.responsivegridsystem.com [24] was used to show how the flexible grid works. The differences between Figure 3 and Figure 4 shows the same layout only adapted through its flexibility. If the page is shrunk further a new state is entered to adapt the page to a smaller viewport. Media queries are used to set up these different states.
Figure 8 shows how the page is adapted when showed in a slim viewport were the state has changed from that used in Figure 3 and Figure 4.

---

V. Result

5. Research about responsive application design

During this thesis no documentation was found about responsive design for applications created in XAML and C#. To reach the goal of this thesis a solution was to gather pieces of smaller possibilities which together could form some idea of responsive design for Windows 8 applications created in C# and XAML. Knowledge, such as requirements, goals and possible components of responsive application design was gained through the literature study about responsive web design.

The basic concept of responsive design is to respond to the environment, to move. It can be walls bending when people get closer\textsuperscript{12} or components being moved in a webpage to better fit in the page when it’s resized. The idea in the beginning of this thesis was that if it can be done for the web, then it could probably be done for applications.

The idea of responsive web design and how it can make a difference on the web worked as an example for what responsive design could mean for applications. This research about responsive application design was based on the knowledge received from the literature study of responsive web design. The information in this chapter is a result of much research and testing of findings as well as a lot of own ideas.

5.1 What is Responsive application design?

To learn how to create responsive applications for Windows 8, what is meant with responsive Windows 8 applications, must first be specified. The first criterion in responsive application design should be that the application window should adapt to fit the screen size. This criterion could be matched to the use of a flexible grid in responsive web design. The criterion can be fulfilled both in Windows Store and Windows Phone

\textsuperscript{12} See video Interactive Wall at http://vimeo.com/4661618
8 applications, although the application windows usually aren’t resizable and typically take up the entire screen. There is though still a need for responsive design, partly because the size of the screen differ on different devices, but also because the phone should support both landscape and portrait mode, while the store application should support both regular and snapped view. This means that both applications must be able to change when the size available for the application changes.

Another way of thinking of responsive applications is regarding to one of the main reasons for responsive web design, the thinking of “one-app-to-fit-them-all”. Normally one application is created for each platform, one for phones, a second for computers and a third for tablets. This aspect of responsive design is to try to create one application to fit all these platforms. During this project no solution was found for creating only one project to run on both phones and computers. It is probably impossible to create an entire sharable application, but different applications can still share a lot of application code. Since responsive web design has to do with the design of the page, responsive application design could be seen as a way of using a shared user interface between applications for different platforms. However, to create that entire sharable user interface is difficult and for good reasons, not recommended by Microsoft. [18] Information about sharing can be found in chapter 5.2 Sharing code between Windows 8 and Windows Phone 8 and in Appendix 1. In this thesis a sharable user interface was produced anyway, at least as far as trying to create a common part of a user interface for both computers and phones running Windows 8. This was done in order to test if responsive application design can be used in this sense. There are a lot of differences between the phone and computer implementation and at the moment it seems that each platform needs its own project. Windows Store and Windows Phone 8 applications do not work exactly the same and doesn’t share all libraries, 

---

13 When an application is in snapped view it occupies a narrow region of the screen leaving the rest of the screen available for other content such as another application.
Although different projects are used, a lot of code can be shared between the projects by using different sharing techniques. In this thesis a part of the user interface was created to become part of this shared base, to show how a responsive design can be used and shared between different device types. [13] [18]

5.2 Sharing code between Windows 8 and Windows Phone 8

The goal with responsive application design is to “build one for them all” and to stop the madness of adapting the design for each device separately. In this thesis, research was performed to find a way of binding together the design, in C#/XAML applications, for both Windows 8 computers and Windows Phone 8 smartphones with different screen sizes.

When creating applications for both Windows 8 and Windows Phone 8 a lot of the code can be shared. When focusing on responsive application design, we are faced with the problem that even though lots of code can be shared, this is not true for all code. When creating C#/XAML applications the XAML documents are used to set up the user interface of the application. These XAML document also have an attached C# file which is in this thesis referred to as the code behind. It’s easier to handle differences of the platforms in code behind, than in the XAML documents. The focus for the solution in this thesis was to try to do as much as possible from the XAML document and only use the code behind when needed. One reason for this is that only code from the XAML document is visible in the designer, which is a helping tool in visual studio to show a preview the layout.

Another sharing obstacle is the need for different projects for the different platforms. There are also differences between the different types of smartphones and computers, for example not all smartphones are of the same size. Responsive design is needed to design good looking applications to fit all different kinds of phones and computers respectively. By using different techniques, much of the applications can
be shared between Windows Phone 8 and Windows Store applications, including much of the user interface. Sharing techniques is reviewed in Appendix 1.

In order to share a XAML page between Windows Phone 8 and Windows Store applications all of the XAML page must be working for both kind of applications. This results in some difficulties when only sharable elements can be used in the XAML document.

5.2.1 Using sharable elements in XAML layout

Windows Phone 8 and Windows Store apps have many components which are similar and named equally, but also many components which are not. The components in Windows 8 applications are called controls. In the XAML document only equally named controls can be used. Another limitation is that not all properties for the controls are the same for both platforms and only the ones that are can be used.

The containing page also differs between the platforms. Windows Phone 8 applications use the control \texttt{PhoneApplicationPage}, of the namespace \texttt{Microsoft.Phone.Controls}, while Windows Store apps use the control \texttt{Page}. Since these controls are named differently they cannot be used in a sharable user interface. There is another problem with this approach. Namespaces are defined differently in Windows Phone 8 applications and Windows Store applications. Therefor namespaces cannot be used in a sharable user interface. Not even the local namespace can be added to the XAML document. This is a problem since it prevents local resources from being used from the shared XAML documents. [18]

Microsoft has posted an article with a comparison of the controls [22]. The following controls can be used in both Windows Phone 8 and Windows Store applications:
### Table 2: Sharable controls

<table>
<thead>
<tr>
<th>Element</th>
<th>ItemsControl</th>
<th>ScrollViewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Border</td>
<td>ItemsControl</td>
<td>ScrollViewer</td>
</tr>
<tr>
<td>Button</td>
<td>ItemsPresenter</td>
<td>Slider</td>
</tr>
<tr>
<td>Canvas</td>
<td>MediaElement</td>
<td>StackPanel</td>
</tr>
<tr>
<td>CheckBox</td>
<td>PasswordBox</td>
<td>TextBlock</td>
</tr>
<tr>
<td>ContentControl</td>
<td>PopUp</td>
<td>TextBox</td>
</tr>
<tr>
<td>ContentPresenter</td>
<td>ProgressBar</td>
<td>ToggleButton</td>
</tr>
<tr>
<td>Ellipse</td>
<td>RadioButton</td>
<td>Tooltip</td>
</tr>
<tr>
<td>Frame</td>
<td>Rectangle</td>
<td>UserControl</td>
</tr>
<tr>
<td>Grid</td>
<td>RepeatButton</td>
<td>ViewBox</td>
</tr>
<tr>
<td>HyperlinkButton</td>
<td>ScrollBar</td>
<td>VirtualizingStackPanel</td>
</tr>
<tr>
<td>Image</td>
<td>ScrollContentPresenter</td>
<td></td>
</tr>
</tbody>
</table>

Other elements can also be added to the page, but it has to be added through the attached C# file, were conditional compilation can be used to implement the layout differently for the different platforms. In this list of controls one of the controls, the *UserControl*, is more interesting than the others. The *UserControl* can be used to form a new component by one or several other controls. The layout of the page can be created as a *UserControl* and inserted to the page. Since the *UserControl* is sharable it is a great possibility for sharing parts of the user interface by adding the same *UserControl* to the different application pages. The *UserControl* has many similarities with an application page. It has a XAML document to form the layout and an attached C# file. To make it sharable, namespaces cannot be used and only sharable controls, events and properties can be used to setup the XAML layout. The one thing that
differs and makes it possible to share is that it is referenced equally as `UserControl` in both applications. [18]

The limitation to only use properties and events that is equally named for both platforms is a true obstacle since many properties and events is named differently. One of the biggest obstacles is the *Tap* event which handles when a control is clicked. It is named differently for the platforms and cannot be used from the XAML document. One tap event that seems to be sharable is the click event of the button. Therefor all images and other content that should be clickable should be either within a button or in front of a button. This is one of the obstacles that make the sharable user interface difficult to use and often not worth the effort.

### 5.3 Testing for responsiveness

Testing if an application is responsive is done by checking that the application looks good on several different screen sizes. This involves readable content, which does not overlap and a layout that is acceptable in order to create a good reading experience at the viewport. Windows Phone 8 applications are not resizable, and Windows Store applications are only shown as a full screen application or in a snapped view. This doesn’t result in very good responsive testing. One way of testing Windows Store applications, is to try it on different screens with different sizes to run the application. Another idea is to instead use the simulator, which can change resolution, to run the application. Figure 9 shows how to change the resolution for the simulator.
The testing is still rather slim, since very large and very small screens cannot easily be tested without getting hold of such screens. By testing the application in snapped view, at least smaller resolutions can be tested.

For Windows Phone 8, several different emulators with different resolutions can be used to test the responsiveness of the user interface of the application. Also here the restrictions of the screen size are obvious. At the moment the emulators’ resolutions covers all existing ones on the market, but that might not be true for the future. For now and for this project it will have to suffice to use the different resolutions of the emulator and the simulator to test the applications.

The following image, Figure 10 shows the supported resolutions of Windows Phone 8, which are also the resolutions available for the emulator.
One of the biggest problems, that took a lot of time during the development of the applications, was the testing of the Windows Phone social application. The program Fiddler was used to control the messages sent from the application. Setting up the emulator with Fiddler was difficult and often it did not work. Around mid-time of the project it finally worked to the fully. Today more exact instructions are available\textsuperscript{15}. These instructions where a lot of help, but some additional settings were needed to make it work for this project. In the instructions it says that the computer name should be used when setting the registration hostname. I have discovered that it is the full computer name that must be used. In this project https traffic must also be decrypted in order to see the content of the sent packets. This is needed because messages sent with the https transport protocol are encrypted. Https decryption can be set using the instructions found on the Fiddler page.\textsuperscript{16} For this to work, the root certificate found in the https tag of the

\textsuperscript{14} Image from: Microsoft (2013) [15]
\textsuperscript{15} Information on how to set up Fiddler to trace traffic from the Windows Phone 8 emulators is found at Nico’s digital footprint: \url{http://www.spikie.be/blog/post/2013/01/04/Windows-Phone-8-and-Fiddler.aspx}
\textsuperscript{16} Information on how to setup https decryption in Fiddler is found here: \url{http://fiddler2.com/documentation/Configure-Fiddler/Tasks/DecryptHTTPS}
Fiddler options page, needs to be downloaded and installed both at the computer and the emulator. When this is downloaded to the emulator the https decryption must be turned off, otherwise the emulator will have no internet access before the Fiddler root certificate is installed.

Another issue with the testing of the applications was the need for unregistered certificates in order to connect to Infor Social Space. The emulator is reset on each run and the certificates must therefore be downloaded and installed at every time the emulator is reopened. This was solved by learning how to save the state of the emulator so that the installation of the certificates weren’t reset on restart.

Testing with different phone and computer screen resolutions can be done. But at the moment only predetermined screen resolutions can be tested and there is no simple way to test the extremes.

5.4 Components in responsive application design

When developing a responsive Windows Phone 8 and a Windows Store application one way of finding a solution is to find building blocks replacing those in responsive web design. Responsive web design is achieved by using flexible grids, flexible images and media queries. Though these are the building blocks of responsive web design, they are not necessarily the components of responsive application design but they are a good starting point. In this chapter, the components of responsive web design are the base for researching the possibilities of responsive design for Windows 8 applications created in C# and XAML.

5.4.1 Flexible Grid

The first cornerstone of responsive web design is a flexible grid. In the design stage this grid is a helping image to provide a visual tool for creating a structured, fluid layout. Since this grid is visual, the developer still has to style the webpage to fit the grid. If the grid is well defined, the grid shows the different sizes of the columns, rows, gutters and so on. Then, the developer must carefully follow these numbers when developing the page and when finished check against the grid to make sure that it is followed.
When developing in the XAML designer a lot of code is auto regenerated. This is often an advantage, but sometimes margins and other properties are specified which makes the layout differ from the flexible grid. When designing the user interface it is important to make sure that it does function as planned and the visual grid is a great tool to use to make sure that the elements are added correctly to the page.

When developing a Windows Phone 8 application from a template, an image of an alignment grid, shown in the emulator in Figure 11, is provided as part of the template. The code for showing this image just needs to be uncommented in order to present it on the page. This alignment grid can be used to control that there is structure on the page and can be used as a simple visual flexible grid. If there is a need for another kind of flexible grid, that flexible grid can be added to the project as an image and shown in the same way as this alignment grid.

![Figure 11: Alignment grid on emulator](image)

In XAML another grid is available. This grid is a functional grid that can contain the content directly. It is similar to an invisible table, but richer. This functional grid can be to implement the design of the flexible visual grid. Since the flexible grid is a visual tool, it can also be used in
application design. The XAML grid together with other containers can be used instead of the div-tags that are defining the containers in html.

A flexible grid is flexible and stretches to fill the browser window. Implementing a fluid layout in responsive web design is done by using percentages to setup the layout. When creating an application percentages cannot be used when specifying the widths in XAML code. Instead the container grid can be used to set up the layout.

**XAML grids**

The XAML grids are a very useful tool when structuring content. In the applications created during this thesis the root content container is a grid. When creating an application from a template the root grid is already added to the XAML document. Often grids are also used in the layout as content holders in templates. The default grid `<Grid>` is flexible and has from a responsive view the following interesting properties:

- Width and Height is set to auto, MinWidth and MinHeight are equal to 0 and MaxWidth and MaxHeight are set to infinity. HorizontalAlignment and VerticalAlignment is set to stretch. If some other alignment is used the grid will only adapt to fit its content, not stretch to fill the viewport. The different widths and height can be set to static sizes by specifying the size in pixels if a non-responsive approach is endeavored.

```xml
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition />
    <ColumnDefinition />
    <ColumnDefinition />
  </Grid.ColumnDefinitions>
</Grid>
```

A grid contains columns and rows, which are defined as column and row definitions. The example to the right shows a grid with 3 columns with equal widths.

In the XAML grid, columns and rows can be of different sizes. `ColumnDefinitions` only has three properties; Width, MinWidth and
MaxWidth. Widths of column definitions can be set to a specific pixel value, auto or a star value.

The value auto makes the column adapt to fit the content, but this might overflow the Grid container and therefore cut information. If instead the columns should adapt to fit the size of the grid, a number and the marking of a * (star) is used. If all columns have the same star number they will all be of the same size and stretch to fit the grid. If one column has a larger star value, this column will be wider. For example, if there are four columns with the widths 1*, 1*, 1* respectively 2*, the last column will be twice as big as the others and take up 2/5 of the grid, which is equal to 40%. As a standard a column has the width 1*.

All columns can use different settings for the widths, and there are a lot of rules in which order a width property is fulfilled. If the goal is to have a completely flexible layout that satisfies a visual grid, a recommendation is to use only star values as the values of the width. This way, the page will be fully flexible and all column widths will be of some percentage of the grid width, since no solid widths mess up the splitting between the different columns. Often some columns need to be set with auto as the value of the width. For example components such as menu items or thumbnails shouldn’t grow with the rest of the layout. When auto is used for one of the columns, its width will be decided before the star columns, which have to share the remaining area. The properties MaxWidth and MinWidth can also be used to prevent a column from growing too big or too small.

The same is true for the heights of the row definitions. But when designing with both columns and rows, the design must be thought of extra carefully. It might be better to use auto as the setting for the row definitions, to ensure that no content is being cut off on small screens or large whitespaces taking up space on larger screens where the content doesn’t fill the container. If the height of the page is too small, content might still be cut off if no ScrollView is added to the page or grid.
To make sure that the components follow the grid layout the horizontal alignments of the components should be set to *stretch*. This way a component stretches to fill the column it is added to. If the columns width is defined as auto, the horizontal alignments might not need to be set to *stretch*.

In responsive web design the grid have several columns and sometimes also rows. The content can span several columns and rows. This goal of a visual grid is also satisfied by the XAML grid, where the content can be defined to span one or several columns or rows. The XAML grid is great for implementing the layout in this way.

Flexible grids also have gutters and other margins. In XAML a lot of properties can be set for the grid, but defining gutters is not one of them. An outer margin can be set for the grid, but the inner margins must be applied to the content itself. One way to do this is by specifying styles for the different components of the page and setting the margin in these styles. For example the style for setting the margin of all text blocks of the page would look as follows:

```xml
<Style TargetType="TextBlock">
    <Setter Property="Margin" Value="5" />
</Style>
```

The style should be added to the resources of one of the containers holding the text block. It can be added to the page resources if it should be used for all *TextBlocks* at the page or at the closest container if only the *TextBlocks* in that container should acquire that style. When a style is named or has another identifier it will not be used by default, but must instead be referred as a style reference. So if only some *TextBlocks* should use the style, the style can be named with the property *x:Name* and referred to from each *TextBlock* that should use the style.

### 5.4.2 XAML Images

Flexible images are the second principle of responsive web design. When using html an image is added with an image tag, `<img>`, and becomes
flexible by adding the max-width criteria in the style sheet. In XAML there is also an image tag, `<Image>`, to hold the image, which in this thesis sometimes is referred to as the image holder.

Images should be able to stretch in different ways depending on what is desired in a specific situation. From the chapter of flexible images in responsive web design we get the following requirements of techniques:

The image should be able to

- Stretch to always fit the container
- Be cut to always fit the container
- Be cut to fit the container only if the image is bigger than the container
- Stretch to fit the container only if the image is bigger than the container

It is also good if the image holder could: Scale or crop the image that best suits the current size of the window.

In a XAML image the `Stretch` property of an image are as a standard set to `uniform`. Other alternatives for the stretch property are `none` (to not stretch the picture at all), `fill` (to fill the container) and `uniformToFill` (fills the entire container but with the ratio between height and width conserved)

As a standard when an image container is added to an XAML document, it is resized to fit the container. This is done by using the setting the property `Stretch` to `uniform`. The container (for example a column of a grid) of the image holder can be used to decide the maximum size of the image. When the value `uniform` is used, the image keeps its ratio between height and width. The one direction that first restricts the image by reaching the size of the container, also decides the size in the other direction in order to keep the ratio between them. [15]

By using these settings the first requirement is met:
• Stretch to always fit the container

The textbox below shows the representing XAML code. In this case, only Source is specified, since the Stretch property is uniform by default. The property can also be added to the tag as Stretch=“Uniform”. Height and width is set as auto and alignments are set to stretch as default.

```xml
<Image Source="Images/myImage.jpg" />
```

The designer though, has to make sure that the image is of the correct proportions. If the image is only restricted in one direction, it will be centered in the other. If this is not the desired experience, there are other property values to investigate. The alignment properties decide in which direction the image should flow. If the container is instead supposed to be filled completely the designer has to decide if the image should instead be stretched to fill the container by doing it uniformly or to stretch the widths and height without keeping the ratio between them. [15]

The second alternative can be achieved by choosing fill as value for the Stretch property. The use of this value is another way of fulfilling the first requirement:

• Stretch to always fit the container

```xml
<Image Source="Assets/img4.jpg" Stretch="Fill" />
```

The first alternative can be done by instead choosing uniformToFill as the value for the Stretch property. With the value uniformToFill the second requirement of images is achieved:

• Be cut to always fit the container

```xml
<Image Source="Assets/img4.jpg" Stretch="UniformToFill" />
```
The image can also be cut in a specified way using the clip property. The clip property specifies which part of the image will remain. Here is an example where everything but a small circle is cut away:

```
<Image Source="/Assets/img4.jpg">
  <Image.Clip>
    <EllipseGeometry RadiusX="30" RadiusY="30"
                     Center="50,40" />
  </Image.Clip>
</Image>
```

The last option available for the stretch property is the value *None*. This setting fulfills the third image requirement:

- Be cut to fit the container only if the image is bigger than the container

```
<Image Source="/Assets/img4.jpg" Stretch="None" />
```

With this setting the image isn’t scaled at all. If the image is bigger than its container, its overflow is cut. The image is as standard aligned as stretched and will be centered if the image is smaller than the container. If the image shall be aligned in some other way, the alignment properties must be changed. When the image is bigger than its container, it’s difficult to control which part of the image should be cut with only alignments. Then the property clip can be used to specify how to cut the image. [15]

The fourth requirement is more complicated. Resizing the image to fit the container is done by using the value *uniform* for the property *Stretch*. If the image should not be stretched to be bigger than its real size, the *MaxWidth* property can be set to its true size in pixels. In web design, this is done by just setting the max-width for all images to 100%. In C# and XAML applications it’s more complicated. The trick is how to know the true size of the image. If the size is known, it can be set directly as a pixel value but if the size is not known, it is more difficult. In Windows Phone 8 and Windows Store applications the image source can
be used to connect to the properties of the image. The \textit{MaxWidth} property of the image holder can be bound to the width of the image by binding to the \textit{PixelWidth} property of the source in the XAML code as following:

\begin{verbatim}
MaxWidth="{Binding Source.PixelWidth, Mode=OneWay,
RelativeSource={RelativeSource Self}}"
\end{verbatim}

But here a problem occurs; the image must be opened before these measurements are available. Just binding the \textit{MaxWidth} directly will set it to 0, and the image will not be visible. One way to work around this problem is to set the \textit{MaxWidth} from an event handler that is executed when the image is opened. At which time the image source measurements will be available.

\begin{verbatim}
<Image Source="Assets/img4.jpg" Stretch="Uniform"
ImageOpened="OnImageOpened"/>
\end{verbatim}

\begin{verbatim}
private void OnImageOpened(object sender, RoutedEventArgs e)
{
    var s = sender as Image;
    if (s != null)
    {
        BitmapImage b = s.Source as BitmapImage;
        if (b != null)
        {
            s.MaxWidth = b.PixelWidth;
        }
    }
}
\end{verbatim}

This fulfills the requirement:

\begin{itemize}
  \item Stretch to fit the container only if the image is bigger than the container
\end{itemize}

The last option is one that is recommended by Microsoft for Windows Store applications, but not for phone applications [15]:

53
• Scale or crop the image that best fits the current size of the window

To make this possible, first an image must be available in several sizes. The images must be added to the same folder and they must be named in regard to the resolution of the image according to Microsoft’s naming conventions. For Windows Store applications the images should be named as following: [16]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>myImage.scale-100.ext</td>
<td>for typical 96dpi device and</td>
</tr>
<tr>
<td>myImage.scale-140.ext</td>
<td>for 140% the original size</td>
</tr>
</tbody>
</table>

The images are referenced without qualifiers [16]. Here is an example from the XAML document:

```xml
<Image Source="images/myImage.png" />
```

The best suited image will be chosen automatically, regarding to contrast settings and display scaling. [16]

This is also possible for Windows phone, but in order for it to work, code must be written to handle the determination of the resolution and for picking the correct image. This last technique will not be used in the project, and it is unnecessary in the matter of responsive design. More information about this subject for both Windows Phone 8 and Windows Store applications can be found on Microsoft’s webpage. [15] [16]

**Style several images at once**

There is a disadvantage with the solution of adding an event handler to set the *MaxWidth* of the image to solve:

• Stretch to fit the container only if the image is bigger than the container
The disadvantage is that the *MaxWidth* must be set for each image, which is a difference compared to web design, where all are set with one statement. Normally a property for controls can be set as a resource style, which can be applied to all controls of a specific type. The problem is that events in a style need to be set with an *EventSetter*, which is not available in Windows Store and Windows Phone applications. The event handler must therefore be attached to each image at the page. [25] There is an exception to this rule. When the container can use an *ItemTemplate* and a source, to specify the content of the container, the event can be set in the *ItemTemplate*.

### 5.4.3 Media queries

Media queries do not exist in C# or XAML. Instead we have so much more possibilities to write code. XAML elements have different width properties such as *Width*, *MinWidth* and *MaxWidth*. These values can be bound to some other property or element, specified in pixels or set to *Auto*, which corresponds to infinity for *MaxWidth*, 0 for *MinWidth* and “fit to content” for *Width*. There are also other properties that should be used to make the application responsive. Many elements have the *HorizontalAlignment* and *VerticalAlignment* properties, which can be set to *stretch* to fit the container. Some containers also have the properties *HorizontalContentAlignment* and *VerticalContentAlignment*, which sometimes also have to be set to *stretch* in order to make the content flexible.

These properties have been used earlier to make a fluid layout. Many of the requirements of responsive design have been solved in the recent chapters, but there are still some requirements left. The following techniques for the layout should also be possible:

- Elements in the layout can be rearranged for certain windows sizes.
- The number of columns and rows used, can be adapted to better suit the size of the window.
Another requirement for the elements in the page also needs to be solved:

- Elements in the layout can be hidden for certain window sizes.

The mutual commandment in these requirements is the window size. In responsive web design the media queries are used to set breakpoints, where the layout should be changed. But in C# and XAML applications there are no media queries.

In Windows Store applications a WrapGrid or a VariableSizedWrapGrid can be used to automatically adapt the number of columns to fit the window size. They are easy to use if all items are of the same size, but are more difficult to use when all elements are of different size. One possibility is to let larger components span multiple rows and/or columns. But these containers are not available for Windows Phone 8 and therefore these techniques cannot be used in a sharable user interface between Windows Store and Windows Phone 8 applications.

In the Windows Phone toolkit\textsuperscript{17} there exists a WrapPanel that can be used to fulfill the second requirement. If the application has two columns of information, these two can be added to WrapPanel. If the layout is wide enough to fit them both, they will be on the same row, if not the second one will instead be put under the first one. This is a great solution for Windows Phone 8, but since the toolkit is added as a namespace, and namespaces is added to the XAML-document in different ways for Windows Phone 8 and Windows Store applications, it can’t be used from the XAML document in a shared UserControl. The sharing obstacles regarding the XAML-file are truly making this difficult. But there are possibilities for sharing in the code behind.

The applications can use different wrapping controls if they are added in the code behind, for example if the content is first added to a content control in the XAML document, then collected in the code behind and

\textsuperscript{17} The Windows Phone toolkit was published in October 2012 and is available at the site of CodePlex: \url{http://phone.codeplex.com/}
put into our wrapping control before that container is added to the layout. In the code behind, conditional compilation can be used to add the WrapPanel for Windows Phone 8 and the VariableSizedWrapGrid for the Windows Store application. This is a workaround and of course the whole content, which should be in the wrapping content control, can be added in the code behind without first being initialized in the XAML document. In the code behind the WrapPanel and Variable-SizedWrapGrid must also be initialized and set up with the correct properties. The VariableSizedWrapGrid isn’t as simple to use as the WrapPanel is. Since we now have the benefits of the code behind sharing techniques, a WrapPanel can be constructed for the Windows Store application\(^\text{18}\).

By using WrapPanels from the code behind of the page, the second requirement can be fulfilled:

- The number of columns and rows used can be adapted to better fit the size of the window.

To solve the other requirements, we need knowledge about the size of the page. After the initialization, the width of the page can be retrieved. There is also a possibility to add an event handler for when the size of the page changes. This might be needed when the orientation changes or when the user shifts between snapped view and regular view for Windows Store apps. This way the size of the window is always known. If the change of the size is passing some predefined threshold, which divides the different states, a method that updates the layout to suit the state can be executed.

In responsive web design one example of the first requirement, to rearrange certain components for different states, could be changing the menu and moving it from the left side of the page to the top or vice versa. One idea to perform this in application development is to have an

\(^{18}\) Code for creating a WrapPanel for Windows Store apps is available at [http://www.codeproject.com/Articles/24141/WrapPanel-for-Silverlight-2-0](http://www.codeproject.com/Articles/24141/WrapPanel-for-Silverlight-2-0)
extra empty row at the top available for the menu if it’s going to change. A row without content doesn’t take up space in applications if the height of the row is set to auto. When the threshold for moving the menu is reached, the menu can be moved from the code behind. This is one solution which solves the requirement:

- Elements in the layout can be rearranged for certain windows sizes.

Elements of a page have a visibility property. This property can also be updated when some threshold is met for hiding or showing an element of the page. From the code behind a visibility dependency property can be added. This property can in turn be bound to the elements that should be hidden for some state. This way the method that is run when the threshold is reached can simply change the visibility of that dependency property, which in turn will update all elements bound to that property. If the control is added directly to a grid column the Width of the column it’s added to must though be defined as auto or changed from code behind, otherwise the column it is added to will take up space even when the element it contains is hidden.

### 5.5 Creating a responsive user interface

In the chapter 5.4 *Components in responsive application design*, the building blocks for responsive application design were discovered. Such as the XAML grid, that can be used to set up the basic layout, the images that can be stretched in different ways and the wrapping panel, which can adjust the number of columns to fit the page. An application is though more than just images and text, and many more objects and elements need to be responsive.

A typical webpage contains a menu, text and images. In application development there are a lot of other elements that are also used. To manage this in the Windows Phone 8 and Windows Store applications, the developer needs to design carefully. This chapter shows how to set up some different controls of the user interface to be responsive.
5.5.1 Making elements responsive

Elements such as containers to text blocks, buttons and containers that are used at the page must also be responsive. These elements should also be able to:

- Stretch to fit the container
- Adjust the container so it suits the element

All controls don’t start with the same default values. The default values are the values that the properties have when only the tag is set, for example `<TextBlock />`. A button in a Windows Store app does not automatically stretch as the image does. The button has a default value of `left` as `HorizontalAlignment` and `center` as `VerticalAlignment`. The `Width` and `Height` are just like the image set to `auto`, which results in a button that changes size to fit its content. If they instead should scale to suit the container, the alignments should be set to `stretch`. In the Windows Phone 8 application the button alignments are as default set to `stretch`, so the designer needs to make sure that it stretches correctly in both applications. A problem with for example `Buttons` and `TextBlocks`, is that if the text is longer than what fits in the container its cut. As default the `TextBlock` uses the value `No Wrap` for the wrapping property. Although the `TextBlock` normally stretches to fit the container, the content of the `TextBlock` is cut when it’s longer than the width of the container. If the value `Wrap` is used instead the text continues on the next line and is not cut off. When the application is flexible the `TextBlocks` should always have this functionality. Therefore it might be a good idea to make wrapping the default style of all textboxes on the page. This is done by adding the style to the grid or page resources as such:

```xml
<Page.Resources>
    <Style TargetType="TextBlock">
        <Setter Property="TextWrapping" Value="Wrap"/>
    </Style>
</Page.Resources>
```
The button content text does not wrap either and it does not have any text wrapping property. But the problem can be circumvented by adding the following default style for buttons to the grid or page resources:

```xml
<Style TargetType="Button">
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <TextBlock Text="{TemplateBinding Content}" TextWrapping="Wrap" />
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

If the style is named or is set with a key, the style will not be assigned to targets by default and must be added as a static style resource to each object that should use it.

With these styles added, all Buttons and TextBlocks will use the wrapping style. But this is not true for all other elements. If a container shall be used with text as content a style template just as the button template above can be created, if the text isn’t already wrapped as default. Not all controls can have this template. For those that can’t use the template the content can instead be added as a TextBlock to achieve the TextBlock style on the content. A problem with adding the style to the button or any other content presenting element with a template is that the content of the element only can be of type TextBlock since it’s specified in the ControlTemplate. If content of several types should be used, it is better to instead set the content by adding it as an own tag and set the styles for each content type as a specific resource. Since the style is already correct for TextBlocks, the content of the button can be added as a TextBlock:

```xml
<Button x:Name="myButton">
    <TextBlock Text="Click the button, this is the button text END" />
</Button>
```
Adding styles to a document is also a great way of making sure that the elements of a type in the whole page works in the same way at both platforms. The alignments can be set even though it is already default in one of the applications.

To make sure that the elements always stretches to fit the container, according to the requirement, a style for each target type that is used can be added where the alignments are set to stretch. Some controls also have the `HorizontalAlignment` and `VerticalContentAlignment` properties, which also should be set to stretch.

```xml
<Style TargetType="Button">
  <Setter Property="HorizontalAlignment" Value="Stretch"/>
  <Setter Property="VerticalAlignment" Value="Stretch"/>
</Style>
```

If the content is larger than its container the overflowing content will be cut off. One good solution is to not restrict the container in both the vertical and horizontal direction. If the `Height` of the row is set to `auto` the container will stretch in the vertical direction to fit the content. But content might still be cut off if the application page is too small to show it all. Another solution is to add the element or the container to a `ScrollViewer` control, which makes the container scrollable so that all content can fit. With one of these settings the requirement “Adjust the container so it suits the element” is fulfilled.
6. The Infor Social Space Application

Since no way of creating just one application to fit all platforms were found, two applications, one Windows Phone 8 application and one Windows Store application, were created. The applications were created in order to test the found solution for responsive application design. Since responsive application design in this thesis is specified as to design a responsive sharable user interface, the different pages of Infor Social Space was created to be a sharable part of the user interface for the applications.

The sharable parts of the application such as the feed page, post page, colleagues’ page, groups’ page and profile page are set up as different UserControls. These different UserControls are added to the non-sharable application pages. The attached C# files of the UserControls were used to handle differences between the applications.

Responsive design is difficult to use in applications. The most difficult part is to make the user interface sharable. This is difficult because there are many differences between Windows Phone 8 and Windows Store applications which limit the possibilities. The end result of the responsive Infor Social Space application weren’t fully responsive. The UserControls were shared between the applications, but they were used differently on the non-sharable application pages. Max values have also been used to setup the UserControls which prevents the application from being fully flexible. A lot of margins are set up with fixed values which also make the page less flexible.

The following images and explanations will show how the applications were used to test different parts of the responsive solution.
Figure 12: The feed page shown on a Windows Phone 8 emulator in landscape mode.

Figure 13: Feed page shown on Windows Phone 8 emulator in portrait mode.
Figure 12 and Figure 13 shows the feed page of the Infor Social Space application in the Windows Phone 8 emulator in landscape respectively portrait mode. In landscape mode two buttons for commenting and deleting a post is shown that is not shown in the portrait mode image. This shows how objects can be hidden or shown depending on the width of the screen. The buttons Visibility property is bind to a property in the code behind. Each time the window is initialized or resized a method is run which decides if the buttons should be shown or hidden and sets the Visibility property in the code behind which in turn updates the Visibility property of the buttons.

The margins between the different posts are also dependent on the width of the application. The posts in Figure 12 are more separated than the once in Figure 13, since the application window is wider in landscape mode.

The HorizontalAlignment property of the feed page UserControl is set to stretch to make the layout stretch to fit the size of the window.

The posts are presented in a ListBox by binding the ItemSource property to the collection of posts. An ItemTemplate is used to setup how the post should be presented within the ListBox. Each post is added to a separate Grid, within the ListBox, which stretches to fill the ListBox container since both alignments of the Grid and the alignments and content alignments of the ListBox is set to stretch. The profile images of the feed page are set up using the value uniform as Stretch property. This makes the images stretch to fit the column of the grid it’s added to. The difference in the width of the profile images is easier to notice when comparing the images of Figure 12 with Figure 15. The container of the images is the first column which has a MaxWidth of 80. The MaxWidth property is set in order to prevent the images from becoming too big when the application window grows. The last column contains the comment- and delete buttons and also has a MaxWidth of 80. It uses the value auto as Width to make the column collapsible when its content is hidden. This leaves the rest of the space to the middle column
containing the message which uses a star value as value of the Width property.

Figure 14: Feed page for the Windows Store application shown in simulator with smaller resolution in portrait mode
Figure 15: Feed page for the Windows Store application shown in simulator with higher resolution in portrait mode.
Figure 14 and Figure 15 shows the feed page of the Infor Social application in portrait mode with two different resolutions. The feed page user control is in both cases stretched to fit the application window and even if the width differs the layout is flexible and stretches. Normally this is done by setting the horizontal alignment to *stretch*. But in this case a *GridView* is used as container in the application page and the *GridViewItem* is allowed to grow to infinity. Therefor the *MaxWidth* property is instead used to set the width of the *UserControl* and it is set to a calculated value from code behind. This is just one possibility for making the layout flexible.

The application pages have been set up differently in the phone and in the store application. In the phone application the feed page always stretches to fit the application Window. But in the store application states are used to decide if only one or several *UserControls* should be shown at a time. The states are created by running a method that controls the width of the application window and sets the sizes of the different elements. In this application if the application window is wider than 900 pixels, the *UserControls* doesn’t stretch anymore but instead takes up a maximum width of 800 pixels which allows for more parts of the *GridView* to be shown. In Figure 16, both the feed page and the colleagues page is shown when the resolution used in Figure 15 is switched to landscape mode and the width exceed 900 pixels. Figure 17 shows the page in the largest simulator mode, were several parts of the page is shown at the same time.
<table>
<thead>
<tr>
<th>FeedPage</th>
<th>Colleagues</th>
</tr>
</thead>
</table>
| Sofi Klockare  
Trying to post from phone emulator 2013-04-12 15:12:42  
Nr of comments: 5 | Fredrik C Eriksson  
System Development Manager |
| Sofi Klockare  
Trying to post from phone emulator 2013-04-15 10:23:28  
Nr of comments: 1 | Han Lin Yap |
| Martin Hansson | Karin Portillo  
Software Architect |
| Katarina Ljungdahl | Lovisa Gannholm |
| Manipunath Ganimaty |  |

Figure 16: Feed page and colleague page shown at Windows 8 simulator in landscape mode.
Figure 17: Several parts of the page are visible on a wide screen without scrolling.

Each part of the solution has been tested individually although not all parts have been used within these social applications.
VI. Discussion

7. Discussion
Starting off this project a smaller study was conducted to learn about Windows 8. To achieve knowledge about responsive design, a literature study in responsive web design was performed. During this study not only the components and techniques for how to apply it were found, but also the possibilities and the main reasons for using responsive web design. The two latter was perhaps even more rewarding for this project than the first.

The intent with this thesis was to find out if responsive design can be used when developing Windows 8 applications in C# and XAML. This goal was broken down into the following three sub questions:

- What is responsive web design?
- What is or what could responsive design be in Windows 8?
- Can responsive design be used in Windows 8 applications?

In chapter 4 *Responsive web design*, the answer to the first question can be found. In short responsive web design is one way of designing webpages so that they adapt to fit the viewports they are being viewed at. This technique uses a flexible grid and flexible images to make the layout stretch. Media queries are used to setting breakpoints to form different states. In each state the layout of the page can be redesign by for example changing the numbers of columns, hiding components and changing styles.

The second question was answered in chapter V *Result*, and some possible concepts were presented in chapter 5.1 *What is Responsive application design?*. In this thesis responsive design for applications means to create a responsive user interface which adapts to fit the
screen and is sharable between Windows Store and Windows Phone 8 applications.

The perhaps most important question is the last one. The answer to this question depends very much on the answer to the second question. It doesn’t seem to be possible to create only one application that should work for all Windows 8 devices. If this would have been the answer to the second question, then the answer would have been “No”. But the chosen answer for the question in this thesis, is that responsive application design is limited to only share the user interface between the different applications, then the answer is instead yes, or at least partly yes. In chapter 5 *Research about responsive application*, different techniques is shown for creating several responsive pieces which together can be put together to form an almost responsive user interface.

By specifying the layout in a *UserControl* and only using controls and properties that can be used by both platforms in the XAML document, parts of the layout can be shared. Although *UserControls* can be used to set up most of the layout, the navigation in the Windows Phone 8 application should be handled from the page. The *UserControls* must also be added to the page and because Windows Store apps and Windows Phone applications use different kinds of pages, with different names, the pages are developed separately for each platform. No local resources can be used within the shared XAML documents.

To make the layout of the *UserControl* responsive, basically controls should be set to stretch horizontally by setting the *HorizontalAlignment* to *stretch* and the *Width* to *auto*. A grid can be used to set up the basic layout and a wrapping grid can be used to set up the columns of content. If a container has the *HorizontalAlignment* property, this should also be set to *stretch*. Images can be defined to follow the layout in different ways by using the *Stretch* property. Elements can be shown or hidden by setting the *Visibility* property and a *WrapPanel* can be used to automatically use a suitable number of columns with information. The code behind can be used to setup different states and
managing the update of the layout of the page. More in-depth instructions can be found in chapter 5 Research about responsive application design.

7.1 Analysis of result

Responsive design within Windows Phone 8 and Windows Store applications is not a completely new idea. Microsoft has written at least one article about how to code for multiple resolutions within Windows Phone 8 [15]. But this is just an adaptive technique such as fluid design in chapter 4.1. The design will just be stretched to fit the screen of the phone. For now, that might be enough. Only a couple different resolutions exist for phones with Windows Phone 8 and that is probably how it will be for phone using this operating system. But in the future for the next operating systems it would probably be good to operate against a more responsive approach.

Another disadvantage with this technique is that sharing user interface between different platforms will not give a good user experience. Just as fluid layout is not good enough to create a good experience on the web for both phones and big computer screens, this fluid technique will not do a good job for applications. It is however difficult to share the user interface between platforms.

The solution presented in chapter V Result, is just one solution for using responsive design in applications. It probably exists other ways of creating a responsive application. Not all XAML controls have been tested within the solution and it probably exist more obstacles then those found during this project.

No good solution\(^\text{19}\) of how to share the entire user interface was found. There are so many differences and difficulties to work around which are making it really complicated to find good solutions. Especially sharing the user interface has been challenging. The design becomes limited by

\(^{19}\) Except for adding the entire content in the code-behind, but that technique isn’t very good.
only using sharable controls, properties and events and it must constantly be tested in both applications to make sure that no code has been written which doesn’t work for one or the other of the platforms. One of the biggest show stoppers has been that conditional compilation can’t be used when writing XAML code, which makes differences difficult to work around. Other limitations are that local resources cannot be used from a shared XAML document. One of the platforms often limits the other in the XAML document sense they both must be able to execute the code. The specific phone and computer styles can also not be used in the shared XAML documents.

In comparison to responsive web design, responsive application design is more complicated. The flexible grid can be used in both designs, but the implementation is different. Sometimes there are advantages with implementing the layout in application design, especially if the layout is completely specified in XAML since then the designer in Visual studio 2012 can be used to visualize the layout. The XAML grid is also easy to work with when the basics are known. Although it’s easy to create a mostly fluid layout by using a grid and star values, more work is needed to make the margins and content adaptable. Therefore it’s difficult to make the layout follow the flexible grid exactly.

For the flexible images it’s easier to make the images stretch to always fit the container in responsive application design. But to make it only stretch when it’s smaller is much more difficult. Instead of just specifying the \textit{max-width} to 100\%, which is done in responsive web design, the width of the image must be collected from an event handler run when the \textit{ImageOpened} event is run and then the \textit{MaxWidth} can be set to the retrieved value.

No direct replacement for Media queries exist in responsive application design. There is a \textit{SizeChanged} event which can be attached to an event handler to check the size and run some method to change the layout. Wrapping panels can be used to automatically adapt the number of columns used for presenting content. To hide elements, the elements
can be bound to a visibility property which can be set from the method which updates the layout.

An advantage for responsive application design is that the code in the file which is attached to XAML document can be used to change the layout completely. From code behind the layout can be adapted differently for the different platforms and new controls can be created for the page if that is sought.

In the code behind of the application pages conditional compilation can be used to handle differences. But the layout parts defined in code behind will not be shown in the designer of Visual Studio. The code must be run or debugged in order to see how the code behind affects the layout.

A disadvantage is that it is more complicated and it is easy to make mistakes. If a developer knows the limitations and possibilities with using responsive application design, it can be used to create a partly sharable user interface. This way this part of the user interface must only be maintained from one place, but since it is more complicated to create and maintain it will probably not save the developer any time if the sharable parts aren’t small and well modularized.

In web design the webpage can be tested on the computer by changing the size of the browser window to see how it will look on other devices, while responsive application design demands testing on all different platforms to be sure that the code is platform compatible as well as responsive. Testing is therefore an important part of using responsive design in applications.

Since there are a lot more information and instructions for creating responsive web design, it is much easier to use. If complete instructions, also existed for responsive design in applications, it would probably still be more complicated than using responsive web design, but it would be easier to use, than it is today. For this operating system, it will probably
never be easy to use responsive design the way it is used in this thesis for applications created in C# and XAML.

It makes sense to advance in some future version of Windows to make platforms more closely linked and provide some way of creating sharable applications.

### 7.2 Discussion about the work

Above, the final results of the study were presented and analyzed. The entire work around this thesis is left to discuss.

After deciding what the focus for this thesis would be the literature study and soon after also the development of the Windows Phone 8 application was started. The development of the application was started this soon in order to make sure that it would work to create the application and to use the API for Infor Social Space. It might have been better to first only focus on the literature study about responsive web design, but if there would have been some problems with the application development, then it might have been complicated to circumvent those problems in time. I think it’s good to work against probable errors such as these early in projects.

There were a lot of problems getting the phone emulator to work with Fiddler and to save the state of the emulator. Perhaps it would have been better to ask for help earlier either from co-workers or the Microsoft support. The need for the certificates for connecting to Infor Social Space was a bit annoying. At the beginning I didn’t know about the need for them. These certificates also resulted in a lot of hours of testing, when one of them passed the expiration date without my knowledge. These two problems made the first goal of the application, to write code for fetching and posting data to Infor Social Space, difficult to solve and it took a lot more time than planned.

The idea of first learning about responsive web design and then continue to find solutions for applications created in C# and XAML were rewarding. It worked very well in this project and I think that it would be
harder to search for solutions for applications directly. This is partly because I then would not know what responsive design is. But even if the concepts were known, there would have been more difficult to know where to start. When researching responsive web design, much knowledge about possible components and requirements of responsive application design was found that would probably not be known otherwise. Webpages and application have several similarities such as the same type of visiting devices and also often the same or similar content to present. This makes it possible to share much of the requirements that would not be sharable with a responsive wall that moves or expands. Although the basic requirement is to move to respond to the environment it can be divided into many different requirements depending on field. The knowledge of responsive web design gave a hint of how to describe responsive applications design and what to look for during the research.

It has been difficult to find the correct information when searching the web. Many times when searching for classes and other information about Windows Phone 8, only other results for Windows 8, which are often not applicable on Windows Phone 8, were found. Many of the solutions to the obstacles have been found in the forum on stackoverflow.com and the forum and development pages of msdn.microsoft.se. Often the working part of the solution hasn’t been the top solution of the thread. A lot of the found possibilities were discarded for not working for either Windows Phone 8 applications or Windows Store applications.

In chapter 5.1 **What is Responsive application design?**, responsive application design was defined as a responsive user interface which can be shared between the different Windows 8 applications. Another way of defining responsive application design could be to create one responsive application which could be used from all Windows 8 platforms. This would change the result of the entire study, since no way of creating this one application was found during this study.

---

20 See video Interactive Wall at [http://vimeo.com/4661618](http://vimeo.com/4661618)
In this thesis, a solution was found for how to use responsive design in applications. There might be other solutions, but more time would have been needed to continue the research. If the example application wouldn’t have been developed, more time could have been used for research and for creating more small applications. But then the responsive parts would not have been tested together as one unit. Less testing could mean that it’s more challenging to discover shortages in the chosen solutions and some ideas for solutions can also more easily be missed.

More time could also mean a larger literature study about responsive web design and a possibility to create a web application in html and css3. This study could tell if responsive design for web applications is equal to responsive web design and if these applications can be shared in a greater extent than applications created in C# and XAML.

I am content with the choices made during this project and if I did it again I would probably choose a similar approach.

7.3 Recommendations for future studies

There is still much to be investigated regarding responsive application design. Not all of the controls in chapter 5.2.1 Using sharable elements in XAML layout, have been tested and there might be other properties that must be set for some of them in order to make them responsive. UserControls should also be more tested to see if there are more possibilities in using them then have been found during this study.

There might also be other possible solutions that can be used to create a responsive user interface. One such idea is to use converters. In this project converters weren’t used because they are used as a local resource. This will result in problems in the XAML document since the local namespace is defined differently in Windows Phone 8 applications in comparison to Windows Store applications. But it could probably be used from the code behind. It would also be interesting to learn if a responsive approach without the sharable part would be more advantageous. In such approach a lot of new possibilities opens up when
local resources can be used and controls, properties and events doesn’t need to be similar between platforms.

It might also be interesting to examine how a responsive web application can be created for Windows 8 when html and css3 is used as programming languages. As described earlier in the discussion chapter, it might be interesting to see if responsive design for the web applications is equal to responsive web design.

This thesis is limited to only test the user interface for Windows Phone 8 and Windows Store applications. The user interface have not been added and tested on other Windows 8 devices such as the Windows 8 tablet with the operating system Surface.

If the Windows Phone 8 emulator or Windows Store simulator should be used in future development then a recommendation is to make them run together with Fiddler or some other snooping program early. It’s difficult to know if a web request is actually sent and what have been sent without using such a program.
8. Conclusion

Responsive web design is a technique for making webpages adapt to better fit the screen they are being viewed at. The techniques use a flexible grid to create a structured and fluid layout and make the images and other media flexible to follow the design. The last component is Media queries which can be used to set up different states. In these different states the layout could be specified differently. On small screens unnecessary components can be hidden and the information could be stacked in a single column to become easier to view. In larger screens the columns can be increased and more details can be shown.

The main result of this thesis is that responsive design can be used for Windows 8 applications created in C# and XAML. But it is quite complicated and requires a lot of work to figure out exactly what can and what cannot be done in order for the design to be responsive.

Responsive application design can be specified in at least two ways. The first is to specify responsive application design as a way of only creating one application that can be used on all platforms. No way of creating this one application has been found during this thesis. The other way, which has been discussed in this thesis, is to create a responsive sharable user interface. This clarifies that the user interface should adapt to the size or resolution of the screen and that it can be used on both Windows Phone 8 and Windows Store applications.

I cannot recommend using the full responsive solution found during this study. There are too many limitations and difficulties to overcome. But a flexible approach is to be recommended and if possible a responsive approach within each platform, without the sharable part, would be of good use at least for the computer side, were many variations in screen size exists.
Whether or not responsive application design should be used, a conclusion from this project is that it is possible to use responsive design when creating Windows 8 applications in C# and XAML at least so far as creating a partly sharable responsive user interface.
References

Published Sources

[1] Callahan, Ben & Young, James & Hixon, Jeremy & Knight, Kayla & Nyman, Lewis & Andrew, Rachel & Koblentz Thierry (2012). Responsive design. Smashing magazine, version 1


Unpublished Sources


Visited: 21 Dec 2012. Downloaded: 17 May 2013

Visited: 17 Mar 2013. Downloaded: 17 May 2013

Visited: 19 Nov 2012, Downloaded: 17 May 2013


Visited: 21 Dec 2012. Downloaded 17 May 2013
   <http://www.alistapart.com/articles/responsive-web-design/>
   Visited: 15 Dec 2012. Downloaded 17 May 2013

   Downloaded: 14 Jun 2013

   Visited: 5 Apr 2013. Downloaded: -

   <http://msdn.microsoft.com>

   Downloaded: 30 May 2013

   Downloaded: 30 July 2013

   Downloaded: 20 May 2013

   Downloaded: 20 May 2013
<http://windows.microsoft.com/sv-se/windows-8/meet>  
Visited: 17 May 2013. Downloaded: -

Visited: 16 Nov 2013. Downloaded: -

<http://dev.windowsphone.com/en-us/develop>  

Downloaded: 20 May 2013

<https://dev.windowsphone.com/en-us/design/process>  
Visited: 26 Nov 2012. Downloaded: 17 May 2013

www.responsivegridsystem.com  
Downloaded: 24 June 2013

[25] Sahu, Ashish. *Stretch an image until it’s true size* [e-mail]  
Received: 24 May 2013.

[26] Snook, Jonathan. *Sizing with CSS3’s vw and vh units* (2011) [www]  
<http://snook.ca/archives/html_and_css/vm-vh-units>  
Visited: 17 May 2013. Downloaded: 17 May 2013

*Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification.* [www]  
<http://www.w3.org/TR/CSS21/>  
Visited: 28 Dec 2012. Downloaded: 17 May 2013

<http://www.w3.org/TR/css3-mediaqueries/>  
Visited: 18 Jan 2013. Downloaded 17 May 2013
Appendix 1

1. Sharing techniques
The following chapter contains information about the different sharing techniques which Microsoft recommends for application development. The source to this chapter is *Maximize code reuse between Windows Phone 8 and Windows 8* [13] with subpages.

1.1 Model-View-ViewModel
The Model-View-ViewModel pattern can be used to separate UI and app logics. Microsoft recommends this technique since app logic is more likely to be common for both platforms. Microsoft recommends that the application UI should be tailored for each platform to create the best possible user experience. [17]

This is not in line with the goal of this thesis. Although this thesis is mostly about the user interface it might still be good to separate the interface and the logics.

The following is an image from Microsoft’s webpage that shows the structure of the pattern.

In regard to application development in C# and XAML the views are all the XAML user controls used to set up the interface. Each page of the application is a view. The main app page will contain these user controls.

The View Model is a C# file. This file is connected both to the views and the Model and forms a bridge between them. The View Model could preferably be initialized right after the initialization of the main page.
The Model is all the apps logic. The model files are the rest of the C# files which are not directly connected to the user interface that handles the functionality of the application.

The Model notifies its views when properties are changed and the View Model uses an observable collection to forward the notifications to the views.

1.2 Portable class libraries
Portable class libraries can be used for both Windows 8 and Windows Phone 8 because they share the same .NET Framework engine. When a portable class library is created the target platforms are specified and the library can be used for all those platforms. The library is then added as a reference to the projects that will use it. Not all code can be portable. The code needs to be managed\(^\text{21}\), no conditional compilation is allowed, no usage of Windows Runtime APIs and no UI constructs. The user interface code needs to be compiled separately for each platform, so it cannot be shared using portable class libraries.

Portable class libraries can be used together with the Model-View-View Model pattern. Both the View Model and the Model can be placed in portable class libraries if the code follows the rules of the portable class libraries.

1.3 Add as link
Code can be shared by adding the same file as link in the projects for both platforms. This is an alternative to use when the code cannot be added as a portable class library. This code is compiled ones for each platform. Any code that can be isolated and is platform independent can be used in both applications using this technique.

---
\(^{21}\) Managed code is a Microsoft expression for code that can be executed in another execution environment then the computer called Common Language Runtime virtual machine.
The *add as link* technique can also be used for the user interface. Much of the XAML code is platform independent for Windows 8 and Windows Phone 8 and can therefore be shared using this technique.

This technique is therefore good to use when trying to create a platform responsive user interface.

### 1.4 Windows Runtime Component

Functionality in Windows Runtime Components can be used in applications of any of the supported languages for the platforms. There are some differences between the platforms which language can be used to create and run the Windows Runtime Component, for the phone only C++ is allowed in creating the component while also C# and visual basic can be used for computer applications. In this project, the focus is to only use C# and therefore this technique will not be applied. The technique could be of use since both platforms supports to use the components in C# applications, but it isn’t in the scope of this project.

### 1.5 XAML User Interface

The XAML code is not portable between the Windows 8 and Windows Phone 8 platforms. Both Windows Phone 8 and Windows 8 support user controls. By creating the user interface parts in user controls and removing all platform specific code those parts of the user interface can be shared and compiled for each platform by using the “add as link” technique.

#### 1.5.1 Why Microsoft doesn’t recommend sharing User Interface

There are several difficulties with sharing the user interface between Windows 8 and Windows Phone 8. A lot of XAML code is similar for both Windows 8 and Windows Phone 8, but it is often implemented specifically for each of them. The namespace prefixes differ and conditional compilation cannot be used to choose one or the other.

Microsoft also gives some solutions to circumvent these obstacles, such as creating the User Interface from C# code during initialization instead of using the XAML document and load platform-specific code at runtime from resources and add it to the page by injecting it as a string.
Much of the XAML code can be reused, but local resources cannot be used easily from the XAML document since that requires a local namespace being set, which is done differently in Windows Phone 8 and Windows Store applications. Here is an example of how the namespace SharedApp.Shared is specified in Windows Phone 8 and Windows Store applications.

```xml
xmlns:shared="clr-namespace:SharedApp.Shared"
(For Windows Phone 8)

xmlns:shared= "using:SharedApp.Shared"
(For Windows Store apps)
```

### 1.6 Conditional Compilation

Conditional compilation can be used to only compile the code that is intended for each platform. This way code that can be built on one platform is ignored by the compiler on that platform. The conditional compilation is written as an #if-statement as in the textbox on the right. The statement is `NETFX_CORE` for Windows 8 and `WINDOWS_PHONE` can be used for Windows Phone 8.

Conditional compilation cannot be used in portable class libraries or in XAML code. Sometimes it might be better to instead isolate the code in another class.

### 1.7 Partial classes

If a class contains both specific and independent parts of code partial classes can be useful. The platform independent code is added to a class defined as partial. This class is added as link to the projects. A new class with the same name and with the keyword partial is added to both projects to handle the platform specific code.

### 1.8 Inheritance

Another way of handling differences and similarities is by defining a parent class which contains all shared functionality and defines the platform
specific functions as abstract. This class is added as link to the projects. In the projects the class is referenced by adding a class which derives from the abstract parent class. These project classes must implement the platform specific abstract methods of the parent but can do so differently with no regard to the other platform.

1.9 Interfaces
A useful ability of interfaces is that its methods can be called without knowing the implementations. This way an interface can be created which declares the specific methods but doesn’t provide the bodies. A sharable class then can take in an interface of this type as an input parameter and store it as a local variable. From this variable the interface methods can be called. In both projects the specific implementations of the methods are added to a new class that implements the new interface. When the sharable class is initialized the specific class is sent as a parameter, which makes the method accessible.
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Sofi Klockare