Architecture patterns in web applications and implementation of find local food
I would like to thank people that contributed in this thesis.

My supervisor Mr. Tobias Andersson, for all his support,

Mr. Mathias Hedenborg for his invaluable guidance and reviewing this thesis report,

Mrs. Diana Unander Nordle for her help in finding the thesis project in a company,

Jan Melkersson coordinator in Ljungby municipality and

Last, but not the least, my family with their support that always shows to me.
ABSTRACT

Nowadays there are different forms and technologies for implementing web applications without much concern for their architectural model. Without a good and stable architecture those applications may fail during their operations. Thus, this thesis project is focused on exploring different architectural models for creation of web solutions. Moreover, this thesis project is also part of the collaboration between Linnaeus University and Ljungby municipality with the aim to create a system that grow up local economies by showing the local producers’ locations, providing customers with different information about the products etc.

The result of the thesis shows that an application can be developed in various architectural models. The choice of one architectural model over another depends on the project needs. I found that by using ASP.NET MVC framework there is more separation of concerns, easy maintenance of the code, and unit testing for an application. Search engine of find local food system can be used to search for different local producer’s location and their services.

KEYWORDS: ASP.NET MVC, REST, TIER, LAYER, ASP.NET WEB FORMS,
List of figures

Figure 2.1-Phases for building an application[3] ..........................................................4
Figure 2.2-Logical View of web application [2] ..........................................................7
Figure 2.3-Components interaction in web environment[2] ........................................8
Figure 4.1-3-Tier architecture ......................................................................................11
Figure 4.2-Application tiers[3] ................................................................................12
Figure 4.3-N-Tier architecture[3] ...............................................................................14
Figure 4.4-MVC architecture[41] ..............................................................................14
Figure 4.5-HTTP request from Client to Server [28] .................................................15
Figure 4.6-Server interactions with business logic tier[28] ........................................16
Figure 4.7-Server responses as HTML output to Client[28] ........................................16
Figure 4.8-MVC seperations at Server [28] ...............................................................17
Figure 4.9-Complete MVC architecture in web applications[28] ................................18
Figure 4.10-Factors before making the choice[45] ....................................................20
Figure 5.23-Example of DetailsView code .................................................................21
Figure 5.24-Example of code logic in class model and controller .............................22
Figure 5.25-URL syntax in ASP.NET web forms ..........................................................21
Figure 5.26-URL syntax in ASP.NET MVC .................................................................23
Figure 5.27-The newsletter box example with textbox and button .........................22
Figure 5.28-Code in aspx page and behind the page, ASP.NET web forms ..............24
Figure 5.29-View source code page,ASP.NET web forms .........................................25
Figure 5.30-HTML code and code logic for newsletter box ......................................25
Figure 5.31- View source code page,ASP.NET MVC ................................................26
# Table of Contents

1. **INTRODUCTION** ....................................................................................................... 1  
   1.1 Problem ................................................................................................................... 1  
   1.1 Structure of the thesis ............................................................................................. 1  

2. **SOFTWARE ARCHITECTURE** .................................................................................. 2  
   2.1 What is Architecture ............................................................................................... 2  
   2.2 What is Design ....................................................................................................... 4  
   2.3 Architectural Style .................................................................................................. 5  
   2.4 Architecture in web environment ........................................................................... 4  
   2.5 Architecture views in web applications .................................................................. 7  

3. **LANGUAGES IN WEB APPLICATIONS** .................................................................... 10  
   3.1 Languages ............................................................................................................. 10  
      3.1.1 PHP ................................................................................................................ 10  
      3.1.2 Perl ................................................................................................................. 10  
      3.1.3 ColdFusion .................................................................................................... 10  
      3.1.4 JSP ................................................................................................................. 11  
      3.1.5 ASP ................................................................................................................ 11  
      3.1.6 ASP.NET ....................................................................................................... 11  

4. **ARCHITECTURE IN ASP.NET** ............................................................................... 12  
   4.1 Tiers and Layers ................................................................................................... 12  
      4.1.1 What is a Tier ................................................................................................ 12  
      4.1.2 What is a Layer .............................................................................................. 12  
   4.2 1-Tier 1-Layer model ............................................................................................ 13  
      4.2.1 1-Tier 2-Layer model ..................................................................................... 14  
      4.2.2 1-Tier 3-Layer model .................................................................................... 14  
   4.3 2-Tier model ........................................................................................................ 14  
   4.4 3-Tiers model ....................................................................................................... 14  
   4.5 N-tier model ......................................................................................................... 14  
   4.6 Model View Controller (MVC) ............................................................................ 15  
   4.7 MVC in web applications .................................................................................... 16  
   4.8 ASP.NET frameworks .......................................................................................... 19  
      4.8.1 ASP.NET web forms .................................................................................... 19  
      4.8.2 ASP.NET MVC ............................................................................................ 20  
      4.8.3 Making the choice ASP.NET web forms or ASP.NET MVC ...................... 20  

5. **IMPLEMENTATION OF FIND LOCAL FOOD** ...................................................... 21  
   5.1 Home page ........................................................................................................... 21  
   5.2 Find local farms page ........................................................................................... 21  
   5.3 Market place page ............................................................................................... 21  
      5.3.1 Database ........................................................................................................ 21  
      5.3.2 Administration page ...................................................................................... 21  
   5.4 Seasonal news page .............................................................................................. 21  
   5.5 Newsletter page .................................................................................................... 22  
   5.6 About us page ....................................................................................................... 22
6. CONCLUSION AND FUTURE WORK .................................................................28
   6.1 Conclusion.................................................................................................... 28
   6.2 Future work .................................................................................................. 28

7. REFERENCES ..........................................................................................................29

APPENDIX A ............................................................................................................... 32
   A.1 Home page ..................................................................................................... 32
   A.2 Find local farms page ..................................................................................... 32
   A.3 Market place page .......................................................................................... 35
   A.4 Database & Administration page ..................................................................... 36
   A.5 Newsletter page .............................................................................................. 40
   A.6 Contact us page .............................................................................................. 41
1. INTRODUCTION

With the invention of Internet and web, many web applications nowadays do not follow the principle of client/server interactions where every application consists of its own user interface installed separately in computers; instead these applications interact through web browsers, web servers and application servers which is one reason that makes them more popular. The common samples usage of web applications includes: entertainment, online sales, webmail, government, e-commerce etc [2]. According to [1] these applications followed an ad hoc systematic approach which could bring them to an unsuccessfully development, deployment and maintenance because of the complexity and sophistication of them. Poorly approaches of development applied on these applications can cause them to a not intended result, failure during their operations, not reusable for the future changes etc [1, 2].

This thesis report will go through different approaches and architecture that can be used for developing web applications. By considering the approaches and architectures described in the coming chapters, we will ensure in scalability, maintainability and reusability for those applications.

1.1 Problem definition

The focus work in this thesis is to compare architectural models for creation of web application and implementation of find local food system. The aim of this system is to grow up local economies by showing information and location of different local producers and their product services. For development of this system I will use ASP.NET language. Then we will see if it is possible the same system find local food to develop in both two frameworks such as ASP.NET web forms and ASP.NET MVC. Moreover, for a project to be extensible for future work, easy maintenance of the code, loosely coupled, rapid development of the project, unit testing for every component in the project, search engine friendly URL, and clear separation of concern of the code etc, are some of the factors that will be discussed in this thesis and which will help before deciding which framework to use for better solution of the project.

1.2 Structure of thesis

This thesis report contains the following structure: Chapter 2 briefly get into software architecture by describing and defining architecture, design and the relationship between them. It also describes architectural styles and architecture for web environment by categorizing different views used in web applications. Next, chapter 3 describes different languages used nowadays for creation of web applications by showing their strength and weakness. Chapter 4 begins with describing architecture of ASP.NET and show how it can be separated into tiers and layers. This chapter will also get into Model View controller (MVC) design pattern and describe how this pattern can be used for web applications. The overall implementation of “find local food” project with some screenshots put in appendix page is described in chapter 5. Conclusion and future work describe in chapter 6 close discussion of this thesis report.
2 SOFTWARE ARCHITECTURE

There are different tools and platforms that can be used for building an application. However, these tools and platforms may not decide for designing your applications based on the requirements and specifications. Without good consideration of the key requirements, design for any particular problem etc, can lead your application to fail. This risk can be caused because of not good understanding of architecture which involves: unstable, difficulty on deployment and hard management for software [38].

2.1 What is architecture?

Software architecture is a “blue print” for an application that most developers and architects must understand before starting creating any application [3].The definition given from Shaw and Garlan 1996 define software architecture as [4]:

“Software architecture encompasses the set of significant decisions about the organization of a software system including the selection of the structural elements and their interfaces by which the system is composed; behavior as specified in collaboration among those elements; composition of these structural and behavioral elements into larger subsystems; and an architectural style that guides this organization. Software architecture also involves functionality, usability, resilience, performance, reuse, comprehensibility, economic and technology constraints, tradeoffs and aesthetic concerns.”

There are some more issues that are needed to have in consideration for a system: it is important to understand the relationship between these components and connectors and trying to find a way of building new system from the variations of different older systems, there is a risk that components may change over time as long as requirements change, but having a stable architecture in place lead to a successful software system and brings possibility for engineers’ to make different choices in the process of design, it is essential for large and complex system to have an architectural representation etc [5].On the other hand, when building architecture for the system there are several architectural points that are necessary and considerable for software architecture: [6]

- Multiple Views
- Architectural style
- Style and engineering
- Style and materials

However, this thesis will not get into all these points but there will be a description for some of them in relation with web applications’ architecture.

2.2 What is design?

While software architecture is a “blueprint” for an application, design that comes after architecture is a blueprint for the implementation phase [7]. During this phase designers choose the architectural pattern and the methodology platform that they are going to use for solving a problem. Figure 2.1 shows phases for building the application and how the architecture is related to design and business requirements [3].

2
The following definition is given for those phases [6]:

- "Requirements are concerned with the determination of the information, processing, and the characteristics of that information and processing needed by the user of the system."

- "Architecture is concerned with the selection of architectural elements, their interaction, and the constraints on those elements and their interactions to provide a framework in which to satisfy the requirements and serve as a basis for the design."

- "Design is concerned with the modularization and detailed interfaces of the design elements, their algorithms and procedures, and the data types needed to support the architecture and to satisfy the requirements."

- "Implementation is concerned with the representation of the algorithms and data types that satisfy the design, architecture and requirements."

For simplifying understanding, let us consider a simple analogy of building a house. Usually there are some phases for building the house: first before construction of the house starts the company that is supposed to build the house must know the requirements from the customer. After the requirements are known from the customer the architect begins with design of architecture until the customer’s needs are fulfilled. The architect may have options to choose from different style (i.e. if the house will be build as one floor or two floors, if it will be residential or vocational etc.). Once the architect is done with its job the architecture is passed to the company which means that the architect does not care anymore how the company is going to build the house or what kind of tools is going to use. Put it simple, the architect’s job is just to implement the architecture, and the company which is supposed to build the house must follow that architecture during the building phase. So, the company begin building the house by following the architecture from the architect and once it is done the customer is ready to use the house according to his/her needs [7].

Similarly, the project starts by considering the requirement needs and then it is responsible of the architect to come up with an architecture application that fulfils these actual requirement needs. After the architecture is known the design phase begins by following the actual implementation of the architecture for the project. During this phase the methodology framework is chosen that is supposed to be used in the development phase and involves code programming in a project. Summing up, the role
of architecture is just to fulfil the requirements specification and it does not go deep into how the implementation for the project is going to be implemented. The responsibility during design phase is to choose the implementation methodology that is supposed to be used during the development phase (i.e. if the project is going to be implemented through Java, PHP, or ASP) [3].

2.3 Architectural Style

As architecture definition for a system is described as a collection of elements and the relationship among these elements—connectors, the architectural style is an abstract description of these elements, connectors and interactions among them[5,6]. Garland and Shaw define an architectural style as: [8]

“family of systems in terms of a pattern of structural organization. More specifically, an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined. These can include topological constraints on architectural descriptions (e.g., no cycles). Other constraints—say, having to do with execution semantics—might also be part of the style definition.”

It is important to understand the concept of style because by having a good understanding of style we can make sure that the architecture met the requirements; we can compare any architectural style versus another etc [8].

Moreover, there are numerous architectural styles that can be used for suiting individual needs in an application. The following shows some of the architectural styles [7]:

- N-tier
- Pipes and filters
- Object Oriented
- Layered System
- Client server
- Data centric etc.

For example, in object oriented style, here the object that encapsulate data and operations represents the elements while message passing represent the connectors as a mechanism for communication and corporation among elements. In client server style here the client and server are the main elements where client make a request while server response the request. N-tier is another style that is used in distributed systems. This style consists of client element, server element and middleware element but the description for this style will be explained in coming chapters in the context of web applications [3, 7, 8].

2.4 Architecture in web environment

Web applications are distributed applications that we can find everywhere nowadays and are becoming part of our life. Just few examples to mention: we find web applications in online transaction where we order and purchase things, in schools, libraries, governments, entertainment and other parts of industry. These applications are part and exist under umbrella of World Wide Web (WWW) with the aim to share and present information. If we compare these applications with traditional client/server applications these applications are more popular since they can operate in different platform and as a client interface use web browser that is accessible from every operating system with no need for any additional software installation [1][2].
However, even though these applications bring a lot of benefits to users there is a risk that they can fail due to not concerning of the development process and maintainability of these applications. According to [1] “the development process for web application has been an ad hoc and has lack a rigour and systematic approach”. The lack of development process of these complex and sophisticated applications can cause web crises.

On the other hand, “Berners-Lee[1996] wrote that the Web’s major goal was to be a share information space through which people and machines could communicate located in different parts of the world”[11].

But, when people communicate and share information such as videos, images, text, document etc, there must be a system that provides a consistent interface for this structured information. There also must be an architecture that concerns the following issues: how to minimize latency in network interactions, extensibility of the system is another issue because of requirements changes overtime web must be prepared for these changes, security in architectural elements and the platform they operate, existing architectural elements must be designed for additional features added later and easy deployment [2, 11].

The issues above motivated the development of Representational State Transfer architecture (REST) as a foundation for the modern web architecture. It represents abstractions of architectural elements in distributed hypermedia system and how modern web applications should work. Perry and Wolf distinguish three classes of architectural elements of REST with constrains upon these elements that define the bases of web architecture [1, 2, 10, 11].The following elements include:

Data elements
Data elements are the primary concern of the REST. Description of these elements will be in relational with other object style. In REST information moves from the location where is it is stored to the location where it can be used which is in contrast with other distributed paradigms [2, 11].

Architect in distributed hypermedia can choose three options during data interactions: first render that data where it is located and send to recipient (client/server paradigm), second, hide the data with render engine and send to recipient, third, the recipient can choose its own render engine if the data is sent as a raw data along with data type description. However, there are some disadvantages of using the above options, in the first option of client/server paradigm even that the client encapsulate the data and make the client easily, the server side face the problem of scalability. In the second options the amount of data transfer may increase because of the functionality of the render engine that may be not know to recipient. In the third options, the data type is required to be understood between sender and recipients [2, 11].

REST architecture tries to solve these issues in which components communicate by transferring the data according to format matching and selection of data based on request of recipient. Moreover, REST provides a generic interface for encapsulation of information and it does not face the problem of scalability in the client/server paradigm [2, 11].

Connecting Elements
An abstract name for information in REST is designate as “resource”. A resource can be any document, file, object etc. On the other hand, components in REST use “resource identifier (URL)” to identify the interactions. When the component receives a resource it performs some actions by using “representation” (i.e. html document, object, file image) to get the set for that resource and transfer that representation among all other
components [11]. What about the encapsulation during the interactions between the
components? There must be a mechanism that encapsulates the activities when a
resource is received and when a representation is transferred. Thus, REST use
connectors as an abstract interface to handle this encapsulation by hiding the
implementation of the resource and communication. There are different types of
connectors used in REST [11]:

- Client
- Server
- Resolver
- Tunnel
- Cache

However, the main connectors are client which make the request in communication and
server which response to that request [11, 2].

**Processing elements**

REST use different components and categorize them according to their roles they have
in an applications. The following components include:

*User agent*- An example of this component is Web browser that gets access to the
information and renders that information depend on the need for an application. To
access and render the component uses a client connector.

*Origin server*- this component acts as a recipient for a resource and is responsible for
representation of the intended request made by the client. To access the representation
this component uses the server connector.

There are other types of components used in communication in REST architecture
which acts as intermediary components. Such components include: “proxy” and
“gateway” [11, 2].

Furthermore, REST architectural style since 1994 is also used as guidance for
building and deployment of architecture for web applications. The applied REST in web
brought some benefits such as: it helps to identify any particular problem, compare and
find alternate solutions for a problem, make sure that there would not be any violation
on constrains for any protocol extension etc [39].

In the beginning REST is referred to as “HTTP object model” but this name leads to
misunderstanding since the behaviour of the user in application involves with selecting
a link and then result in redirect to another page. Thus, REST is known as
“Representational State Transfer”. Moreover, REST applied to the most important
elements in web architecture and as a mechanism for communication between
components in web interactions: URI (Uniform resource identifier, now known as
Uniform Resource Locator, URL) and HTTP (Hypertext Transfer protocol). In
hypermedia systems we saw that REST used resources to identify the interactions
among the components, whereas in web architecture URI is used as a resource [39].

On the other hand, the applied REST in HTTP help to identify the following issues:
the performance of web applications that are based on HTTP, any intention for
developing a new protocol, limits for any HTTP extensions for those that already exist
in the architectural model etc [39].

2.5 Architecture Views in web applications

“As Philippe Kruchten observed that complex system cannot be understood just from a
single viewpoint” [12], web applications can also be seen as four views namely as:
logical view, development view, process view and physical view. Each of these views
cannot be fully independent but must have a relation in order to have a good understanding for the applications [2, 12].

**Logical View**

This view is an abstract view and is concerned with the structure of the application by identifying the components and the relationship between these components without considering the details implementation of each component [2]. Web applications can be comprised of tiers. An example is 2-tier and 3 tier model which is depicted in figure 2.2

![Logical View Diagram](image)

As can be seen from the figure, 2-tiers and 3 tiers have the presentation logic which concerns in preparing the User Interface and facilities the users interactions. Components in this tier communicate with the business logic tier which contains all the logic functions for manipulating or modifying data in database tier. In contrast to 3 tier model, 2-tier mixed the business logic and database code into one tier, so any changes made in business logic may affects changes in database. On the other hand, 3-tier model brings more separation of concerns between tiers since presentation, business and database components are independently separated into tiers such as changes made in any tier would not affects changes in other tier [2, 13]. In coming chapters when the architecture for ASP.NET will be discussed, there will be more description about separation on tiers and layers.

**Development View**

This view is concerned with the actual implementation of the application. Moreover during this view there are some other details that are considered essential to take in consideration: structure of links between pages which is an important detail because as the web grows in size the interactions between the pages can cause to send visitors to an unwanted page without providing them with the content they navigate through. In order to avoid this issue links must be controlled regularly [2, 14].

Another aspect is the session management technique. Since web applications use the HTTP as a ”stateless” protocol for communication between client and server there must be a way for saving the states (sequence of commands) or maintaining the information about a session (sequence of transmitted commands) in the interactions. Web applications use ”cookies” as a mechanism for storing states and maintaining the sessions. A simple example where cookies are used can be found in e-commerce site such as shopping cart where you add items and how the site remembers the added items. How this can be achieved is HTTP contains an agreement during the transmission of
message, agreement this through two categorizations of cookies: 'Set-Cookie’ and ‘Cookies header’. Set-cookie header is a response from the server to the browser and contains two attributes: name/values. When the request is made to the server it forces the browser to create a cookie header and save the attributes so whenever the next future request is made to the server the cookie header has already the attributes saved[2,14].

Another concern of this view is the implementation technology where developers can choose whether “programming approaches” (i.e. Java servlet, CGI, Perl) that deals with programming logic on pages or “template approach” (i.e. Cold Fusion, JSP) that deals with structure formatting on pages[2,14].

**Physical View**
This view is concerned with the mapping of components from other views to the real technology [2]. Web application consists of the following physical components that are depicted in the figure and shows how these components interact in order the application to get executed.

![Figure 2.3 Components in web interactions](image)

As can be seen from the figure web applications contains web browser as a component that users are able for making a request to some services in the web server (i.e. user’s request can be the search of any file, document or image in a database). Then the browser task is to get users’ request and transmit it via HTTP protocol to the web server. Further, if the user’s request is made directly to the database then the web server must contact the application server which is supposed to retrieve data by executing any SQL queries and send it to the database for getting the request user’s result. After that when the web server gets the result from the application server it returns back as HTML page to the web browser [2, 9].

**Process View**
During the interactions of components in a system, usually there is a need for additional components. Thus, this view is concerned with scalability and performance of components in system interactions [2, 11].
3. LANGUAGES IN WEB APPLICATIONS

In this chapter we will explore some of the languages used for creating web applications. Then we will focus in language that “find local food” project is implemented by describing its architecture and frameworks.

3.1 Languages

Before choosing a programming language it is important to have the following points in consideration: what will be my server platform that is going to be used for the application, does the application contains the database and what will be, what about the experience in the field of programming, what will be the operating system that sometimes plays a key role before choosing a language etc. By having in consideration the mentioned above it is much easier for making the choice for right programming language that best fits the application. Nowadays, web applications may be implemented in various scripting and server side programming languages. The most popular languages used in industry are PHP, Perl, ColdFusion, JSP, ASP.NET etc that the following subsections is going to describe them in details [15]. For my project “find local food” I will use ASP.NET language which is a part of .NET platform and it has a great environment development. With you of ASP.NET I will have alternative to choose between programming languages either C# or VB.

3.1.1 PHP

PHP stands for “PHP: Hypertext Preprocessor” and is a server side scripting language that is used for creating dynamic web pages. Its structure code in page is embedded with PHP tags and HTML tags so whenever users open the page the server process an output and return to the browser as HTML code not as script code which prevents the issue of stealing the code. It supports numbers of database such as: MySQL, Oracle, ODBC, Sybase etc and can easily work with others environments. There are some strengths and weakness of using this language [16, 17]:

Some strengthens includes:

- It is open source language. You can use it for free.
- It runs in the client interface without need for program compilation or any special development environment or IDE.
- It supports and runs in different platform environment.

Some weakness includes:

- Case sensitivity
- Error handling

3.1.2 Perl

Perl stands for Practical Extraction and Report Language and is an open source and interpreted programming language with a huge numbers of tools and libraries that make possible for web developer to create dynamic web applications. The weakness of this language is slowness because of its implementation as interpreted not compiled language. The strength of it is simplicity, elegant and during implementation of any web site [18, 19].

3.1.3 ColdFusion

ColdFusion is a server-side programming language that is used for rapid creation of dynamic web applications. Compared to traditional static web site, ColdFusion contains
ColdFusion Markup language (CFML) in addition to HTML. By using ColdFusion you can easily perform different tasks such as accessing the database, querying the database, data manipulation within database, accessing server mail, accessing any other objects etc. The strength of this language is that it is easy to program and understand because of its CFML based-tag like HTML, it is compiled language which makes it faster compared to interpreted languages. As a weakness of this language is that it may not work well for non-windows platform for larger project that involves the problem of performance, scalability and stability [14, 20].

3.1.4 JSP
JSP stands for Java Server Pages and is an open source programming language supported by Oracle. You can develop rich web applications without a need to know java Script which is strength of this language. Compared to other web development languages this language is extensible that allows java tag library developers to add tag handlers [18, 19].

3.1.5 ASP
ASP stands for Active Server Pages and is a side-server scripting language that is used for creating interactive web applications. Interpreted JavaScript and VBScript languages are only languages available in ASP. When the request is made from the browser the web server calls the asp page and renders the process of coding from top to bottom for executing any script functions. After that the web server prepares a content to send to the browser as a response. Pages in ASP combine a mixture of code such as XML, HTML, JavaScript, CSS etc which are weakness of this language due to the fact of difficulty of maintaining these pages. Another weakness of this language is that it runs under the Internet information Service (IIS). So whenever IIS needs to restart or stop for any reason, can cause pages in ASP to crash [21].

3.1.6 ASP.NET
As the web applications increase in size and because of the new technologies and tools emerged, it is hard nowadays for classic ASP programmers to develop maintain efficient web applications. This is because of the lack of tools for debugging which leads programmers in troubleshooting code due to the limitation of interpreted JavaScript and VBScript coding. Another aspect would be a mixture of “spaghetti” code that ASP comprises inside of HTML tags. Thus, Microsoft came with .NET technology that provides numbers combines components for build web applications. ASP.NET is the part of this technology that brings flexibility for web developer to create complex web applications based in the core concept of Object oriented programming model. There are a lot of advantages of using this technology compared to others: its code is compiled not interpreted that makes the execution of pages faster without performance problem; it supports the feature of object oriented such as: reusability of the code, encapsulation, inheritance etc; it gives the ability to code in strong typed languages such as C# or Visual Basic (VB) because of the Intermediate Language (IL) that .NET contains; deployment of the application to server is easy by just copy all the components assemblies and the files of the website; etc. A drawback of this technology is the fact that it works only in the windows platform. Since, my project is going to be build in this technology language, the coming chapter will get deep on details by considering its architecture and frameworks [22, 23].
4. ARCHITECTURE IN ASP.NET

Before starting implementation for an application it is really important to consider its architecture. There are still some developers that use ASP.NET and just try to jump in coding without any consideration for application architecture. By having a good architecture in place we enhance scalability, maintainability and reusability without facing any problem later on for the application. [40]

The following subsection will describe how ASP.NET is divided into tiers and layers and which architecture model best fits for the given web application.

4.1 Tiers and Layers

The following subsections are going to describe the meanings of tiers and layers.

4.1.1 What is a tier?

“A tier is a unit of deployment.”[3] The meaning of tiers is to physically separate the code in different assemblies and possibility of code deployment in distributed manner.

4.1.2 What is a layer

“A layer is a logical separation of responsibility within the code.” [3] The meaning of logical separation is to organize the code of the application in different namespaces but under a single physical tier. [3]

4.2 1-Tier 1-Layer model

In general, web based applications consist of the following architecture:

![Figure 4.1 3-Tier architecture](image)

The client tier is a web browser that is used for rendering the result from other tiers. The application tier is where business logic code, data access code is defined. This tier consists of the following elements: User Interface (UI), Business Logic (BL) and Data access layer (DAL). Database tier is primarily concern for storing data into tables, updating information data stored in tables etc. This form of architecture for web based application is known as 3-tier architecture [3]. For now let’s focus only on the application tier and see how this tier in ASP.NET project can be divided into further
layers just to introduce loose coupling between layers. So, the application layer is depicted as in the figure 4.2:

![Application Tiers Diagram](image)

Here the presentation layer contains all the controls, pages (i.e. aspx, ascx), master pages, CSS that can be used for creation of User interface of the project etc. Business logic layer contains class libraries or functions that perform the business rules. This layer is really important and thus is important not to mix the code on it. Data access layer has its methods for accessing the data in data layer. Through data access layer data can be queries, updated, deleted in database. Example of tools used in this layer for accessing the database includes: DataSet, LINQ to SQL etc [3].

Now if we separate the components in their own tier we will have 5-Tier architecture with include of client tier. However our focus is only in the application tier for our ASP.NET project with aim to break the code in separate tiers and layers in order to achieve loose coupling. So, 1-tier and 1-layer exist under one tier where all the presentation, business logic and data access layer code is mixed. This model present tightly coupling between layers such as changes made in any layer affects all other layers. As an example consider classic ASP where all the HTML and JavaScript code is mixed in UI without having any logic or physical separation of the code. This model can be used for simple applications that are not intended to be reusable in the future. [3]
4.2.1 1-Tier 2-Layer model
Compared to 1-tier 1-layer model the User interface (UI) and business logic code in this model are logically separated in different folder under the same tier. This model has more separation of concern since the presentation layer code is not mixed with the business logic code which is in contrast with classic ASP where all the code is mixed. However, the business logic and data access code still remain in the same layer. This model can also be used for small and simple applications [3].

4.2.2 1-Tier 3-Layer model
Compared to 1-tier 2-layer model the business logic and data access are logically separated in different folder under the same tier. This model has clean separation of concern since there is no mixed code between layers. However, this model is not intended for scale large project applications and cannot be thought for deployment of layers in distributed manner [3].

4.3 2-Tier model
This is another model where the business UI code and business logic code are separated in physical tier. This physical separation is more scalable compared to other model mention because changes made in any tier would not affect any changes in another tiers. So, the project by using this model will contain UI code in one assembly and Business logic and data access code in another assembly. This model can be used for larger scale application but still can be problematic since the business logic and data access code remains in the same tier. Any changes made in data access code affect changes in business logic [3].

4.4 3-Tier model
We saw that 2-tier model contains the business logic and data access code in the same tier. But there is a need especially for large and complex applications that the business logic and data access code not to be mixed. Thus, 3-tier model separate every components such as: User interface (UI), business logic, and data access in independent physical tiers. This physical separation into tiers makes easy for developers to maintain the project and work independently in every tier. Moreover, by using this model there can be a replacement of any components in project without need to recompile the whole project and without any affects in other components [3].

4.5 N-Tier model
In the above subsections we were focused in the application tiers for our project in ASP.NET. We separated the components into tiers and layers and we saw the advantages of using 3-tier model. For that moment we skipped the database tier and presentation tier just to be clearly during the separation of main application into layers. If we include now the database tier and presentation tier we will have 5-tier architecture which will contain: presentation tier, User interface tier, business logic tier, data access tier, and database tier physically separated. Let just emphasis something here why we are separating those components in different physical tiers. This is because we want that our project to be scalable, loose coupled, maintainable, and distributed for future work. However, this separation of components into layers and tiers can cause performance problem for an application due to the fact that it is faster for components to run in one tier rather that separating them in distinct tiers. But this all depends how the architectural model is constructed. If the physical separation on tiers is constructed to run in single machine then we will fetch the problem performance, but if it is constructed to work in distributed manner then there will be an increase performance for
an application [3]. The N-Tier model is another architecture that is used for complex and large applications. In this model different components (physical separated in tiers) are distributed in multiple machines. The following diagram depicted in figure 4.3 shows how the components are distributed.

![N-Tier architecture](image)

As can be seen from the figure every component is separated in its own machine in communication with other components. This model best fits for larger complex applications (i.e. shopping cart). Moreover, it is important to know when to use this model for a project. All this depends on the project needs, for instance if we have to create a simple project like “guestbook” [44] it is better to choose other models rather than choosing this model because of its complexity and expensiveness. Summing up, choosing the right model for an application depends on the project needs [3, 24].

4.6 MVC design pattern.
The architect Christopher Alexander emphasize that architectural design has its own pattern:
“Each pattern describe a problem which occurs over and over again in our environment, and then describe the core of the solution for that problem, in such as way that you can use this solution a million times over, without ever doing it the same way twice.[25]”

Model View controller is another architecture that was first used in Smalltalk in 70’s. “Since that time, the MVC design has become common place, especially in object oriented systems [26]”. Rapid common use of this architecture comes also in ASP.NET that helps developers to avoid the mixing of the presentation layer with the domain logic. The diagram model of this architecture is depicted in figure 4.4:

![MVC architecture](image)

Controller- as can be seen from figure the controller is responsible for receiving any input from the users and make calls to the model and view for updating their data. As
an example think of controller as a police in the traffic lights that is supposed to control the directions of the cars [41].

*Model-* the model represents the domain logic for an application. All the data information that is needed for an application goes inside the model. As example, think for the model as a logical code of the classes and functions [41].

*View-* We said that model contains the data information that is needed for the application. This data should be displayed in any way to the users. Model by itself does not have a mechanism for displaying data. Thus, view is for rendering and display the content of model to the users. As an example think for the view as html code that is display to the browser [41].

### 4.7 MVC in web applications.

In the previous section we described model view controller and its elements. In this section we will see how this architecture can be used in web applications when a request as HTTP is made from the browser and received from the server. The figure 4.5 shows this interaction [28]:

![MVC Architecture Diagram](image-url)

As can be seen from the figure the client which is a browser makes numerous HTTP requests to the server. These requests are GET and POST request that are inside of HTTP request [28]. When the server receives a request it accesses the business logic tier which contains the code logic and access the database for getting the result of intended request. The figure 4.6 shows this interaction:
When the server gets the result from the business logic with access to database then it is send that result to the client as HTML page. The figure 4.7 shows this interaction:

Moreover, when a request is made from client to server as HTTP request (GET and POST) events there are some additional parameters added on those events in order to identify the message for every request. This mechanism is widely used in web application but is rather complex as for every event there is a need for additional parameter attached to it. Model View controller is an architecture that facilities the events issues [28]. The figure 4.8 shows MVC in web interactions:
Compared to traditional interaction in web application this architecture contains controller that is responsible for getting input from the users and then contact the business logic tier (Model) and view for any actions based on the input request from the users. Its responsibility is to get the users inputs contact model and then represent data to an appropriate view. On the other hand, the model is there to represent the data and behaviour of data. Its responsibility is just to notify controller and views for any changes made in data without any other knowledge about them. View is there to represent the graphical output as HTML pages. Its responsibility is to display content of the model. The purpose of this architecture is to separate the model from the view such as changes made in model would not affect any changes on views or there may be added other views without any changes made in model [28, 29]. The complete architecture of web interactions with MVC is depicted in figure 4.9.
4.8 ASP.NET frameworks

ASP.NET contains various frameworks that help developers for creating rich web applications. Some of the frameworks include [30]:

- ASP.NET web forms
- ASP.NET MVC
- BFC
- MonoRail
- Vici MVC

However, we will not discuss all of these frameworks but we will focus only in two of them in which “find local food” project will be implemented.

4.8.1 ASP.NET web forms

ASP.NET webforms framework is the next generation of Microsoft that help developers for creating rich, flexible and scalable web applications. Web forms contain numerous server controls that are intended the easiest way to create the User interface (UI) by just dragging and dropping the elements in the workspace. These forms are similar to Windows forms except they contain properties, methods and events for response to the controls elements. The code structure in web forms is organized in aspx file pages which contain pure HTML and code behind classes [33].

Web forms and ASP.NET in comparison with classic ASP brings more facilities for developers such as: the code as HTML is separated from application logic that is in contrast with classic ASP where the code is mixed up in one page, by using web forms there is less code to write because of the data binding that they own, compiled code can be performed in multiple languages, PostBack mechanism helps to maintain POST data for every control and save the properties of every control in the View State.[33].

However, even that web forms provide a lot of flexibility for creation of web applications many criticisms comes from different developers regarding to them. This is because many developers encountered some limitations and problems in their developing pages. According to these developers the problems of web forms comes in PostBacks, View States, separation of concerns, lack support of testing and even driven approaches, problem of managing the code that resides behind the scene and which contains inline and mixed code together with a huge amount of ViewState etc. The use of ViewStates into pages can cause the pages to blow and increase in size, affects in performance issues for every back and forth request etc [32].

Another issue is with PostBacks mechanism which is used to maintain the state of server controls. So whenever a server control is added to the page the output will be rendered as HTML form tags together with the action attribute that tells the page where the form control is used. This can cause an issue to URL rewriting for instance whenever a request is made from the browser for a specific page then when a PostBack is occur the browser will show the rewritten URL the same as in action attribute [34].

Another concern when working with web forms is the lack of separation of concerns. By using web forms it is easy to mix up your User interface code with the logic code because of facilities of server controls that you just drag and drop into the workspace. This form of coding makes hard to maintain and unit test the applications. While there are a few drawback of using ASP.NET web forms, you can still develop scalable, stable and rich web applications. However, by seeing the above mentioned downsides for web forms framework Microsoft brought up new release framework known as ASP.NET
MVC which is not intended to replace web forms but is consider as an alternate framework for developing web applications [32, 34, 35].

4.8.2 ASP.NET MVC
Like web forms ASP.NET MVC is a framework that is build at the top of ASP.NET. ASP.NET MVC framework helps developers to improve some problems that arise with web forms framework and develop stable, testable and maintainable web applications. It is based on the model view control (MVC) design pattern and provides a clean separation of concerns that helps to work independently in every component and apply unit testing. This framework works quite differently and omits some of the parts that web forms use [35, 36, 37].

In contrast to web forms, ASP.NET MVC eliminates ViewState, PostBack mechanism or event driven approaches which means that there is no properties or methods to be set, there is no heavily ViewState code in the pages, events are controlled and accessed through the action methods that are part of controller etc. The User interface (UI) through ASP.NET MVC is design and work directly with HTML, Javascript and Css. In contrast to web forms when the request is made through URL, that request access some resources that are inside of the model, the model then response to the incoming request with a view as output. Another point to note here is that when request is made the input access to the action methods that are part of controllers which is in contrast to web forms where the request goes direct to the pages that are saved on the disk. Pages in ASP.NET MVC are HTML tags do not contain any code behind as long as the code logic stays in the model. This helps for better unit test of the application and clean separation of concern. Summing up, now it is up to the developers which framework to choose for the next project by considering pros and cons of both frameworks [35, 36, 37].

4.8.3 Making the choice ASP.NET web forms or ASP.NET MVC
Choosing between ASP.NET web forms and ASP.NET MVC framework for your project is a personal choice and all depends on the project needs. With use of ASP.NET MVC we want not to say that ASP.NET web forms are going away. ASP.NET web forms are still a good pattern that you can create rich web applications. When Microsoft releases ASP.NET MVC they did not mean it as a replacement framework. Instead, it simple brought up a new alternative framework for creation of web applications. As a web developer it is really important to know the differences of these frameworks before starting creating an application even that they are kind of similar.

The figure 4.10 depicts some factors that are needed to take into consideration before making the choice for your project if the development platform is going to be ASP.NET web forms or ASP.NET MVC.
<table>
<thead>
<tr>
<th>ASP.NET WEB FORMS</th>
<th>ASP.NET MVC</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Based on”Page controller” pattern.</td>
<td>• Based on”Model View controller” pattern.</td>
</tr>
<tr>
<td>• Rapid development of your project.</td>
<td>• Takes more time to write the code but worth of it later in your project.</td>
</tr>
<tr>
<td>• Tons of control that makes it easy by just dragging and dropping facility.</td>
<td>• Does not provide the facility of dragging and dropping.</td>
</tr>
<tr>
<td>• Possibility of adding third party controls</td>
<td>• Does not contain the box controls</td>
</tr>
<tr>
<td>• No control over your HTML markup.</td>
<td>• Full control over your HTML markup.</td>
</tr>
<tr>
<td>• No clean separation of concern for your project.</td>
<td>• Clean separation of concern.</td>
</tr>
<tr>
<td>• Contains the ViewState mechanism that makes pages to blow in their contents.</td>
<td>• No ViewState mechanism</td>
</tr>
<tr>
<td>• Contains the PostBack mechanism that makes harder for search engines to rank pages.</td>
<td>• No Postback mechanism</td>
</tr>
<tr>
<td>• Testability for the project almost impossible.</td>
<td>• Testability everywhere based on Test driven development (TDD).</td>
</tr>
<tr>
<td>• Based on the page lifecycle during runtime.</td>
<td>• Simple runtime stack without page lifecycle.</td>
</tr>
<tr>
<td>• No flexible extensibility over your project</td>
<td>• Responsibility for every components</td>
</tr>
<tr>
<td>• No friendly Users/SEO over your URL</td>
<td>• Replacement of any components without affects changes on other components.</td>
</tr>
<tr>
<td></td>
<td>• Extensibility everywhere.</td>
</tr>
<tr>
<td></td>
<td>• Friendly Users/SEO and clean URL</td>
</tr>
<tr>
<td></td>
<td>• Better design of your code.</td>
</tr>
<tr>
<td></td>
<td>• Loose coupling between components.</td>
</tr>
</tbody>
</table>

Figure 4.10 Factors before making the choice [45].
5 IMPLEMENTATION OF FIND LOCAL FOOD

The thesis project Find Local Food is based on some ideas from other find local food sites [42,43] and it is developed in ASP.NET language within two forms of frameworks ASP.NET web forms and ASP.NET MVC. The layout of the system find local food is based on simply and good design through Cascading Styling Sheet (CSS) and Flashes. The system contains the navigation links such as: home, find local farms, market place, seasonal news, newsletter, about us, contacts us, which the following subsection is going to navigate through all those navigation links by describing their purposes in the project.

5.1 Home page
This page will contain updated daily information for different local producers. The home page of project is depicted in appendix page [Figure 1].

5.2 Find local farms page
This page contains the Google map integrated in the system that makes easy for users to search for different local producers that are located in the Ljungby city. The Google map contains some markers that points to the local producer’s location and a pop up window by showing the service and profile for each individual producers. Users can easily interact with the Google map by seeing exact location of the local producers. When the user clicks in the marker the pop up window will be appear that shows some information such as postal code, name of producer and a picture of a product service for a particular producer. Moreover, the system contains a search engine where users are able to search for local producers according to postal code or name of the city [Figures 2, 3, 4 5, 6].

5.3 Market place page
This page contains some categories of products (i.e. bakery, fish, meat, drinks etc.) and list producers according to specific category. When the user will click for a specific category (i.e. bakery, fish) the system will be redirected to another page that shows all the producers that are part of the particular category. Moreover, users are able also to see the profile for any particular producer for more information [Figures 7, 8, 9].

5.3.1 Database
We saw that when the users click in the specific category the system list the local producers for that particular category. How this is possible is that the system contains the relational database tables between Category-Producer and ProducersContact-Producer as one-to-many relationships [Figure 10].

5.3.2 Administration page
The system contains the administration page that only logged in users can access to the database and manipulate with it (i.e. create, edit or delete items in the database). After the administrator of the page is successfully logged on the system then the system will direct to the admin page where the administrator can manipulate with items in database such as create, edit or delete options. The administrator first choose the add categories link then he will be directed to categories tables. After the administrator has given the data for creating new item, the data will be saved in the database and then it will be shown in the market places page [Figures 11, 12, 13, 14, 15, 16].
5.4 Seasonal news page
This page is going to contain different news about local producers and their products according to the season’s events.

5.5 News letter page
This page contains possibility of subscribing the newsletter. Users can easily subscribe for news letter by just providing the email address and receive monthly news letter from local producers. This facility of the system will help producers to keep updated their customers with different news about their products. After the user has provided the email address, the email is going to be registered in the database [Figures 17, 18].

5.6 About us page
This page is going to contain some information and purposes of the system.

5.7 Contact us page
This page will contain the possibility that users can submit any questions by either sending email or by sending a message direct. When the users choose a form for sending a message from the page, the data are saved in database [Figures 19, 20, 21, 22].

5.8 Analysis of the ASP.NET frameworks in project
There are differences when programming in ASP.NET web forms and ASP.NET MVC. With use of web forms there are various controls that you can drag and drop into workspace for performing any task. For instance, in the project find local food I used detailsView control for rendering all categories products from database.

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="150px" AutoGenerateRows="False"
DataSourceID="SqlDataSource1" <Fields>
<asp:TemplateField HeaderText="SortExpression="CategoryName" ItemTemplate>
<asp:Label CssClass="table" ID="Label2" runat="server"
Text="Categories products"></asp:Label>
<img src="Images/Bakery1.png" class="RightSideImages" width="60" height="60" alt="" /> <br/>
<a class="three" href="Bakery.aspx">Bakery</a><br />
<img src="Images/meat1.png" class="RightSideImages" width="60" height="60" alt="" /> <br/>
<a class="three" href="ProducersMeat.aspx">Meat</a><br />
<img src="Images/Vegetables.png" class="RightSideImages" width="60" height="60" alt="" /> <br/>
<a class="three" href="Drinks.aspx">Drinks</a><br />
</ItemTemplate>
</asp:TemplateField>
</Fields>
</asp:DetailsView>

Figure 5.23 Example of DetailsView code

In contrast, ASP.NET MVC use the model which contains the code logic in a class and then it is the controller’s responsibility to choose the appropriate view for rendering the data from model.
Another difference of ASP.NET web forms and ASP.NET MVC is in URL routing. In web forms, when a request is made from the users that request is accessed to the files as aspx pages that are saved on the disk.
Whereas, in ASP.NET MVC when the request is made from the users as HTTP, routing mechanism forward that request to the controller which contains action method classes. In contrast to web forms that contains aspx pages or any other string query in URL, MVC provides clean URL.

Moreover, ASP.NET web forms use PostBacks when any button control is clicked. I will take an example when the button for newsletter subscription is clicked.
So when the user gives the input value in text box and clicks the button for subscription, the method is post while the action is aspx page that contains code behind to response to the event. Web forms use the mechanism of ViewStates to save the properties for every control.
In contrast, ASP.NET MVC implementation does not use controls, it uses HTML helpers methods. It also does not use code behind pages, postbacks and viewstates. Instead when the request is made it goes through the controller’s action methods which then contact the model to get the data result and return it back as html code to the appropriate view.

```csharp
using (MyDataBaseMailingMessageEntities3 dabl = new MyDataBaseMailingMessageEntities3())
{
    subscribe.SentDateTime = System.DateTime.Now;
    subscribe.IPAddress = Request.UserHostAddress.ToString();
    dabl.AddToNewsLetter(subscribe);
}
```

Figure 5.30 HTML code and code logic for newsletter box
The following screenshot shows MVC source code in a page without ViewStates string code as hidden field.

Figure 5.31 Source view page, ASP.NET MVC
6 CONCLUSION AND FUTURE WORK

This chapter close this thesis report by describing the goals that we have achieved so far and what can be extended in the project as a future work.

6.1 Conclusion

We saw in chapter 2 how architecture and design are important to understand for large and complex systems. Then we implemented the same project “find local food” in two frameworks that ASP.NET comprises which were the goals to achieve. The purpose of project is developed with the aim of showing location of different local producers and their product services.

During implementation of this project in both frameworks of ASP I realized that ASP.NET MVC brings more benefits and options to choose compared to web forms even that it requires more code to write. With use of ASP.NET MVC in my project I had a clean separation of concern which applied in an easy way testing of every components, clean URL, no code behind pages as long as the logic code is reside in the model component etc. On the other hand, the development project through web forms framework was more easy and simple as long as I used different controls by just dragging and dropping into workspace and write code behind classes to response to events for these controls. Maintainability of the code was hard in this framework since any changes made in code affected the overall application not to work. Moreover, testability for the code was impossible. Finally, I conclude that is a personal choice which framework to used when implementing the applications. However, if you want to have control over your project, easy maintain of the code, clean URL route, then ASP.NET MVC as a framework best fits for your project.

6.2 Future work

As a future work, there is possibility for extension development of find local food project. The same project can be developed through ASP.NET MVC but working on mobile devices.
7 REFERENCES


[7] Kai Qian, Xiang Fu, Lixin Tao, Chong-Wei Xu, Jorze Herrera “Software Architecture and design Illuminated”


[9] Selena Sol, The thick of things,


[12] Grady Booch, The architecture of web applications,


[17] Site References

[18] Choosing a web application development language

[19] eGrove System, “How to choose a programming language for web application”,
http://www.idealmarketers.com/?articleid=1067886, 29-Mar-10


[22] Mathew MacDonald and Mario Szpuszta, “Pro ASP.NET 3.5 in C# 2008”


http://www.exforsys.com/tutorials/application-development/what-is-n-tier.html


[28] White paper, Using Model View Controller pattern in web interactions,
http://www.uidesign.net/Articles/Papers/UsingMVCPatterninWebInter.html, Thursday,June 01-2000.

[29] Tony Martson, The Principles of MVC design Pattern,
http://www.tonymarston.net/php-mysql/model-view-controller.htm, May 2, 2004


[37] Jeffrey Palermo Ben Scherman Jimmy Bogard,” ASP.NET MVC in action”.


[42] http://www.bigbarn.co.uk/


APPENDIX A

A.1 Home page

Figure 5.1 Home page

A.2 Find local farms

Figure 5.2 Google map
Figure 5.3 Manipulations with Google map

Figure 5.4 Markers in Google map and pop up window
Figure 5.5 Google map search through city

Figure 5.6 Google map search through postal code
A.3 Market place page

Figure 5.7 Market place page

Figure 5.8 List of producers
A.4 Database & Administration page

Figure 5.9 Producer information

Figure 5.10 Relational database
Figure 5.13 Add Category option

Figure 5.14 Create Category
Figure 5.15 Added category in database

Figure 5.16 Listed category in market place page
A.5 Newsletter page

Figure 5.17 Newsletter subscription

Figure 5.18 Database table for newsletter subscription
A.6 Contact us page

Figure 5.19 Contact us form

Figure 5.20 Sample contact us form
**Figure 5.21** Sample message sent

**Figure 5.22** Database table for contact us message