National College of Ireland

PGDip in Cloud Computing

2011/2012

James Peyton - 10206515

Emily Maycock - 11207864

Brian Raymond - 10206248

Stephen O’Rourke - 11107065

Hotel Directory

Project in

Enterprise Frameworks



**Table of Contents**

[Executive Summary](#_Toc311720608) 2

[1. Introduction](#_Toc311720608) 3

[2.](#_Toc311720611) 6

[3.](#_Toc311720608) 13

[4. Project Plan](#_Toc311720608)

[5.](#_Toc311720608)

6.

[7.](#_Toc311720608)

[8.](#_Toc311720608)

[9.](#_Toc311720608) 3

[10.](#_Toc311720611) 6

[11.](#_Toc311720608) 13

[12.](#_Toc311720608)

[13.](#_Toc311720608)

14.

[15.](#_Toc311720608)

[16. Appendix A Team Project Management](#_Toc311720608)

**1. Introduction**

**blah blah**

**2. Background research and investigations**

**3. Requirements analysis**

**4. Project Plan**

When the project was initially started each team member took it upon themselves to research the .Net framework and to gain an understanding of working with C# programming. The plan in place was for each member to have an attempt at all parts in the creation of the web application. All team members at some stage in the development of the application had problems using GitHub and had to continuously “clone” their folders on their local machines. Emily Maycock had continuous problems with GitHub which severely hampered her attempts to commit files.

Team members Emily and James looked at creating a database whilst team members Brian and Stephen set about creating the models and eventually the parser.

A breakdown of all the work completed to date is contained in the Appendix A.

**5. Software development methodology employed**

**Visual Studio**

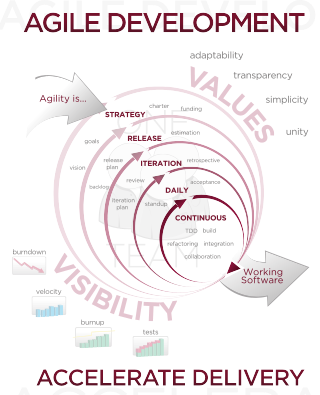
The application was built in the Integrated Development Environment (IDE) Microsoft Visual Studio 2010. This environment provided all of the necessary tools for constructing each aspect of the application. It also automatically generates tests which ensure that each component functions correctly before it is fully implemented.

Microsoft’s object-oriented language, C#, was used to code the application.

ASP.NET

**Agile Software Development**

Although not developing a full enterprise-level application, the team-based approach to the project meant that some of the aspects of agile software development were used.



**Fig 5.1 Agile Development Diagram**

Some of the hallmarks of the agile approach include adaptive planning and evolutionary development and delivery. It also promotes rapid and flexible response to change. Throughout the development of the ‘Hotel Directory’, team members were responsible for different aspects of the system and flexibility was crucial as circumstances changed and goals were re-evaluated.

As of 2001, the ‘Agile Manifesto’ has been used as a set of guidelines for employing this approach to software development. Although this was not followed strictly during the development of the Hotel Directory, many of the principles were followed. For example:

* Individuals and Interactions – Interactions between team members and often programming in pairs to achieve goals or overcome obstacles
* Working software – A working version of the application was up and running quickly
* Responding to change – If one approach was not working, focus was switched to another method or idea. The application was scrapped and restarted at one point. This was necessary and allowed the project to progress.
* Simplicity – It was decided early on that application should cover the requirements of the project, but not become overly complicated.
* Face-to-face conversation – The most substantial progress was achieved when 2 or more team members collaborated.

**6. Use cases/Wireframes**

**7. Architecture/Design approach**

Models (Class Models / Data Models etc.)

**8. Implementation of particular OOP constructs**

Object oriented programming (OOP) is a programming paradigm using “objects” or data structures which consist of data fields and methods used together with their interactions to design applications. Unified Modelling language (UML) is used today as the standard for specifying, visualising, constructing and documenting the layout of software systems. A construct helps defines the relationship between UML models and accompanying implementation code in C# and this would include classes, interfaces, inheritance and associations. In OOP, classes are the actual containers for all of our coding.

**Classes**

In the Hotel Directory application a number of model classes are created which form the basis of our code first entity framework application. Each class created has attributes and behaviours and the behaviours are the actions that the class performs. In OOP, classes are the actual containers for all of our coding and these classes are held in another container called a package. Packages allow this application to logically group together to form the make-up of our Hotel Directory application.

**Interfaces**

An interface is sometimes referred to as a contract between two entities and using interfaces we can invoke functions from different classes through the same interface reference.

**Inheritance**

Is the process describing the hierarchical relationship between classes and in (UML) inheritance is noted by a solid line with a closed arrow that points to a superclass. The relationship is called a generalisation. In the application the Data Access Layer or (DAL) layer inherits from the ingestion folder containing the CSV file.

**NOT FINISHED YET, DONT EVEN KNOW IF THIS IS RIGHT!!!!!!**

**9. Design Patterns and Architectural Patterns**

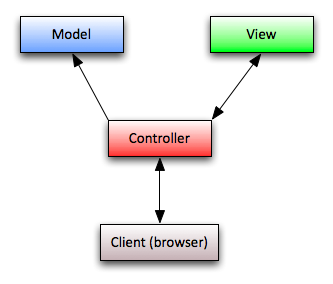
Domain Model?

**Design pattern**

- Strategy pattern/adapter design?

**Architectural patterns**

The principal architectural pattern employed with this application is MVC, which incorporates the Model, View and Controller. This allows for code reusability and ‘separation of concerns’.



**Fig 9.1. MVC structure**

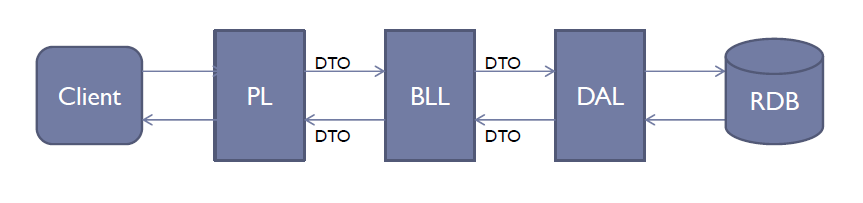
There have been a few revisions of this architecture since its inception. The latest, MVC 4, is used here. One of the core components introduced with MVC 4 is the ASP.NET Web API, which is well-suited to building REST-ful services. Features of the web API include:

* **Full support for routes:**Support for the full set of route capabilities of ASP.NET Routing.
* **Content negotiation:**Automatic support for XML, JSON, and Form URL-encoded formats.
* **Model binding and validation:**Model binders allow for simple data extraction from the different parts of an HTTP request. These parts are conerted into .NET objects which can be used by the Web API actions.
* **Code-based configuration:**ASP.NET Web API configuration is achieved using code only. Other related setup files are created automatically
* **Improved support for Inversion of Control (IoC) containers:**Support provided for IoC containers.
* **Scaffolding:**Use the **Add Controller** dialog to quickly scaffold a web API controller. This is based on an ‘Entity Framework’ based model type.

**5 layers implemented**

The ‘Hotel Directory’ web application has been built upon the 5 layers of a layered architecture system, as follows:

* User Interface/Client – HTML page containing data generated by the Hotel Directory
* Presentation Layer -
* Business Logic Layer -
* Data Access Layer – Persists hotel information from csv sources into the database, and retrieves data back for eventual presentation to the user (after filtering through higher layers).
* Database – SQL server DB which retains hotel information.



**Fig 9.2. Five layers**

**How cross-cutting concerns have been handled**

**10. Security of the application**

**11. Configuration of the application**

**12. Scalability of the application**

**13. Testing Approach**

Unit testing

Functional testing (errors encountered, etc)

**References**

ASP.NET MVC 4 : Official Microsoft Site. 2012. ASP.NET MVC 4 : Official Microsoft Site. [ONLINE] Available at: <http://www.asp.net/whitepapers/mvc4-release-notes>. [Accessed 24 July 2012].

Agile software development - Wikipedia, the free encyclopedia. 2012. Agile software development - Wikipedia, the free encyclopedia. [ONLINE] Available at: <http://en.wikipedia.org/wiki/Agile_software_development>. [Accessed 24 July 2012].

Object Oriented Programming. 2012. Object Oriented Programming. [ONLINE] Available at: http://tonyfear.netau.net/index.php?option=com\_content&view=article&id=124:object-oriented-programming&catid=3:c-category&Itemid=5. [Accessed 25 July 2012].

**16. Appendix A - Team Project Management**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Team Jebs | | | |  |
| To-do List | | | |  |
|  |  |  |  |  |
| **Task No.** | **Task** | **By** | **% Complete** | **Date** |
| **A** | **Documentation** |  |  |  |
| 1 | Proposal |  |  |  |
| 2 | Design Use cases | EM | 50% |  |
| 3 | Final Project Report | All | 5% |  |
| 4 | Update main document with use cases | EM | 0% |  |
| 5 | Use Cases created | EM | 100% | 11/7/2012 |
| 6 | Git problems resolved update | JP | 100% | 11/7/2012 |
| 7 | Use Cases | EM | 100% | 13/7/2012 |
| 8 | MVC3 Design Pattern | JP | 100% | 17/7/2012 |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **B** | **General** |  |  |  |
| 1 | Install MVC 4 | All | 100% | 10/06/2012 |
| 2 | Create MVC4 Project and upload to Git | EM | 100% | 14/06/2012 |
| 3 | Data Sets Added | BR | 75% | 19/6/2012 |
| 4 | Data Sets Added | BR | 85% | 4/7/2012 |
| 5 | New project created | SOR | 100% | 7/7/2012 |
| 6 | Database Created | BR | 75% | 10/7/2012 |
| 7 | Home Page update | SOR | 100% | 15/7/2012 |
| 8 | Data Set Added | JP | 100% | 16/7/2012 |
| 9 | Database tables updated | JP | 100% | 16/7/2012 |
| 10 | References Added | JP | 100% | 21/7/2012 |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| 15 |  |  |  |  |
| **C** | **Data Modelling/Parser** |  |  |  |
| 1 | Model Attempt | BR | 100% | 17/6/2012 |
| 2 | Model Attempt | SOR | 100% | 24/6/2012 |
| 3 | IData Parser | SOR | 100% | 3/7/2012 |
| 4 | Parser London | SOR | 100% | 3/7/2012 |
| 5 | Parser New York | BR | 100% | 3/7/2012 |
| 6 | Model DB Attempt | SOR | 100% | 7/7/2012 |
| 7 | Parser Las Vegas | JP | 100% | 16/7/2012 |
| 8 | Parser Load | BR | 85% | 17/7/2012 |
| 9 | CSV Parser | JP | 100% | 21/7/2012 |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| 15 |  |  |  |  |
| **D** | **DAL/Business Logic** |  |  |  |
| 1 | DAL Layer created | SOR | 20% | 10/7/2012 |
| 2 | DAL Layer updated | EM | 100% | 23/7/2012 |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **E** | **MVC** |  |  |  |
| 1 | Controllers & Views created | BR | 100% | 17/7/2012 |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **E** | **Test Driven Development** |  |  |  |
| 1 | Test class London Parser | SOR | 100% | 10/7/2012 |
| 2 | New Test Methods | SOR | 100% | 17/7/2012 |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |