Universitá degli studi di Modena e Reggio Emilia

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea in Informatica

SVILUPPO DEL SITO ABILITAZIONI USL IN ASP.NET TRAMITE VISUAL STUDIO .NET

Relatore:
Prof. Riccardo Martoglia

Tesi di Laurea di:
Andrea Baraldi

Anno Accademico 2010 / 2011

Indice

Introduzione

L'obiettivo di questa tesi é l'analisi dello sviluppo del sito Abilitazioni USL. Il sito nasce per essere utilizzato in un ambiente come un'azienda USL per gestire le attivitá effettuate dal personale. Il sito conterrá tutte le anagrafiche dei dipendenti presenti all'interno dell'azienda con i relativi dati e ruoli. All'interno dei dipendenti vengono nominati alcuni responsabili che avranno la possbilitá di accettare o rifiutare le attivitá inserite per ogni utente.
Il sito permette di dividere gli utenti presenti in gruppi in relazione ai ruoli che ricoprono. La funzionalitá principale riguarda la possibilitá di inserire un'attivitá per un determinato utente, specificandone tutte le caratteristiche, che verrá tenuta in sospeso fino a quando un utente responsabile non ne deciderá la convalida o il rifiuto.

Figura 1: Schermata Iniziale del sito Abilitazioni USL

Il sito offrirá la possibilitá di analizzare in dettaglio le attivitá presenti effettuando diversi tipi di ricerca. Infatti sará possibile ricercare le attivitá inserite riferite ad un determinato utente o un determinato gruppo.
Di seguito analizziamo in maniera molto sintetica la struttura del sito nelle sue diverse sezioni. Il sito Abilitazioni USL si divide principalmente in 5 parti:

- **Anagrafiche** Permette di visualizzare i dati anagrafici degli utenti presenti con la possibilitá di visualizzare gli applicativi che hanno associati
- **Gestione gruppi** Visualizza i gruppi presenti nel sito, permettendo la visualizzazione degli applicativi e degli utenti associati a quel gruppo.
- **Riferimenti per Gruppo** Dopo aver effettuato il login a un determinato gruppo nella sezione Gestione Gruppi permette la visualizzazione e la modifica delle attivitá per quel determinato gruppo.
- **Elenco Responsabili** Visualizza la lista dei responsabili.
- **Applicativi** Visualizza la lista degli applicativi, permettendo la visualizzazione delle anagrafiche associate.
- **Operazioni** Visualizza la lista delle operazioni che devono essere confermate o respinte.

La schermata iniziale del sito é visibile in figura 1.

Per lo sviluppo del sito Abilitazione USL si é scelto il linguaggio ASP.NET in quanto permette la visualizzazione da parte di tutti gli utenti utilizzando solamente un browser web.
La prima parte della tesi ha come obiettivo quello di analizzare i vantaggi nell'utilizzo del linguaggio ASP.NET. Successivamente verrá analizzata in maniera teorica la struttura del database che conterrá i dati inseriti.
Nella seconda parte della tesi verranno analizzate in dettaglio le operazioni che potrá compiere l'utente all'interno del sito definendo in maniera ancora astratta le operazioni che dovranno essere effettuate. Successivamente verrá analizzata con maggiore dettaglio la struttura di ogni tabella specificando il tipo i dato che verrá inserito in ogni campo.
Nell'ultima parte della tesi verranno analizzate le principali funzioni che strut-

turano il sito e offrono all'utente la possibilitá di effettuare tutte le operazioni analizzate in dettaglio nelle parti precedenti.

Parte I

Il caso di studio

In questa sezione di tesi verranno analizzati gli aspetti piú teorici del progetto sviluppato. Nel primo capitolo verrá analizzato in dettaglio il linguaggio di programmazione utilizzato, evidenziandone gli aspetti positivi e negativi. Nello sviluppo del sito Abilitazioni USL é stato scelto come linguaggio di programmazione Visual Basic .NET e ASP.NET.
Nel secondo capitolo verrá analizzata la struttura del database utilizzato nello sviluppo del sito. Si studierá in dettaglio la struttura del modello ER e le tabelle che saranno utilizzate per implementare lo schema introdotto.

Capitolo 1

Analisi di ASP.NET

Le vecchie applicazioni web basate sul server venivano Sviluppate secondo linguaggi di scripting che utilizzavano un sistema di tag convenzionali. La maggior parte di questi sistemi di sviluppo per applicazioni web si limitava a far eseguire applicazioni o lanciare componenti sul server. Non prevedevano una moderna piattaforma integrata per facilitare lo sviluppo di questo tipo di applicazioni.
Quindi le applicazioni sviluppate prima dell'introduzione di ASP.NET venivano divise in due categorie:

- Gli script venivano interpretati solo a lato server
- Piccole e separate applicazioni erano eseguite a seguito di chiamate del server.

Le classiche pagine sviluppate in ASP (Active Server Pages, la versione di ASP precedente ad ASP.NET) ricadevano nella prima categoria. Lo sviluppatore aveva il compito di scrivere script che contenevano codice integrato. Il file script veniva esaminato da un altro componente che provvedeva a generare usuale codice HTML durante l'esecuzione dello script. Lo svantaggio nell'uso di questo tipo di linguaggio risiede nel fatto che l'esecuzione di un'applicazione script richiede piú tempo rispetto all'esecuzione di un'applicazione compilata. Inoltre l'uso di applicazioni script rende difficoltoso il controllo della sicurezza delle informazioni e non rende evidenti i problemi di un'inefficiente gestione.
Il secondo tipo di approccio, sviluppabile tramite Perl con CGI (Common Gateway Inteface), evidenzia un'altra serie di problematiche. In questi ambienti di

sviluppo il serve lancia applicazioni separate per rispondere alle richieste del client. Queste vengono eseguite e viene generato dinamicamente del codice HTML, rispedito al client. L'esecuzione di questo tipo di applicazioni é molto piú veloce rispetto alle applicazioni script, peró richiede un notevole uso di memoria. Un altro svantaggio é che il server é cosrtetto a creare una nuova istanza dell'applicazione per ogni client che ne fa richiesta. Questo modello rende questo tipo di applicazioni inutilizzabile per sistemi con un alto numero di connessioni simultanee, inoltre sono particolarmente complesse da sviluppare e non sono integrabili con altri componenti.
ASP.NET introduce un modello di sviluppo completamente nuovo, essendo profondamente integrato con il framework sottostante. Infatti ASP.NET non é un'estensione del .NET Framework, ma é una porzione di esso gestito dal .NET runtime. In pratica ASP.NET unisce il concetto di svilupppo per applicazioni a quello di sviluppo web, estendendo gli stumenti e le tecnologie precedentemente possedute dai sistemi di sviluppo di applicazioni desktop.

1.1 Problemi con ASP classico

Il linguaggio ASP classico é comunque un ottimo strumento per lo sviluppo di applicazioni web usando le tecnologie Microsoft. Inoltre, come tutti i modelli si sviluppo, ASP permette di gestire in maniera ottimale alcuni aspetti, mentre presenta comunque alcune problematiche, risolte con l'introduzione di ASP.NET.

1.1.1 Lunghezza del codice

Una classica pagina in ASP contiene solitamente molto codice server-side intervallato da codice HTML. Consideriamo il seguente esempio, che genera una lista in HTML del risultato di una query effettuato su un database.

```
<%
Set dbConn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")
dbConn.Open "PROVIDER=SQLOLEDB;DATA SOURCE=(local);
DATABASE=Pubs;User=sa;Password=sa"
```

```
%>
<select name="cboAuthors">
<%
rs.Open "SELECT * FROM Authors", dbConn, 3, 3
Do While Not rs.EOF
%>
<option value="<%=rs("au_id")%>"><%=rs("au_lname") & ", " &
rs("au_fname")%></option>
<%
rs.MoveNext
Loop
%>
</select>
```

Questo semplice funzione necessita di 16 righe di codice per essere effettuata. Questo modo di scrivere script diminuisce la performance di un'applicazione poiché mescola HTML e script. Infatti quando la pagina viene processata dall'estensione ASP ISAPI (Internet Server Application Programmin Interface) che é in esecuzione sul web server, il motore che processa il codice script viene attivato e disattivato svariate volte durante questa singola richiesta. Questo causa un aumento del tempo necessario a processare l'intera pagina e spedirla al client. Inoltre pagine scritte secondo questo metodo rischiano di diventare enormemente lunghe, rendendo difficile la lettura e il debugging. Indipendentemente dal tipo di approccio scelto nella scrittura del codice, ASP tende a diventare enorme e di difficile interpretazione.
In ASP.NET questi problemi non esistono, in quanto le pagine vengono scritte secondo il modello di programmazione ad oggetti. Le pagine web vengono sviluppate con lo stesso approccio utilizzato per le applicazioni desktop. Questo significa che non 1'e piú necessario combinare codice HTML con codice ASP.

1.1.2 Linguaggi di Script

Al momento della sua creazione ASP sembrava la soluzione perfetta per i programmatori di applicazioni desktop che volevano muoversi nel mondo del Web,

poiché permetteva agli sviluppatori di utillizare un linguaggio familiare come VBScript su una piattaforma server. Questa familiaritá con VBScript peró introdusse alcuni problemi di performance e non solo.
Ogni oggetto usato nel classico ASP veniva creato come un tipo di dato variant che é debolmente caratterizzato. Questo richiedeva un notevole impiego di memoria, influenzando notevolmente la velocit1'a di esecuzione. Inoltre il compilatore e gli strumenti di sviluppo non potevano identificarli nel momento della scrittura del codice. Questo rendeva impossibile creare un integrato IDE (Integrated Development Environment) simile a quello presente in Visual Basic e Visual C++, che permettesse un debug ottimale e mettesse in evidenza eventuali errori. Quindi senza strumenti di debugging i programmatori si trovarono in difficoltá a risolvere i problemi nei loro script.
ASP.NET risolve anche questi problemi, poiché le pagine ASP.NET vengono eseguite all'interno di un CLR (Common Language Runtime), quindi possono essere sviluppate in qualsiasi linguaggio possegga un compilatore compatibile con CLR. Questo rende possibile utilizzare alcuni linguaggi di programmazione ad oggetti come Visual Basic e Visual C#.
É inoltre importante sottolineare che le pagine ASP.NET non vengono interpretate, ma vengono compilate in assemblies (termite utilizzato in .NET per indicare unitá di codice compilato). Questo é uno dei piú significativi cambiamenti introdotti con il modello di sviluppo web ASP.NET 2.0 introdotto da Microsoft.

1.2 Punti di forza di ASP.NET

Andremo ora ad analizzare in dettaglio i punti punti che rendono ASP.NET un linguaggio innvovativo, profondamente differente dagli altri tipi di linguaggio finora utilizzati nello sviluppo di applicazioni web.

1.2.1 ASP.NET é integrato nel .NET Framework

Il .NET Framework é diviso un'accurata collezione di parti funzionali, con un numero totale di circa 7000 tipi (il termine .NET per indicare classi, strutture, intefacce e altri nuclei di programmazione). Ognuno di queste migliaia di tipi

di dato nel .NET Framwork é raggruppato in un contenitore logico e gerarchico chiamato namespace. Differenti namespace hanno diverse caratteristiche. L'insieme dei namespace offerti dal .NET Framework offrono funzionalitá per qualsiasi tipo di sviluppo. Questo grande strumento che li raggruppa é denominato class library.
É interessante notare che il modo in cui vengono usate le classi del .NET Framework per lo sviluppo di applicazioni ASP.NET é lo stesso per qualsiasi altro tipo di linguaggio presente nel .NET Framework.

1.2.2 ASP.NET é Compilato, non Interpretato

Uno degli aspetti negativi di ASP classico risiede nel fatto che gli script in tutte le pagine ASP vengono interpretati, causando una perdita di performance. Questo significa che ogni volta che viene richiesta una pagina al server, quest'ultimo deve interpretare il codice ASP linea per linea per tradurlo in linguaggio macchina di basso livello. É noto che il processo di interpretazione del codice é lento.
Le pagine ASP.NET, al contrario di ASP, vengono compilate (infatti non é possibile eseguire un programma sviluppato in VB.NET o C# senza averlo prima compilato).
La compilazione delle pagine sviluppate in ASP.NET avviene in due fasi. Nella prima fase il codice in C# o VB.NET viene compilato in un linguaggio intermedio, chiamato Microsoft Intermediate Lamguage (MSIL). L'introduzione di questo linguaggio intermedio é il ragione fonadmentale per cui il .NET puó essere considerato come linguaggio interdipendente. Essenzialmente tutti i linguaggi .NET vengono compilati nello stesso codice MSIL. La prima fase di compilazione puó avvenire automaticamente quando la pagina viene richiesta per la prima volta, oppure puó essere eseguita in precedenza effettuando una precompilazione. Il file MSIL compilato viene generato in assembly.
La seconda fase di compilazione avviene prima che la pagina vada in esecuzione. In questa fase il codice MSIL viene compilato in linguaggio macchian di basso livello. Questa fase viene denominata compilazione just-in-time (JIT) e avviene nello stesso modo per tutte le applicazioni .NET. Uno schema riassuntivo della compilazione nelle applicazioni .NET é visibile nella figura 1.2.2.

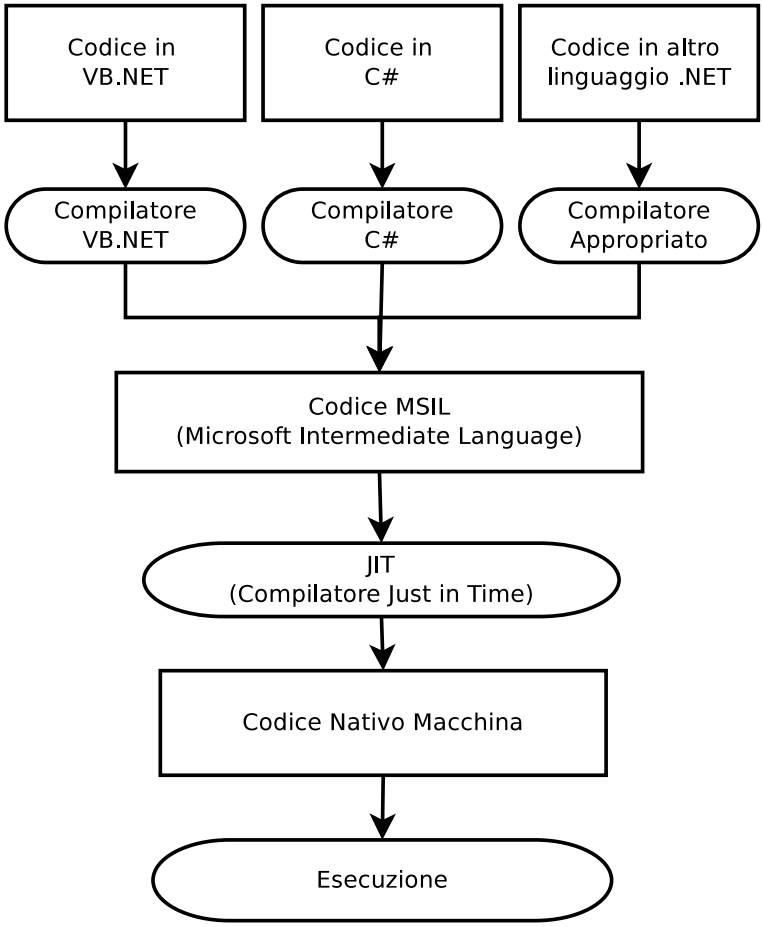

Figura 1.1: Schema di compilazione delle applicazioni .NET

La compilazione .NET é suddivisa in due fasi proprio per offrire al programmatore una buona convenienza in temrini di tempo e un'ottima portabilitá. Prima che un compilatore possa tradurre il codice in linguaggio macchina, é necessario conoscere il tipo di sistema operativo e la piattaforma hardware sulla quale l'applicazione verrá eseguita (ad esempio Windows a 32 o 64 bit). La compilazione a due fasi permette di creare del codice assembly compilato con il codice .NET che potrá essere distribuito su piú di una piattaforma.

Dopo una prima analisi puó sembrare che la compilazione JIT non sia cosí performante in quanto deve avvenire ogni volta che un utente richiede una pagina web del sito. Fortunatamente le applicazioni ASP.NET non richiedono la compilazione del codice ogni volta che la pagina viene rihiesta. Infatti il codice MSIL viene generato una volta sola, e ricreate solo se il codice della pagina viene mod-

ificato. Allo stesso modo il codice tradotto il linguaggio macchina viene salvato in una cartella di sistema.

1.2.3 ASP.NET é Multilinguaggio

La scelta di un particolare linguaggio di sviluppo in ASP.NET non é di particolare importanza per il risultato finale poiché i diversi tipi di linguaggio vengono compilatinello stesso tipi di file MSIL. MSIL é un passaggio fondamentale per tutte le applicazioni gestite (per applicazioni gestite si intende qualsiasi applicazioni scritta per .NET che viene eseguita all'interno dell'ambiente gestito del CLR). Sotto questo punto di vista MSIL é il linguaggio di .NET ed é l'unico linguaggio riconosciuto da CLR.
Per capire meglio la struttura di MSIL consideriamo il seguente esempio, scritto in VB.NET:

```
Namespace HelloWorld
 Public Class TestClass
  Private Shared Sub Main(Args() As String)
   Console.WriteLine("Hello World")
  End Sub
 End Class
End Namespace
```

Questo codice mostra una funzione basilare disponibile in .NET scrivendo un semplice messaggio nella finestra console. Diamo ora un'occhiata al codice MSIL generato dalla prima compilazione:

```
.method public static void Main() cil managed
{
 .entrypoint
 .custom instance void [mscorlib]System.STAThreadAttribute::.ctor()
 = ( 01 00 00 00 )
 // Code size 14 (0xe)
 .maxstack 8
 IL_0000: nop
 IL_0001: ldstr "Hello World"
```

```
 IL_0006: call void [mscorlib]System.Console::WriteLine(string)
 IL_000b: nop
 IL_000c: nop
 IL_000d: ret
} // end of method TestClass::Main
```

É abbastanza facile analissare il codice MSIL per qualsiasi applicazione compilata .NET. Basta eseguire il Disassembler, che viene installato con Visual Studio e .NET SDK (Software Development Kit), posizionato in una directory del tipo C: Program Files Microsoft Visual Studio SDK v2.0 Bin con il nome ldasm.exe. Una volta eseguito il programma é sufficiente aprire qualsiasi DLL o EXE creato con .NET.
Con un pó di pazienza é possibile decostruire il codice IL in maniera abbastanza semplice, apprendendone la sua struttura. La semplicitá con cui é possibile decostruire il codice MSIL introduce dei problemi nel controllo della privacy e sulla sicurezza del codice.
Per uno sviluppatore ASP.NET questo problema diventa in realtá di scarso interesse poiché tutti i file sono memorizzati sul server. Infatti il client non riceve mai il codice compilato MSIL, ma riceve solo il risultato dell'esecuzione dell'applicazione.
Questa vulnerabilitá del codice MSIL é possible risolverla utilizzando un offuscatore che riscrive il codice rendendolo di piú difficile comprensione (ad esempio rinominando le variabili utilizzando nomi privi di senso). Visual Studio include al suo interno un offuscatore, denominato Dotfuscator.

1.2.4 ASP.NET é eseguito all'interno del Common Language Runtime

Forse l'aspetto piú importante di ASP.NET da ricordare é che viene eseguito all'interno del CLR. L'intero .NET Framework, cioé tutti i namespace, applicazioni e classi, si riferiscono ad un unico codie gestito. Tralasciando un'analisi dettagliata del CLR, elenchiamo solo gli aspetti che portano maggiore beneficio:

- *Gestion automatica della memoria e collezioni garbage*

  Ogni volta che l'applicazione crea un'istanza di una classe, il CLR allo-

ca spazio nella memoria heap per quell'oggetto. Inoltre non é necessario pulire questa memoria manualmente. Quando il riferimento a un oggetto non é piu' visibile (o l'applicazione termina), l'oggetto diventa disponibile per la collezione garbage. Il collettore garbage viene eseguito periodicamente all'interno di CLR, liberando la memoria inutilizzata dagli oggetti che sono diventati inaccessibili.

- *Type Safety*

  Quando viene compilata un'applicazione, .NET aggiunge informazioni al codice assemply riguardanti le classi disponibili, i componenti, i loro data type e altro ancora. Il risultato é che il codice compilato é completamente autosufficiente. ALtri sviluppatori possono utilizzarlo senza la necessitá di richiamare altri tipi di file di supporto, e il compilatore puó é in grado di verificare se ogni chiamata é valida al momento dell'esecuzione. Questa funzione di sicurezza permette di eliminare errori di basso livello come il buffer overflow che compare spesso nellos viluppo di applicazioni in C++.

- *Metadata estendibili*

  Le informazioni riguardanti le classi e i componenti é solo un tipo di metadata che .NET salva nel codice assembly compilato. Il Metadata descrive il codice e consente di fornire informazioni aggiuntive riguardanti il runtime e altri servizi. Ad esempio il metadata puó indicare aun debugger su come tracciare il codice, o puó indicare a Visual Studio come visualizzare un controllo personalizzato nella fase di design. É inoltre possibile specificare nel metadata quali servizi di runtime abilitare (come metodi web o servizi COM+).

- *Gestione strutturata degli errori*

  La scrittura di codice in Visual Basic o VBScript presentava alcune limitazioni riguardanti le limitate risorse di questi linguaggi per la gestione degli errori. Con una gestione strutturata degli errori é possibile organizzare il codice logicamente e in maniera concisa permettendo una gestione degli errori. É possibile creare blocchi separati per differenziare i diversi tipi di errore. É inoltre possibile nidificare gestori di eccezioni a diversi livelli di profonditá.

- *Multithreading*

  Il CLR mette a disposizione una serie di thread che possono utilizzare diverse classi. Ad esempio é possibile chiamare metodi per leggere file o comunicare con i servizi web in modo asincrono senza la necessitá di creare nuovi thread in modo espclicito.

1.2.5 ASP.NET é orientato agli oggetti

ASP classico fornisce un modello di programmazione ad oggetti relativamente leggero, anche se puó essere esteso tramite pesanti oggetti COM. Mette a disposizione solo un piccolo insieme di oggetti, che sono in realá un sottile strato di dettagli di HTTP e HTML.
Al contrario ASP.NET é completamente orientato agli oggetti. Non solo codice ha pieno accesso a tutti gli oggetti del .NET Framework, ma é inoltre possibile sfruttare tutte le convenzioni di un ambiente OOP (Object-Oriented Programming), come l'incapsulamento e l'ereditarietá. Ad esempio é possibile creare delle classi riutilizzabili, standardizzare codice con le interfacce e utilizzare tutte le funzionalitá in un componente compilato distribuibile.
Uno dei maggiori esempi di orientamento agli oggetti visibile in ASP.NET si trova nei controlli server-based. I controlli server based sono l'esempio di incapsulazione. Gli sviluppatori possono modificare i controlli server tramite il codice personalizzando l'aspetto, fornire dati per la visualizzazione e introdurre una reazione agli eventi. Il codice HTML é nascosto dietro al codice che lo sviluppatore scrive. Per evitare la scrittura di codice HTMl da parte del programmatore cono presenti degli oggetti che provvedono a tradurre la pagina in HTML, una volta che questa é stata renderizzata. In questo modo ASP.NET offre una serie di controlli che permettono di astrarre il basso livello di programmazione HTML e HTTP.
Di seguito é possibile analizzare il codice ASP.NET per generare una standar text box in HTML:

```
<input type="text" id="myText" runat="server" />
```

Con l'aggiunta dell'attributo *runat=server*, questo oggetto statico HTML diventa un controllo completamente funzionale server-side manipolabile tramite il

codice. É possibile lavorare con eventi server-side che vengono generati, modificare attributi, e associarlo ad un'origine di dati.
Ad esempio é possibile impostare il testo della text box quando la pagina viene caricata per la prima volta tramite il seguente codice:

```
Private Sub Page_Load(ByVal sender As Object,
 ByVal e As EventArgs)
Handles Me.Load
 myText.Value = "Hello World!"
End Sub
```

Tecnicamente questo codice imposta la proprietá Value di un oggetto HtmlInputText. Il risultato finale é che la stringa di testo appare nella text box nella pagina HTML dopo che questa é stata renderizzata e inviata al client.

1.2.6 ASP.NET é Multidevice e Multibrowser

Una delle maggiori difficoltá che si presentano ad uno sviluppatore web é affrontare la vasta gamma di browser web differenti. Le diverse marche di browser differiscono per il loro supporto HTML. Lo sviluppatore Web deve decidere il linguaggio che userá per mostrare le informazioni nel browser, come HTML 3.2, HTML 4.0, o qualcosa di completamente diverso come XHTML 1.0 o anche WML (Wireless Markup Language) per dispositivi mobili. Questo problema, incentivato dalle compagnie di browser, ha afflitto gli sviluppatori sin da quando il World Wide Web Consortium propose la prima versione HTML. La situazione diventa ancora piú difficoltosa se si decide di utilizzare un'estesione client-side HTML come JavaScript per creare una pagina web un po' piú dinamica.
ASP.NET riesce a risolvere questo problema in una maniera intelligente. Anche se é possibile recuperare dal client informazioni sul browser in uso e le sue capacitá in una pagina ASP.NET, ASP.NET incentiva gli sviluppppatori ad ignorare queste condiderazioni utilizzando un approccio nuovo. Infatti ASP.NET rende disponibile una ricca suite di controlli server web che adattando automaticamente il codice HTML in relazione alle capacitá del browser usato dal client. Un esempio é dato dai controlli di convalida, che usano JavaScript e DHTML (Dynamic HTML), che migliorano il comporamento di questi controlli

se il client lo supporta. Questo permette di mostrare in maniera dimanica messaggi di errore senza che l'utente sia costretto a inviare la pagina al server web per effettuare controlli. Queste carattiristiche sono opzionali, ma permettono di gestire in maniera ottimale qualsiasi tipo di browser, senza escludere nessun client nella visualizzazione.

1.2.7 ASP.NET é facile da distribuire e configurare

Uno dei piú grandi rompicapi che si trova e risolvere uno sviluppatore web giunge al moemnto della distribuzione di un'applicazione completa su un server. Non solo é necessario trasferire i file delle pagine web, ma devono essere traferiti i database e i componenti procedendo ad un'inevitabile registrazione di componenti. Questo comporta la riconfigurazione di tutte le impostazioni di configurazione.

ASP.NET semplifica notevolmente questi passaggi. Infatti in ogni installazione del .NET Framework sono disponibili le stesse classi core. Questo rende la distribuzione di applicazioni ASP.NET molto semplice in quanto non é necessario registrare nuovamente tutti i componenti. Talvolta é sufficiente copiare tutti i file in una directory vurtuale su un server (utilizzando un programma FTP e XXOPY da riga di comando). Se la macchina host ha installato il .NET Framework non sono richiesti tempo di registrazioni lunghi.

Distribuire i componenti dell'applicazione é altrettanto facile. É sufficiente copiare i gruppi di componenti quando si distribuisce l'applicazione web. Poiché tutte le informazioni sui componenti sono memorizzate direttamente nel metadata del file assembly, non é necessario lanciare un programma di registrazione o modificare il registro di Windows. Basta posizionare questi componenti nella corretta directory (la sottodirectory bin della directori dell'applicazione web), e il motore ASP.NET li rileverá automaticamente rendendo disponibile il codice per le pagine web.

La configurazione con ASP classico é un altro momento in cui si presentano notevoli difficoltá nella distribuzione di un'applicazione. In particolare é di particolare difficoltá trasferire informazioni di protezione, come gli account utente e i privilegi utente.

Anche in questo caso ASP.NET rende questa fase particolarmente semplice e

rapida, riducendo al minimo la dipendenza dalle impostazioni di IIS (Internet Information Services). La maggior parte delle impostazioni di ASP.NET vengono memorizzate in un apposito file web.config, collocato nella stessa directory delle pagine web. Questo file contiene un insieme di impostazione dell'applicazione organizzate in modo gerarchico memorizzato in un formato di facile lettura come XML che é possibile modificare anche con un semplice editor di testo. Quando viene modificata un'impostazione dell'applicazione, ASP.NET segnala questo cambiamento e riavvia l'applicazione in un nuovo dominio (mantenendo il dominio dell'applicazione precedente in vita abbastanza a lungo in modo da terminare l'elaborazione di tutte le eventuali richieste in sospeso). Il file web.config non é mai bloccato, permettendo la modifica delle impostazioni in qualsiasi momento.

1.3 Connessione ad un DB con ASP.NET

Utilizzando la classe Connection é possibile effettuare la connessione a un DBMS. Poiché il database utilizzato nle nostro progetto é stato costruito con Microsoft Access si utilizzerá il provider di dati OledbCommand. Il provider di dati .NET Framework di OLE DB descrive un insieme di classi utilizzate per accedere ad un origine OLE DB nello spazio gestito. DI seguito vengono riportate le classi principali che serviranno allo sviluppo del nostro progetto:

- *OleDbCommand* Rappresenta un'istruzione SQL da eseguire in relazione ad una origine di dati.
- *OleDbConnection* Un oggetto OleDbConnection rappresenta una connessione univoca a un'origine dati. Nel caso di un sistema di database client/server, é equivalente a una connessione di rete al server. A seconda delle funzionalitá supportate dal provider OLE DB nativo, é possibile che alcuni metodi o proprietá di un oggetto OleDbConnection non siano disponibili.
- *OleDbDataAdapter* La classe OleDbDataAdapter funge da ponte tra una classe DataSet e un'origine dati per il recupero e il salvataggio dei dati. La classe OleDbDataAdapter fornisce tale ponte tramite il metodo Fill

per caricare i dati dall'origine dati nella classe DataSet e tramite il metodo Update per inviare di nuovo all'origine dati le modifiche apportate in DataSet. Quando tramite a classe OleDbDataAdapter viene riempita una classe DataSet, vengono create le tabelle e le colonne necessarie per i dati restituiti, se non giá esistenti.

Capitolo 2

Il sito Abilitazioni USL

Il sito Abilitazioni USL é stato strutturato in diverse sezioni principali. Le sezioni presenti sono:

- Anagrafiche
- Gestione Gruppi
- Riferimento per Gruppi
- Elenco Responsabili
- Applicativi
- Operazioni

Ogni sezione effettua operazioni sulle stesse tabelle presenti nel databse, utilizzando peró diversi criteri di ricerca.
La sezione anagrafiche permette di visualizzare la lista degli utenti presenti e le attivitá associate ad ogni singolo utente.
La sezione gestione gruppi permette la visualizzazione dei gruppi inseriti, consentendo la visualizzazione delle anagrafiche iscritte ad un determinato gruppo.
La sezione riferimenti per gruppi permette la visualizzazione delle attivitá degli utenti registrati ad un determinato gruppo.
La sezione Elenco Reponsabili invece visualizza la lista degli utenti che possiedono piú diritti nella gestione del sito.
La sezione Applicativi visualizza la lista degli applicativi presenti.

L'ultima sezione, quella Operazioni, permette agli utenti Responsabili di approvare o rifiutare le attivitá inserite per ogni utente.
Attraverso le funzioni introdotte nel sito é possibile interrogare il database in maniera differente in relazione alla sezione scelta. Inoltre le sezioni Anagrafiche e Gestione Gruppi permettono di inserire nuove attivitá per ogni utente.

2.1 Analisi dei dati

Il sito Abilitazioni USL si appoggia ad un database (GEABDATI.mdb) per gestire tutte le attivitá inserite per gli utenti. In questa base di dati sono presenti 6 entitá: EAANDIPE, ACANAGR, ACGRUP, ACAPPLICAT, TABENTITA e TABMOD.
Lo schema ER é stato creato rispoducendo la struttura del database esistente, evidenziando le relazioni tra le diverse entitá.
Ad ogni utente é possibile associare piú attivita. Ogni attivitá viene associata ad un singolo valore delle entitá ACAPPLICAT, TABENTITA e TABMOD, dove sono specificate i dettagli dell'attiviá che si é voluti inserire.
Per quanto riguarda l'inserimento dei dati nel database, questi erano giá presenti in fase di analisi.

2.2 Schema ER

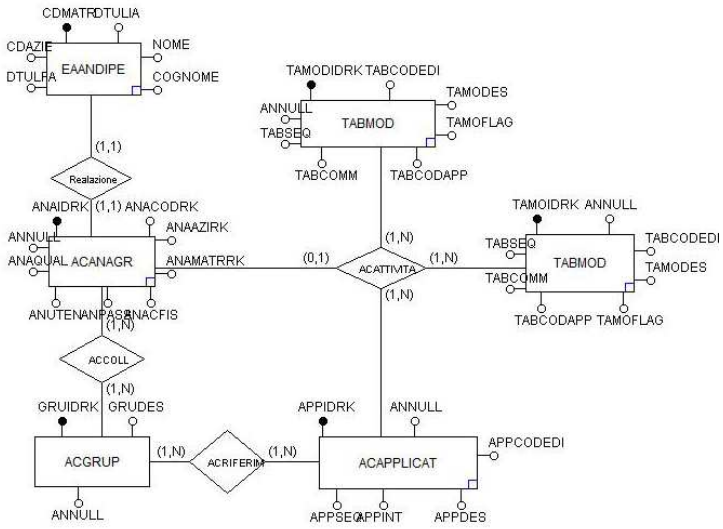

Figura 2.1: Schema ER

2.3 Traduzione dello schema ER in progetto logico

In questa sezione verrá analizzata passo passo la costruzione dello schema logico effettuando un'analisi dettagliata dello schema ER precedente.
Per quanto riguarda le entitá, per ognuna di esse viene creata una tabella che possa contenere i dati. La tabella EAANDIPE conterrá il nome, il cognome e la matricola di ogni utente presente nel sito.

EAANDIPE(CDAZIE, CDMATR, COGNOME, NOME, DTULIA, DTULFA)

La tabella ACANAGR contiene informazioni relative alla qualifica, al nome utente e la password scelti dall'utente.

ACANAGR(ANAIDRK, ANNULL, ANACODRK, ANAAZIRK, ANAMATRRK, ANACFIS, ANPASS, ANUTEN, ANAQUAL)

La tabella ACGRUP contiene la lista dei gruppi che sono stati creati.

ACGRUP (GRUIDRK, ANNULL, GRUDES)

La tabella ACAPPLICAT contiene invece la lista degli applicativi che é possibile inserire.

ACAPPLICAT(APPIDRK, ANNULL, APPCODEDI, APPDES, APPINT, APPSEQ, APPCOMM)

Nella tabelle TABMOD vengono salvate le informazioni relative alle modalitá disponibili.

TABMOD(TAMODIDRK, ANNULL, TABCODEEDI, TAMODES, TAMOFLAG, TABCODAPP, TABCOMM, TABSEQ)

La tabella TABENTITA contiene le informazioni relative alle entitá che é possibile scegliere per ogni aattivitá inserita.

TABENTITA (TBEIDRK, ANNULL, TBETBDRK, TBEDES, TBESEQ, TBECOD, TBEOPE, TBECOMM)

Andiamo ora ad analizzare la relazione presente tra le entité EAANDIPE e ACANAGR. Poiché la relazione presente tra le due entitá é di tipo (1,1) é stato aggiunto il campo ANMATRRK alla tabella ACANAGR collegato al campo CDMATR della tabella EAANDIPE.
La relazione ACCOLL mette in collegamento le tabelle ACANAGR e ACGRUP, specificando gli utenti iscritti ad un gruppo. Per questa relazione é stata creata un'apposita tabella poiché presenta una cardinalitá (1,n). Nella tabella ACCOLL vengono inoltre memorizzate le informazione relative alla data di inizio e di fine della permanenza dell'utente nel gruppo e la data in cui é stato effettuato l'inserimento.

ACCOLL(COLIDRK, ANNULL, COLGRURK, COLANARK, COLDINI, COLDFIN, COLTIPO, COLVARRK, COLRIFRK, COLUTINS, COLDINS, COLOINS, COLOPE, COLUTOPE, COLDOPE, COLOOPE)

AK: COLGRURK, COLANARK

FK: COLGRURK References ACGRUP

FK: COLANARRK References ACANAGR

La relazione ACRIFERIM collega le tabelle ACGRUP e ACAPPLICAT. Tramite la relazione ACRIFERIM vengono specificate gli applicativi associati ad ogni singolo gruppo. Anche in questo caso, poiché la relazione presenta una cardinalitá (1,n), é stata creata una tabella per memorizzare le informazioni relative alla relazione. Anche in questa tabella vengono memorizzate informazioni relative alla data di inizio e di fine dell'aplicativo associato a quel gruppo e la data in cui si é effettuata l'operazione.

ACRIFERIM(RIFIDRK, ANNULL, RIFGRURK, RIFAPPRK, RIFDINI, RIFDFIN, RIFTIPO, RIFUTINS, RIFDINS, RIFOINS, RIFOPE, RIFUTOPE, RIFDOPE, RIFOOPE, RIFVARRK)
AK: RIFGRURK, RIFAPPRK
FK: RIFGRURK References ACGRUP
FK: RIFAPPRK References ACAPPLICAT

La relazione ACATTIVITA mette in relazione le tabelle ACAPPLICAT, TABENTITA, TABMOD e ACANAGR. Poiché questa relazione presenta una cardinalitá (1,n) é stata creata una tabella apposita per contenere tutte le informazioni. Anche in questo caso vengono salvate informazioni aggiuntive riguardanti l'inizio e la fine dell'attivitá e la data in cui é avvenuto l'inserimento.

ACATTIVITA(ATTIDRK, ANNULL, ATANAPRK, ATTRIFRK, ATTAPPRK, ATTTBMRK, ATTVARRK, ATTCODATT, ATTRESPRK, ATTDATINI, ATTDATFIN, ATTDIFF, ATTUTINS, ATTDINS, ATTOINS, ATTOPE, ATTUTOPE, ATTDOPE, ATTOOPE, ATTCOLPRK, ATTMODRK)
AK: ATANAPRK, ATTAPPRK, ATTTBMRK, ATTCODTATT
FK: ATANAPRK References ACANAGR
FK: ATTAPPRK References ACAPPLICAT
FK: ATTTBMRK References TABMOD
FK: ATTCODTATT References TABENTITA

2.4 Relazione tra tabelle

In figura 2.2 é possibile vedere graficamente le tabelle con i rispettivi campi. In particolare sono state evidenziate le relazioni che sussistono tra di loro.

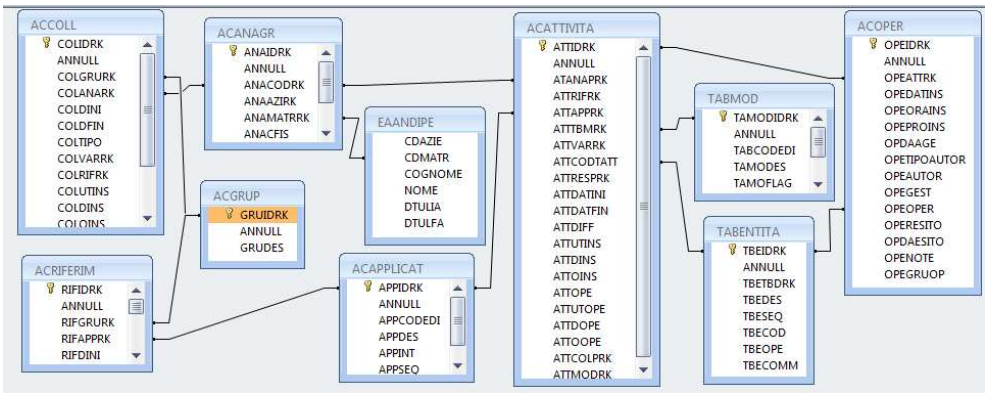

Figura 2.2: Relazione tra tabelle

Parte II

Area Web per il sito Abilitazioni USL

In questa seconda parte della tesi verrá analizzata piú in dettaglio la struttura del sito.
Inizialmente saranno specificate i requisiti funzionali che dovrá possedere il sito Abilitazioni USL. Per ogni requisito funzionale verranno specificate ulteriori proprietá che dovranno essere possedute da ogni sezione presente.
Successivamente verranno analizzati gli Activity Diagram, cioé le operazioni che dová compiere l'utente per ottenere una funzionaliá. In ultima analisi verrá analizzata in dettaglio la struttura di ogni singola tabella utilizzata per la memorizzazione dei dati.

Capitolo 3

Progetto

In questo capitolo verranno analizzate le principali funzionalitá del sito. Verranno introdotti i requisiti funzionali che dovrá posedere il sito Abilitazioni USL e i casi d'uso. Successivamente verranno introdotti gli Activity Diagram dove saranno specificate le operazioni che dovrá compiere l'utente persfruttare pienamente una funzionalitá presente.

3.1 Requisiti Funzionali

In questa sezione verranno illustrati i principali requisiti che sono stati successivamente realizzati durante la fase di programmazione.

3.1.1 Inserimento Applicativo

Introduzione: é possibile inserire un determinato applicativo associato ad un determinato utente. Vengono inoltre specificate altre informazioni, quali il responsabile, le modalitá, il tipo di attivitá, la data di inizio e la data di fine.
Input: dati inseriti in un form con scelte predefinite.
Processing: i dati inseriti nel form verranno salvati nel database.

3.1.2 Inserimento Gruppo

Introduzione: fornisce la possibilitá di inserire un nuovo gruppo.
Input: dati inseriti in un form.

Processing: i dati inseriti vengono salvati nel database.

3.1.3 Approvazione di applicativi

Introduzione: in relazione ai permessi posseduti dall'utente visualizza la lista di applicazioni che puo' approvare o respingere.
Input: Pulsanti di conferma.
Processing: Vengono modificati i campi riferiti allo stato dell'operazione dell'attivitá selezionata.

3.1.4 Aggiunta responsabile

Introduzione: in relazione ai permessi posseduti dall'utente é possibile aggiungere un responsabile.
Input: Lista degli utenti presenti
Processing: variazione dei valori dei campi relativi ai permessi posseduti dall'utente selezionato.

3.2 Casi d'uso

Il responsabile che accede al sito dopo aver effettuato il login puó decidere che operazioni eseguire scegliendo una determinata sezione.

3.2.1 Caso d'uso: Anagrafiche

Precondizioni:

- Il responsabile possiede i permessi necessari alla visualizzazione degli utenti.

Sequenza eventi:

- Il responsabile accede alla sezione Anagrafiche.
- Clicca sul pulsante Applicativi relativo all'utente interessato
- In fondo alla lista degli applicativi di quell'utente clicca sul pulsante - Inserimento

- Completa il form selezionando i diversi campi dai menu.
- Clicca sul pulsante Inserimento

Postcondizioni:

- Al termine della procedura viene visualizzata la nuova lista degli applicativi per l'utente. É possibile uscire o inserire un'ulteriore applicativo.

3.2.2 Caso d'uso: Gestione gruppi

Precondizioni:

- Il responsabile possiede i diritti per visualizzare i gruppi

Sequenza eventi:

- Il responsabile accede alla sezione Gestione Gruppi.
- Clicca sul tasto Login relativo al gruppo interessato.
- Clicca sul tasto Modifica relativo all'applicativo da modificare.
- Compila i campi del Form.
- Clicca sul tasto Modifica
- Il sistema aggiorna i dati relativi all'applicativo editato.

Postcondizioni:

- Al termine della procedura é possibile modificare un'ulteriore applicativo oppure uscire dal gruppo e selezionarne uno differente.

3.2.3 Caso d'uso: Elenco Responsabili

Precondizioni:

- Il responsabile possiede i diritti inserire nuovi responsabili

Sequenza eventi:

- Il responsabile accede alla sezione Elenco Responsabili
- Clicca sul pulsante Aggiunta responsabile da anagrafiche

- Clicca sul tasto modifica relativo all'utente desiderato.
- Il sistema aggiunge l'utente selezionato alla lista dei responsabili, variandone le proprietá

Postcondizioni:

- Al termine della procedura é possibile nominare un ulteriore responsabile oppure uscire dalla sezione.

3.2.4 Caso d'uso: Applicativi

Precondizioni:

- Il responsabile possiede i diritti per la visualizzazione degli applicativi.

Sequenza eventi:

- Il responsabile accede alla sezione Applicativi
- Clicca sul pulsante Modifica
- Compila il form cambiando le informazioni presenti.
- Clicca sul pulsante Conferma modifiche.
- Il sistema aggiorna le informazioni relative all'applicativo modificato.

Postcondizioni:

- Al termine della procedura é possibile modificare un ulteriore applicativo o uscire dalla sezione.

3.2.5 Caso d'uso: Operazioni

Precondizioni:

- Il responsabile possiede i diritti per approvare o respingere le operazioni presenti

Sequenza eventi:

- Il responsabile accede alla sezione Operazioni

- Dalla lista degli applicativi in attesa di approvazione seleziona uno dei due pulsanti presenti, uno per approvare e uno per respingere.
- Il sistema aggiorna le informazioni relative all'applicativo in relazione alla scelta effettuata dal responsabile.

Postcondizioni:

- Al termine della procedura é possibile approvare o respingere un ulteriore applicativo o uscire dalla sezione.

3.3 Activity Diagram

3.3.1 Activity Diagram 1: Anagrafiche

Tramite la sezione anagrafiche il responsabile sará in grado di inserire nuovi applicativi in relazione ad un utente. Dopo aver selezionato l'utente interessato da una lista dovrá inserire le informazioni relative all'applicativo desiderato selezionando i campi nei menu presenti.

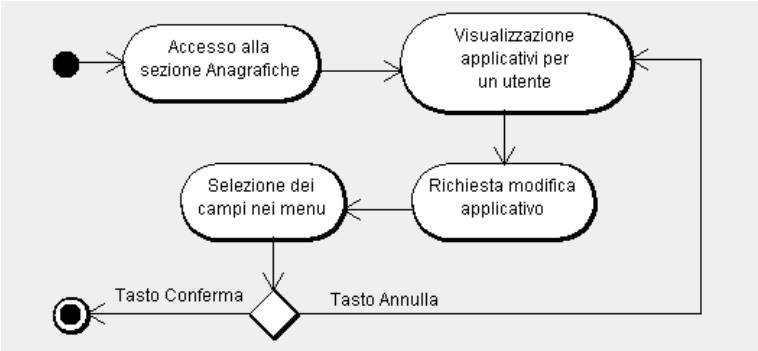

Figura 3.1: Activity Diagram 1

3.3.2 Activity Diagram 2: Gestione Gruppi

Tramite la sezione gestione gruppi il responsabile sará in grado di aggiungere un nuovo gruppo alla lista oppure modificare il nome di uno precedentemente esistente.
Il responsabile puó inoltre visualizzare la lista degli applicativi o degli utenti presenti in un determinato gruppo.
La funzione principale di questa sezione é quella di effettuare il login ad un determinato gruppo visualizzando tutti gli applicativi presenti. Una volta effettuato il login é in grado di modificare i dati relativi ad ogni applicativo presente o inserirne uno nuovo.

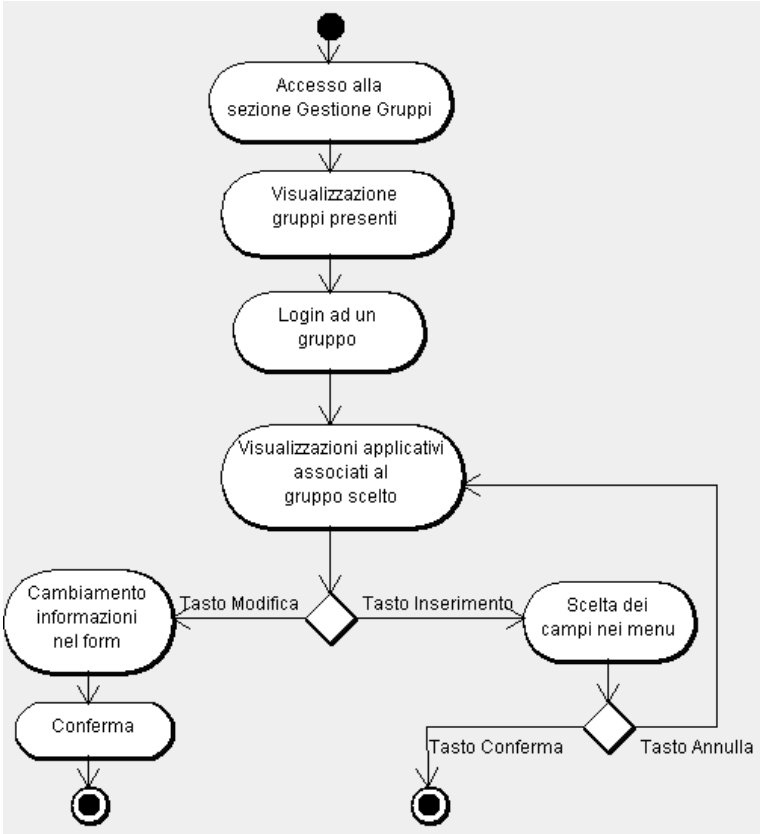

Figura 3.2: Activity Diagram 2

3.3.3 Activity Diagram 3: Elenco Responsabili

In questa sezione il responsabile visualizza la lista degli altri responsabili presenti e ha la facoltá di aggiungerne di nuovi selezionandoli dalla lista delle anagrafiche.

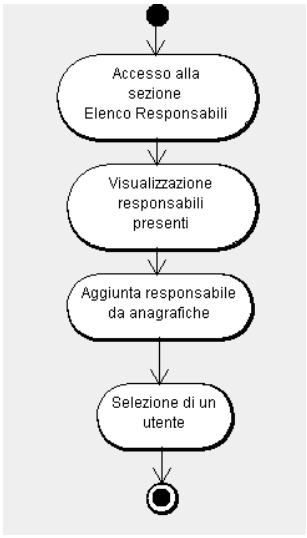

Figura 3.3: Activity Diagram 3

3.3.4 Activity Diagram 4: Applicativi

In questa sezione vengono visualizzati tutti gli applicativi presenti fornendo la possibilitá al responsabile di modificarli. É inoltre possibile visualizzare la lista degli utenti autorizzati e le modalitá.

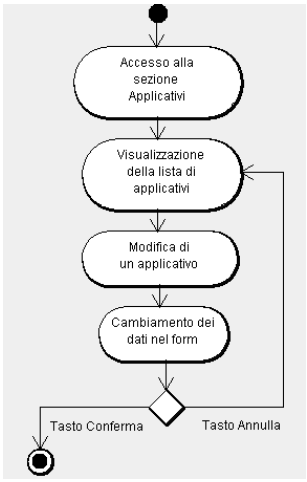

Figura 3.4: Activity Diagram 4

3.3.5 Activity Diagram 5: Operazioni

In questa sezione il responsabile é in grado di approvare o respingere tutte le attivitá in attesa di conferma.

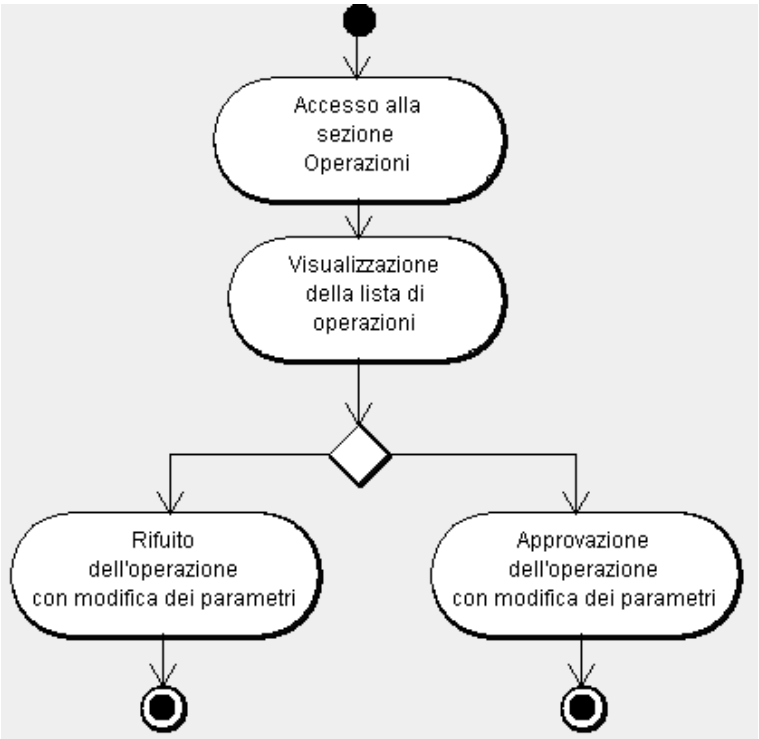

Figura 3.5: Activity Diagram 5

3.4 Struttura delle tabelle

I dati anagrafici degli utenti presenti vengono salvati in un'apposita tabella denominata EAAANDIPE. Ad ogni utente viene inoltre associato un codice matricola che lo identifica univocamente. La struttura della tabella é la seguente:

Tabella EAANDIPE

Nome Campo	**Tipo di dati**	**Descrizione**
CDAZIE	Numerico	Codice Azienda
CDMATR	Numerico	Codice Matricola
COGNOME	Testo	Cognome dell'utente
NOME	Testo	Nome dell'utente
DTULIA	Data/Ora	Data Inizio assunzione
DTULFA	Data/Ora	Data Fine assunzione

Ad ogni utente viene poi associato un nome utente e una password, che userá per poter eseguire le operazioni per cui é abilitato. Questa funzione non é stata completamente implementata, pero' sono state introdotte le funzioni che la predispongono. É inoltre presente il campo ANACODRK che é diverso per ogni utente e viene usato per collegare le diverse tabelle durante le ricerche. Questi dati vengono salvati nella tabella ACANAGR.

Tabella ACANAGR

Nome Campo	**Tipo di dati**	**Descrizione**
*ANAIDRK	Contatore	
ANNULL	Testo	= A Riga Annullata
ANACODRK	Numerico	Codice Effettivo
ANAAZIRK	Numerico	Codice azienda / ente
ANAMATRRK	Numerico	Matricola
ANACFIS	Testo	Codice Fiscale
ANPASS	Testo	Password Utente
ANUTEN	Testo	Utente
ANAQUAL	Testo	Descrizione qualifica

Le attivitá registrate per ogni utente vengono salvate nella tabella ACATTIVITA. In questa tabella l'utente a cui á associata l'attivit1'a viene identificato tramite il numero specificato nel campo ANACODRK della tabella ACANAGR.

Tabella ACATTIVITA

Nome Campo	**Tipo di dati**	**Descrizione**
*ATTIDRK	Contatore	
ANNULL	Testo	= A riga annullata
ATANAPRK	Numerico	ID della anagrafica iscritta
ATTRIFRK	Numerico	ID del record riferimento origine
ATTAPPRK	Numerico	ID del record applicativo
ATTTBMRK	Numerico	ID del record tabella modalita'
ATTVARRK	Numerico	ID del record variante
ATTCODTATT	Numerico	Attivitá
ATTRESPRK	Numerico	ID dell'anagrafica responsabile trattamento
ATTDATINI	Data/Ora	data inizio
ATTDATFIN	Data/Ora	data fine
ATTDIFF	Numerico	
ATTUTINS	Numerico	Utente inserimento
ATTDINS	Data/Ora	Data inserimento
ATTOINS	Data/Ora	ora inserimento
ATTOPE	Testo	tipo operazione (V=var,D=del)
ATTUTOPE	Numerico	Utente operazione
ATTDOPE	Data/Ora	Data operazione
ATTOOPE	Data/Ora	Ora operazione
ATTCOLPRK	Numerico	ID della riga collegato
ATTMODRK	Numerico	ID del record modalita

La tabella ACAPPLICAT contiene le descrizioni di ogni applicativo. Ogni campo é identificato univocamente da APPIDRK che viene utilizzato per collegare questa tabella a ACATTIVITA.

Tabella ACAPPLICAT

Nome Campo	Tipo di dati	Descrizione
*APPIDRK	Contatore	
ANNULL	Testo	= A riga annullata
APPCODEDI	Numerico	ID Edizione
APPDES	Testo	descrizione applicativo
APPINT	Testo	= I applicativo interno al prodotto
APPSEQ	Numerico	Sequenza nell'elenco
APPCOMM	Testo	Commento dell'applicativo

La tabella TABMOD contiene le descrizioni delle modalitá utilizzate nelle attivita. Ogni descrizione é identificata univocamente da TAMODIDRK che permette di collegarla alla tabella ACATTIVITA.

Tabella TABMOD

Nome Campo	Tipo di dati	Descrizione
*TAMODIDRK	Contatore	
ANNULL	Testo	= A riga annullata
TABCODEDI	Numerico	ID Edizione
TAMODES	Testo	Descrizione Modalitá
TAMOFLAG	Si/NO	Flag Scrittura Abilitata
TABCODAPP	Numerico	Codice Applicativo
TABCOMM	Testo	Commento Modalitá
TABSEQ	Numerico	Sequenza per applicativo

La tabella ACGRUP contiene la liste dei vari gruppi presenti nel sito Abilitazioni USL. Ad ogni gruppo viene associata la chiave univoca GRUIDRK che permette di identificarlo nelle altre tabelle.

Tabella ACGRUP

Nome Campo	Tipo di dati	Descrizione
*GRUIDRK	Contatore	ID del gruppo
ANNULL	Testo	= A riga annullata
GRUDES	Testo	Descrizione del gruppo

La tabella ACRIFERIM contiene i dati necessari a collegare i diversi gruppi ai loro applicativi. Viene utilizzato il campo RIFGRURK per collegarla alla tabella ACGRUP e il campi RIFAPPRK per il collegamento ad ACAPPLICAT.

Tabella ACRIFERIM

Nome Campo	Tipo di dati	Descrizione
*RIFIDRK	Contatore	
ANNULL	Testo	= A riga annullata
RIFGRURK	Numerico	Gruppo di abilitazioni
RIFAPPRK	Numerico	Puntatore all'id di ACAPPLICAT
RIFDINI	Data/Ora	Data inizio
RIFDFIN	Data/Ora	Data fine
RIFTIPO	Numerico	Gruppo o anagrafica
RIFUTINS	Testo	Utente inserimento
RIFDINS	Data/Ora	Data inserimento
RIFOINS	Data/Ora	Ora inserimento
RIFOPE	Testo	Tipo operazione
RIFUTOPE	Numerico	Utente operazione
RIFDOPE	Data/Ora	Data operazione
RIFOOPE	Data/Ora	Ora operazione

La tabella TABENTITA contiene i diversi valori delle entitá che possono essere scelte per ogni singola attivitá.

Tabella TABENTITA

Nome Campo	Tipo di dati	Descrizione
TBEIDRK	Contatore	
ANNULL	Testo	= A riga annullata
TBETBDRK	Numerico	id descrizione tabella
TBEDES	Testo	descrizione
TBESEQ	Numerico	Sequenza all'interno del gruppo tabelle
TBECOD	Testo	Codice esterno
TBEOPE	Testo	Codici operativi
TBECOMM	Testo	Commento/ note

La tabelle ACOPER contiene le informazioni relative all'approvazione o al rifuito di un'attivitá inserita.

Tabella ACOPER

Nome Campo	**Tipo di dati**	**Descrizione**
OPEIDRK	Contatore	
ANNULL	Testo	= A riga annullata
OPEATTRK	Numerico	ID del record Attivita'
OPEDATINS	Data/Ora	Data inserimento movimento
OPEORAINS	Data/Ora	Ora inserimento movimento
OPEPROINS	Numerico	Eventuale progressivo inserimento movimento
OPDAAGE	Data/Ora	Data massima
OPETIPOAUTOR	Numerico	Tipo autorizzante
OPEAUTOR	Numerico	Autorizzante
OPEGEST	Numerico	Operatore che effettua l'operazione
OPEOPER	Numerico	Operazione effettuata
OPERESITO	Numerico	Tabella ESITO
OPDAESITO	Numerico	Data variazione esito
OPENOTE	Testo	
OPEGRUOP	Numerico	Gruppo autorizzati

Nello sviluppo di questo progetto sono state introdotte numerose tabelle nel tentativo di ridurre la quantitá di dati memorizzati in ogni singola tabella. Infatti ad ogni elemento presente nelle entitá dello schema ER é associato un ID univoco che viene utilizzato nelle relazioni. Questo comporta un notevole risparmio di memoria di archiviazione in quanto nelle tabelle in cui vengono salvate le relazioni si ha solo un salvataggio di numeri. Con l'aumento dei dati presenti nel database questa si rivela una scelta efficace in quanto nelle tabelle non vengono ripetute stringhe ma solo numeri.

Capitolo 4

Implementazione

In questa ultima parte della tesi verrá preso in analisi il codice che compone il sito. In particolare verranno studiate con attenzione le due classi principali RComuni e RAbiliIO. Queste contengono le funzioni di maggiore importanza in quanto comunicano con il database per il salvataggio e la lettura di dati. In queste due classi vengono dichiarate anche le variabili globali che verranno utilizzate per il passaggio di informazioni da una pagina all'altra durante la navigazione.

4.1 Class RComuni

Per gestire le operazioni basilari del sito Abilitazioni USL é stata creata la classe RComuni. Questa classe contiene alcune variabili che permettono il salvataggio delle informazioni riguardanti le pagine visitate (Ptipop, Pchiamato, Pchiamante) e altre variabili che permettono la memorizzazione dei dati che verranno letti da database o sono in attesa di scrittura sullo stesso.
Di seguito analizzeremo in dettaglio le variabili principali della classe RComuni e le informazioni che vi sono memorizzate.
Nelle variabili

```
Public Ptipop as String
    Public Pchiamato As String
    Public Pchiamante As String
```

```
Public Pprecedente As String
```

vengono memorizzate le informazioni sulla pagina che é stata chiamata e da quale pagina é stata chiamata. Queste informazioni sono necessarie per l'esecuzione delle query in quanto vengono scelti criteri diversi di visualizzazione a seconda della sezioni in cui sono state invocate.
In questa serie di variabili

```
Public Yhaspulsa As String
Public Yhasvisi As String
Public Ypropuls As Integer
Public Yvispulsa As New Hashtable
```

vengono memorizzate le informazioni relative a quali pulsanti rendere visibili. Questi valori vengono cambiati effettuando una connessione al database leggendo le impostazioni sulla visibilitá dei pulsanti in relazione alla pagina su cui si é posizionati.

Vengono inoltre dichiarate due funzioni, Receivepar() e Trasmpar(), che permettono la lettura delle informazioni di sessione e la scrittura dei valori presenti nelle variabili di RComuni nelle informazioni di sessione.

```
Public Function Receivepar()
Ptipop = CType(Session("tipop"), String)
    If Ptipop <> "" Then
     Pchiamato = CType(Session("chiamato"), String)
        Pchiamante = CType(Session("chiamante"), String)
        Pcodn = CType(Session("codn"), Array)
        Ydativar = CType(Session("dativari"), Hashtable)
        Ymethod = CType(Session("metodi"), Hashtable)
        Ycompagina = CType(Session("compagina"), Hashtable)
        Ystkpagina = CType(Session("stkpagina"), Hashtable)
        Yvispulsa = CType(Session("pulsan"), Hashtable)
        Ycodici = CType(Session("codici"), Hashtable)
        Yfiltri = CType(Session("filtri"), Hashtable)
    Else
```

```
        Pchiamato = ""
        Pchiamante = ""
        Pulhash()
        Ptipop = "I"
    End If
End Function
```

La funzione Pulhas() provvede ad eliminare tutti i valori presenti nelle variabili Ydativar, Ymethod, Ycompagina, Ystkpagina, Ycodici e Yfiltri.

```
Public Function Trasmpar()
Session("tipop") = Ptipop
    Session("chiamato") = Pchiamato
    Session("chiamante") = Pchiamante
    Session("codn") = Pcodn
    Session("dativari") = Ydativar
    Session("metodi") = Ymethod
    Session("compagina") = Ycompagina
    Session("stkpagina") = Ystkpagina
    Session("filtri") = Yfiltri
    Session("codici") = Ycodici
    Session("pulsan") = Yvispulsa
End Function
```

La funzione Trasmpar() viene invocata ogni volta che viene lasciata una pagina per accedere ad una nuova.
É inoltre presente la funzione Impostafun() che ha una doppia funzionalitá connettendosi a a due tabelle differenti presenti nel database. La prima funzionalitá provvede alla dichiarazione delle variabili che verranno utilizzate nelle query a seconda della pagina che ha effettuato la richiesta. La seconda carica le informazioni sui pulsanti che dovranno essere visibili o meno.

```
Public Function Impostafun
(ByVal Ypchiamante As String, ByVal Ypchiamato As String)
'Dichiarazione delle variabili
```

```
ConnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
    Server.MapPath("App_Data/GEABDATI.mdb")
oConn = New System.Data.OleDb.OleDbConnection(ConnString)
oConn.Open()
sSQL = ""
sSQL = "SELECT pcm.* FROM  BAPCM00F pcm WHERE"
sSQL += " (pcm.CMANTE like '" + Ypchiamante + "')  "
sSQL += " And (pcm.CMATO like '" + Ypchiamato + "')   "
oCommand = New System.Data.OleDb.OleDbCommand(sSQL.Trim, oConn)
oAdapter = New System.Data.OleDb.OleDbDataAdapter(oCommand)
oDataSet = New System.Data.DataTable
oAdapter.Fill(oDataSet)
If oDataSet.Rows.Count > 0 Then
'Viene svuotato il contenuto della variabile Yfiltri e quelle associate
    'Successivamente vengono controllati
    'gli altri campi presenti nella tabella
    'e, se presenti, aggiunti i valori a Yfiltri.
End If
oConn.Close()

oConn1 = New System.Data.OleDb.OleDbConnection(ConnString)
oConn1.Open()
sSQL = ""
sSQL = "SELECT  * FROM BAELE00F  WHERE"
sSQL += " (BAELE00F.ELCMRK = " + Format$(Comcod, "00000000") + " ) "
oCommand1 = New System.Data.OleDb.OleDbCommand(sSQL.Trim, oConn1)
oAdapter1 = New System.Data.OleDb.OleDbDataAdapter(oCommand1)
oDataSet1 = New System.Data.DataTable
oAdapter1.Fill(oDataSet1)
II = 0
If oDataSet1.Rows.Count > 0 Then
For Each oRow1 In oDataSet1.Rows
    If oDataSet1.Rows(II).Item("ELCOMA").ToString <> "" Then
```

```
            Yhascom1 = oDataSet1.Rows(II).Item("ELCOMA").ToString
                Yhascom2 = oDataSet1.Rows(II).Item("ELPAGI").ToString
                Ycompagina.Add(Yhascom1, Yhascom2)
            End If
            If oDataSet1.Rows(II).Item("ELVISI").ToString <> "N" Then
                Yhaspulsa = oDataSet1.Rows(II).Item("ELELEM").ToString
                Ypropuls = Ypropuls + 1
                Yvispulsa.Add(Ypropuls, Yhaspulsa)
            End If
            II = II + 1
        Next
   Else
        Ex = 0
   End If
   oConn1.Close()
End Function
```

4.2 Class RAbilIO

Questa classe contiene tutte le variabili e le funzioni che permettono la gestioni delle informazioni presenti nel database. Sono presenti funzioni che effettuano la connessione al database, eseguono query e inseriscono dati all'interno delle tabelle. La connessione al database é permessa dalla funzione ConnectDB().

```
Public Function ConnectDB()
ConnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
    Server.MapPath("App_Data/GEABDATI.mdb")
    oConn = New System.Data.OleDb.OleDbConnection(ConnString)
    oConn.Open()
End Function
```

La funzione OperDB() esegue la stringa sql creata durante la chiamat di una funzione presente nella classe RAbilIO.

```
Public Function OperDB()
```

```
oCommand = New System.Data.OleDb.OleDbCommand(sSQL.Trim, oConn)
    oAdapter = New System.Data.OleDb.OleDbDataAdapter(oCommand)
    oDataSet = New System.Data.DataTable
    oDataSet.AcceptChanges()
    oAdapter.Fill(oDataSet)
End Function
```

Per ogni sezione é presente una funzione apposita che provvede all'inserimento, aggiornamento o lettura delle informazioni nel database. Analizzeremo le funzioni solo per una categoria in quanto per le altre le funzioni sono similari. Considereremo le funzioni che provvedono alla lettura delle attivitá presenti. La funzione che effettua la lettura dei valori nelle tabelle é GetAACATTIVITA().

```
Public Function GetACATTIVITA(ByVal Anagra As Integer, ByVal Appli As Integer,
    ByVal Modal As Integer, ByVal Funzio As Integer) As Integer
ConnectDB()
    sSQL = "SELECT * FROM ACATTIVITA WHERE ("
    sSQL += "    ATANAPRK = "
    sSQL += Anagra.ToString
    sSQL += "  and "
    sSQL += "    ATTAPPRK = "
    sSQL += Appli.ToString
    sSQL += "  and "
    sSQL += "    ATTTBMRK = "
    sSQL += Modal.ToString
    sSQL += "  and "
    sSQL += "    ATTCODTATT = "
    sSQL += Funzio.ToString
    sSQL += ")"
OperDB()
If oDataSet.Rows.Count = 0 Then
     GetACATTIVITA = q.ErrSQL(100)
End If
    oConn.Close()
End Function
```

I valori che vengono passati alla funzione si riferiscono all'utente selezionato.
La funzione InsUpdACATTIVITA() provvede all'inserimento di una attivitá all'interno del database. Il suo funzionamento é analogo a quello della funzione GetACATTIVITA, con la differenza che la query SQL sará di tipo INSERT. Come parametri in ingresso vengono presi i valori che dovranno essere inseriti nel database. Viene passato inoltre un parametro Tipoagg dove viene specificato se si tratta di un inserimento (valore INS) o di un aggiornamento di dati nella tabella. Nelle funzioni non sono presenti particolari sistemi di controllo riguardanti la correttezza dei dati in quanto tutte le scelte effettuate dall'utente vengono effettuate tramite menu a tendina con valori preimpostati. Questo garantisce la correttezza dei dati forniti dall'utente.
La chiamata delle funzioni di caricamento per la visualizzazione dei dati viene specificata nello sviluppo del progetto in Visual Basic.NET. Associato all'oggetto utilizzato per visualizzare i dati viene specificata la funzione da chiamare.

Capitolo 5

Conclusioni

Nello sviluppo d questa tesi si é giunti alla costruzione del sito Abilitazioni USL fornendo la possibilitá di eseguire tutte le operazioni specificate nei capitoli iniziali di questa tesi.
Lo sviluppo effettuato utilizzato Visual Studio.NET ha permesso di ottenere un buon livello di astrazione, rendendo l'utilizzazione da parte dell'utente abbastanza intuitiva e rapida. Infatti si é riusciti ad introdurre numerose funzionalitá in un numero relativamente piccolo di pagine a cui é possibile accedere.
Un altro importante aspetto dello sviluppo del sito é dovuto alla scarsa possibilitá di salvare dati incoerenti. La scelta di utilizzare esclusamente menu a tendina porta ad un salvataggio di dati nel database senza errori o incongruenze.
Gli aspetti da migliorare nel sito Abilitazioni USL derivano principalmente dalla struttura grafica molto semplice. Infatti per ottenere Questo risultato é necessario caricare ad ogni cambiamento di pagina numerosi dati, effettuando solitamente una connessione al database.
Un ulteriore aspetto da migliorare deriva dalla presenza di numerose tabelle. Infatti dopo un prolungato utilizzo le quantitá di dati presenti potrebbero diventare notevoli. Potrebbe essere quindi introdotto un sistema di archiviazione in modo da snellire le tabelle maggiormente usate mantenendo in quest'ultime i dati piú recenti. In ultima analisi il progetto potrebbe essere esteso introducendo un sistema di login degli utenti. Questa funzionalitá é stata predisposta nel nostro progetto con funzioni eseguite prima del caricamento della pagina che

devono essere implementate.

Bibliografia

[DM02] K. Sharkey D. McKenzie. *Visual Basic.NET*. Apogeo Editore, 2002.

[GP07] M. Tripolini G. Pirou. *Access 2007, Guida Completa*. Apogeo Editore, 2007.

[JRG02] P. N. Weinberg J. R. Groff. *SQL: the complete reference*. McGraw-Hill, 2002.

[LM06] M.MacDonald L. Moroney. *Pro ASP.NET 2.0 in VB 2005*. Apress, 2006.

[MA02] C. Garret M. Ahmed. *ASP.NET Web Developers Guide*. Syngress Publishing, 2002.