Università degli Studi di Padova

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea in Informatica

Tesi di laurea triennale

Valutazione e utilizzo di nuove tecnologie Microsoft .NET 3.0 in ambito aziendale

Relatore:
Chiar.mo Prof. Massimo Marchiori

Candidato:
Alberto De Bortoli

Anno Accademico 2007/2008

Ai miei genitori
per tutto il supporto che mi hanno
dato nel periodo universitario.

Università degli Studi di Padova
Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea in Informatica
Tesi di Laurea Triennale

Valutazione e utilizzo di nuove tecnologie Microsoft .NET 3.0 in ambito aziendale

Relatore: **Prof. Massimo Marchiori** .....

Laureando: **Alberto De Bortoli** .....

Anno Accademico 2007-2008

Indice

Introduzione

Un problema importante all'interno di un'azienda sviluppatrice di soluzioni software è rimanere aggiornati e sempre al passo con le nuove tecnologie. In ASI questo aspetto è molto sentito e non a caso le attività di studio di nuove tecnologie sono state oggetto di vari stage. Il tempo da dedicare alle attività di aggiornamento all'interno di una azienda è talvolta difficile da trovare, soprattutto pensando ad un personale tecnico saturo di lavoro, progetti e demo con clienti. Nel 2007 ben 3 stage, effettuati da 3 stagisti diversi me compreso, sono vertiti sulla tecnologia .NET 3.0 per aiutare lo staff ASI a sviluppare soluzioni software più innovative.

Lo stage, durato complessivamente 300 ore, ha avuto come obiettivo lo studio critico e lo sviluppo di una applicazione in forma prototipale utilizzando la tecnologia studiata. In questa tesi sarà possibile vedere come si è evoluta l'attività di stage attraverso analisi, progettazione, realizzazione e testing. Durante tutte queste fasi si è cercato di mettere in pratica il più possibile i concetti assimilati nel corso di Ingegneria del software, documentando ogni attività. Sarà inoltre possibile capire quali sono state le difficoltà maggiori e come si è cercato di superarle.

Nel primo capitolo verrà mostrato il contesto aziendale all'interno del quale sono stato inserito. Nel secondo verranno illustrati lo scopo, il progetto di stage e la pianificazione delle attività. Nel terzo capitolo verrà fornita una panoramica sulle tecnologie già conosciute ed apprese che mi sono servite come base per poter realizzare il progetto. Il quarto capitolo, più corposo, riguarderà il progetto vero e proprio. Il linguaggio utilizzato sarà più tecnico rispetto ai precedenti due capitoli in quanto si cercherà di spiegare cos'è stato fatto materialmente e quali risultati si sono ottenuti. Il quinto capitolo invece

vertirà su una breve analisi retrospettiva e su considerazioni finali.

Infine in appendice è presente il glossario, il quale chiarisce alcuni termini tecnici. Le parole presenti in questa tesi, sottolineate la prima volta e successivamente scritte in corsivo, si trovano nel glossario.

Capitolo 1

Realtà aziendale e tecnologie

In questo capitolo verrà brevemente introdotta l'azienda presso la quale è stato effettuato lo stage oggetto di tesi. Verranno poi mostrati il contesto all'interno del quale si è operato e l'ambiente di lavoro.

1 ASI

ASI, dalla data della sua costituzione, si è progressivamente e concretamente affermata sul mercato come azienda veneta di riferimento nell'offerta di soluzioni e servizi nell'ambito dell'*Information & Communication Technology*.

Figura 1.1: logo ASI

1.1 L'azienda

ASI, nel suo quasi ventennio di storia, ha sempre puntato a intensificare la collaborazione con il leader mondiale del software Microsoft. Dopo aver ottenuto la prestigiosa qualifica di *Gold Certified Partner*, che rappresenta l'elite dei Partner Microsoft, ASI ottiene un ulteriore riconoscimento delle proprie competenze nella progettazione e nello sviluppo sulle tecnologie Microsoft: la certificazione della competenza *Networking Infrastructure Solutions*, che si aggiunge a quelle già conseguite. Le competenze certificate da Microsoft risultano essere le seguenti:

- ISV (Independent Software Vendor) Software Solutions
- Information Worker Solutions
- Networking Infrastructure Solutions

Riconoscimenti che premiano l'elevato grado di competenze e di abilità acquisite nel corso degli anni da ASI nel progettare e sviluppare soluzioni applicative d'eccellenza basate sulle migliori tecnologie Microsoft. Tra i prodotti spiccano *plain portal*®, *plain documentale*®, *plain e-commerce*®, *plain agenti*®, *plain sincro*®.

1.2 Visione

Le imprese di qualunque settore tendono a realizzare i propri obiettivi. Per raggiungerli si dedicano in maniera preponderante alla loro attività caratteristica pur investendo tempo e risorse in altre attività accessorie che possono amplificare, migliorare e rendere più redditizia l'attività core. L'*Information & Communication Technology* (ICT) può essere visto come un insieme di attività accessorie che contribuiscono al successo delle imprese. Dalla qualità di questo contributo e dal beneficio concreto che le imprese ne traggono dipende il presente e il futuro del settore IT.

1.3 Missione

Si dice che a decidere gli scambi sui mercati finanziari di tutto il mondo siano sempre più spesso le macchine perché più veloci e più informate di qualsiasi operatore umano. Ciò che spesso si tende a dimenticare quando si parla di tecnologia e di computer è che dietro al freddo calcolo delle cosiddette "macchine" ci sono gruppi di persone che progettano soluzioni sempre più efficaci per indurre le macchine stesse a pensare in un determinato modo. La differenza sostanziale è che se gli uomini che stanno "dietro alle macchine" hanno un pensiero puro, l'agire delle macchine sarà di conseguenza puro, con immensi vantaggi economici per chi ne usufruisce.
La nostra è un'epoca in cui la vita delle aziende è affidata alle grandi capacità di intervento delle macchine e ciò porta facilmente a definire che la specializzazione di chi delle macchine ne progetta il pensiero è tanto importante quanto il lavoro delle macchine stesse, se non di più. ASI è un'azienda che ha investito sulla presenza delle persone e sulle loro competenze professionali realizzando soluzioni software che pensano in modo umano per migliorare il lavoro umano. Il team ASI è composto da cinquanta specialisti tra cui ingegneri, tecnici informatici ed esperti di vari settori merceologici che giornalmente progettano, programmano e producono soluzioni innovative in sincronia con le necessità del mercato e dei clienti.

1.4 Certificazione

Di seguito sono elencati i rapporti di partnership più prestigiosi e le certificazioni aziendali, di prodotto e professionali raggiunte da ASI:

Rapporti di partnership

Figura 1.2: IBM Business Partner

ASI è Business Partner di IBM relativamente all'hardware e al software, in particolare sulle seguenti linee di prodotto: eServer iSeries, Storage Hardware, xSeries, Personal Computer, Printers - Lotus, Websphire.

Figura 1.3: Microsoft Gold Certified Partner

ASI è Gold Certified Partner di Microsoft. I Microsoft Gold Certified Partners rappresentano l'elite dei Business Partners Microsoft e hanno ottemperato ai requisiti più stringenti di Microsoft.

Figura 1.4: CheckPoint Solution Partner

ASI è Solution Partner di Check Point, relativamente ai prodotti di security:
- Firewall e VPN

Certificazioni professionali

Il personale di ASI ha conseguito Certificazioni professionali da parte di IBM e Microsoft che garantiscono l'alta professionalità del team.

Figura 1.5: IBM Certified & Microsoft Certified Systems Engineer

Certificazioni di prodotto

Il logo, rilasciato da Microsoft a *plain*® (per le applicazioni web based), riconosce che la soluzione, certificata da Veritest Corp., supporta pienamente le capacità dei web service e sfrutta i benefici del modello di programmazione .NET quali il supporto multilinguaggio, un livello di sicurezza incrementato ed un'elevata flessibilità.

2 Contesto generale

Il progetto di stage nasce dal bisogno aziendale di sviluppare applicazioni sfruttando tecnologie recenti e innovative. ASI offre un ricco listino di prodotti software dedicati alle aziende. Tale listino risponde al nome di **plain®** e vi sono presenti applicazioni per la gestione di aziende commerciali, manifatturiere, per la gestione della logistica, della documentazione, degli agenti di vendita e dell'e-commerce.

Lo staff ASI raramente ha possibilità di essere aggiornato riguardo alle nuove tecnologie che dovrebbe utilizzare per lo sviluppo, e lo scopo dello stage è stato aiutarlo sotto questo aspetto.

3 L'ambiente

Per svolgere lo stage è stata messa a disposizione da parte dell'azienda una postazione di lavoro. Questa è situata all'interno della sala dedicata agli stagisti di ASI, composta in tutto da 4 postazioni. Durante il periodo di stage tutte le postazioni erano occupate da stagisti. Sono servite inoltre le tecnologie da studiare e da utilizzare per lo sviluppo dell'applicativo richiesto.

3.1 Strumenti hardware

Il PC messo a disposizione ha le seguenti carrateristiche:

- Sistema Operativo: Windows XP Professional ® - Service Pack 2
- RAM: 1024 MB
- Processore: Pentium IV 2100Mhz

È stata inoltre necessaria una interazione con il sistema AS/400 presente in azienda per operare su Database.

3.2 Tecnologie

Le tecnologie utilizzate (apprese durante lo stage o già conosciute) sono molteplici.

- Ambiente di sviluppo e Framework
  - Visual Studio 2005
  - Visual Studio 2008 beta
  - Framework .NET 2.0
  - Framework .NET 3.0
- Linguaggi di programmazione, di markup, d'interazione con il database
  - C#
  - XML e XAML
  - SQL
- Database
  - SQL
- Stesura documentazione
  - Microsoft Office Word 2007

Una descrizione più dettagliata delle tecnologie si trova nel terzo capitolo.

Capitolo 2

Progetto Aziendale

In questo capitolo sarà possibile vedere come l'attività di stage sia inquadrata all'interno di un progetto aziendale ampio e ben definito. Inoltre, verranno mostrati la pianificazione delle attività e gli obiettivi prefissati.

1 Problemi di avviamento

Il piano di lavoro dello stage effettivamente avvenuto non corrisponde con il piano di lavoro inizialmente stabilito in azienda qualche mese prima di iniziare il periodo di stage. Inizialmente era stato concordato con l'azienda uno stage che prevedesse lo sviluppo di una applicazione web per la gestione della posta in ambito aziendale. Tale applicazione voleva essere una sorta di "*wrapper*" del programma di gestione di posta elettronica *Outlook* di Microsoft e doveva essere sviluppato seguendo il "*modello a oggetti di Outlook*" messo a disposizione dal Framework 2.0 e interfacciandosi al server di posta Microsoft Exchange Server.

I primi 10 giorni sono stati impiegati a studiare il dominio del problema e stilare uno studio di fattibilità, dal quale si è dedotto che per sviluppare una applicazione che soddisfacesse le richeste dell'azienda l'unica tecnologia plausibile era tramite l'utilizzo di un Web Service, in quanto *WebDAV* e altre tecnologie sono risultate estremamente obsolete e inadeguate. Inoltre era fondamentale l'installazione di *Microsft Exchange Server 2007* come server di posta aziendale (al posto della ormai superata versione 2003) cosa che i sistemisti ASI non avevano ancora eseguito.

Purtroppo quest'ultimo è risultato un problema insormontabile al quale non si è trovata una soluzione. Il problema era semplice: *Microsoft Exchange Server 2007* richiede piattaforme a 64 bit, delle quali l'azienda era sprovvista.

Poiché sviluppare il progetto seguendo tecnologie superate non avrebbe portato di certo un beneficio all'azienda, il piano di lavoro è stato inevitabilmente cambiato radicalmente con scuse da parte di ASI per non aver pensato prima alle problematiche rivelatesi.

Le conseguenze di questo cambio di stage sono state in primis il cambio di pianificazione, che è stata da me adattata secondo le nuove esigenze su direttive del tutor aziendale, e il cambio di tutor aziendale assegnatomi, nella persona di Marino Della Puppa, in quanto persona adibita all'oggetto del nuovo stage.

Per quanto possibile le ore "perse" (40 ore) sono state recuperate superando

le 8 ore giornaliere di stage e comunque puntando all'obiettivo di portare a termine il nuovo progetto di stage assegnatomi.

2 Progetto e collocazione dello stage

Il progetto di stage ha una collocazione ben preciso all'interno dei bisogni aziendali. Non si tratta di un progetto a sé stante, ma di un caso di studio, seguito dallo sviluppo di un applicativo, che ha come scopo la facilitazione futura di sviluppo software secondo i paradigmi studiati da parte del personale aziendale.

ASI sviluppa l'insieme dei suoi prodotti software utilizzando, come si è già potuto intuire, soltanto tecnologie Microsoft. Il caso di studio da me affrontato ha come oggetto il Framework .NET.

Il .NET Framework è la parte centrale della tecnologia .NET di Microsoft. È l'ambiente per la creazione, la distribuzione e l'esecuzione di tutti gli applicativi che supportano .NET siano essi Servizi Web o altre applicazioni. .NET è corredato da una serie di strumenti di sviluppo delle applicazioni, progettati in modo da funzionare in modo integrato all'interno della piattaforma .NET. Uno dei principali strumenti è l'IDE denominato Visual Studio. Dalla nascita a oggi il Framework ha passato varie versioni, quali: 1.0, 1.1, 2.0, 3.0 e 3.5.

ASI, nel corso degli anni, ha sviluppato i proprio prodotti secondo la versione del Framework in vigore nel momento della scelta della tecnologia per tale progetto. Attualmente ASI offre prodotti sviluppati con la versione 2.0 del Framework e purtroppo anche con la versione 1.0, ormai parecchio datata. Al momento dello stage la versione 3.5 risultava ancora in fase beta e quindi non è stata oggetto di analisi, al posto della quale si è presa in considerazione la versione 3.0. Parte di altri stage prevedeva anche il porting di applicazioni .NET 1.0 alla versione 2.0, mentre il mio stage ha avuto come scopo appunto studiare le potenzialità della versione 3.0 e quali sarebbero i pro e i contro di un imminente passaggio alla più recente tecnologia.

Oltre al caso di studio, il piano di lavoro prevedeva anche lo sviluppo di un applicativo, in forma prototipale, slivuppato con .NET 3.0, ed in particolare con Windows Presentation Foundation e Windows Communication Foundation.

Scopo principale del prototipo era creare un sistema che si interfacciasse

con il database aziendale contenente le informazioni dei clienti e presentarle in forma tabellare in maniera facilmente consultabile. Il programma doveva inoltre permettere l'accesso a tali informazioni tramite internet e avrebbe avuto come utenti finali i rappresentanti aziendali che, anche a distanza dalla sede, potevano reperire le informazioni riguardanti i clienti in tempo reale.

3 Pianificazione delle attività

Per svolgere l'attività di stage è stato concordato di adottare un orario d'ufficio in linea con lo standard aziendale (8.30 12.30, 14.00 18.00), e di fare uso delle strumentazioni di lavoro presenti all'interno della sede.

Il tutor aziendale mi ha lasciato libera scelta nella pianificazione delle 300 ore a disposizione per lo stage consigliando delle linee guida generali per suddividere il lavoro in modo tale da avere la possibilità di apprendere le tecnologie e di conoscere realtà e tempistiche aziendali.

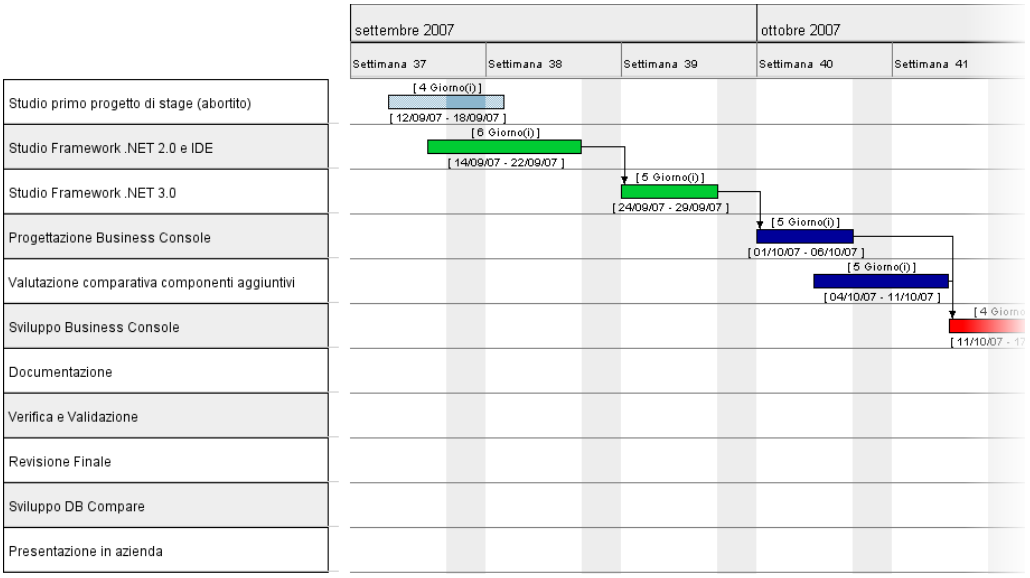


Figura 2.1: Pianificazione tramite diagramma di Gantt - parte 1

Come si può vedere dal diagramma di Gantt mostrato in figura 2.1 e 2.2, lo stage è stato pianificato in modo da coprire le attività principali di:

- **Studio del dominio e delle tecnologie**: dal 12/09 al 29/09
- **Progettazione e Valutazione componenti aggiuntivi**: dal 01/10 al 11/10
- **Sviluppo applicazione**: dal 11/10 al 17/10
- **Documentazione**: dal 15/10 al 25/10
- **Verifica e Validazione**: dal 17/10 al 26/10

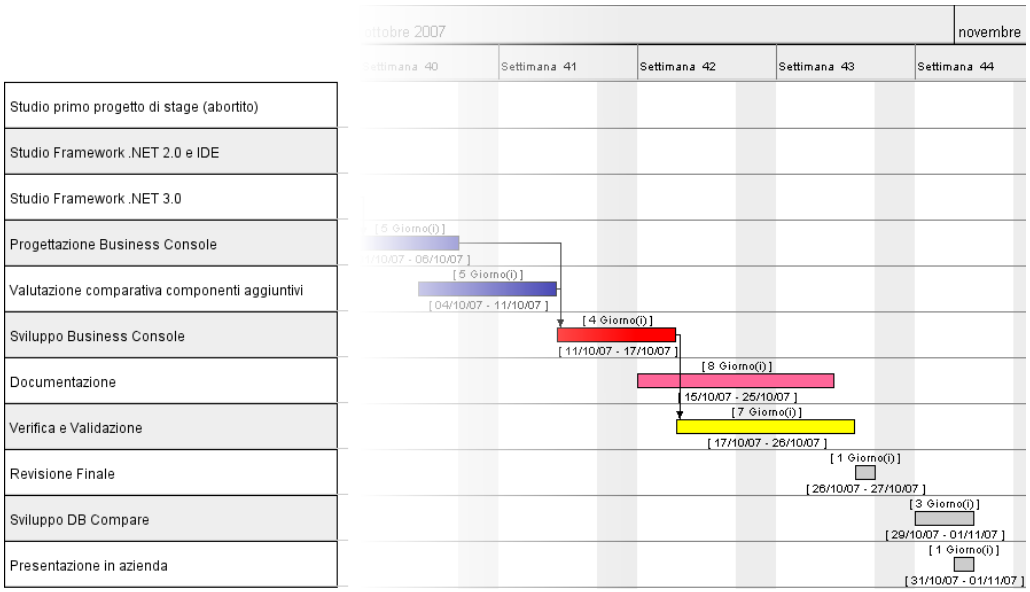


Figura 2.2: Pianificazione tramite diagramma di Gantt - parte 2

Le attività di stage sono state costantemente visionate dal tutor aziendale con cadenze giornaliere e talvolta dedicando anche un'ora della sua giornata lavorativa per seguire gli sviluppi da vicino.

Fortunatamente, grazie anche ad un impegno maggiore del previsto, ad un totale di ore lavorative giornaliere che talvolta arrivava a 10, continuando le attività per quanto possibile anche non in azienda ma a casa nei weekend, sono riuscito a portare a termine tutti gli obiettivi prefissati nei tempi previsti e talvolta addirittura con qualche giornata di anticipo. Ciò ha permesso, oltre che finire lo stage in tempi decisamente più brevi rispetto ad altri colleghi, anche di riservare circa 3 giorni per sviluppare una applicazione richiestami da ASI poiché avevo concluso il progetto di stage.
L'applicazione "**Business Console**" sviluppata verrà talvolta, nel corso della tesi, abbreviata con l'acronimo "BC", e **Batabase Compare**" con "DB Compare". Tali applicazioni verranno descritte nel quarto capitolo.

Andando nel dettaglio, si può vedere come sono state suddivise le 300 ore previste tra le varie fasi.

Fasi	Dettagli	Ore previste
Formazione	studio Framework .NET 2.0 studio IDE Visual Studio 2005 studio Framework .NET 3.0 (WPF e WCF)	100
Analisi componenti aggiuntivi e Progettazione	analisi componenti aggiuntivi di terze parti progettazione interfaccia BC progettazione web service BC	60
Sviluppo	sviluppo interfaccia BC sviluppo web service BC	60
Documentazione	stesura documenti .NET 3.0 stesura manuale BC	60
Verifica & Validazione	verifica e validazione test	20
TOTALE		**300**

Figura 2.3: Pianificazione ore

Oltre alla fase di Documentazione, in tutte le restanti fasi si è pensato di dedicare un periodo che poteva variare dal 15 al 20% delle ore, per il processo di supporto riferito alla documentazione. In questo modo è stato più semplice redigere, prima ad alto livello, e poi in forma definitiva nella relativa fase, i seguenti documenti:

- Analisi WPF
- Analisi WCF
- Valutazione Comparativa componenti aggiuntivi
- Manuale programmatore Business Console
- Manuale utente Business Console
- Consuntivo

Poiché il programma *Business Console* è stato richiesto in forma prototipale sono stati richieste anche 2 manuali distinti, uno per l'utente finale che avrebbe indubbiamente subìto delle modifiche una volta che lo staff ASI avesse modificato l'applicativo, l'altro per il programmatore che doveva essere in grado di capire la struttura del programma e come era stato progettato in maniera chiara.

3.1 Requisiti

In definitiva i requisiti iniziali dello stage (a parte la valutazione dei componenti aggiuntivi, la cui esigenza è nata in un momento successivo) sono stati tutti **obbligatori**, e sono riassunti di seguito:

- Studiare e analizzare WPF
- Studiare e analizzare WCF
- Sviluppare un applicativo in forma prototipale di certo utilizzo all'interno di ASI e che sfrutti le tecnologie studiate
- Documentare tutte le attività di stage
- Documentare gli studi effettuati su WPF e WCF

Il modello di ciclo di vita usato è di tipo incrementale. Questo è dato dal fatto che c'è bisogno di flessibilità e non si conosce bene la fattibilità del progetto, quindi in fase di codifica potrebbe esserci il bisogno di ritornare alla fase di progettazione di dettaglio.

3.2 Seminario Microsoft

Nella giornata del 18 ottobre Microsoft ha tenuto un seminario presso l'Università di Padova, nella sede del DEI (Department of Information Engineering) sulle tecnologie .NET. La scelta di parteciparvi è quindi risultata praticamente d'obbligo per tutti gli stagisti ASI che hanno visto in questa occasione un'ottima opportunità di apprendimento delle nuove tecnologie che sarebbero state utilizzate durante lo stage. Al seminario era presente Mauro Minella di Microsoft, esperto nel settore, che si è reso disponibile a chiarire problematiche riscontrate durante lo stage.

4 Obiettivi

L'attività di stage, come concepita dal CCS (consiglio corso di studi), è stata pensata prima di tutto per raggiungere importanti obiettivi formativi come:

- Capacità di svolgere in modo autonomo alcune ricerche delle soluzioni per lo sviluppo di problematiche aziendali.
- Visione di massima del ciclo di vita di un progetto aziendale.
- Capacità di rapportarsi nell'ambito di un team di lavoro e di svolgere in parziale autonomia, con la supervisione del tutor, tutte le fasi di sviluppo di un piccolo progetto software.
- Conoscenza di massima delle logiche e dei rapporti che intercorrono in un ambiente aziendale.

Per quanto riguarda i bisogni aziendali, in caso di stage concluso con successo, ASI disporrà di una ricca documentazione riguardanti la tecnologia .NET 3.0 che utilizzerà per iniziare lo sviluppo di applicazioni seguendo le linee guida illustrate. Inoltre, avrà un punto di partenza per lo sviluppo ulteriore della *Business Console* aziendale e per quanto riguarda il progetto extra riguardante il *Database Compare*.

Capitolo 3

Tecnologie e Strumenti

In questo capitolo verranno trattati gli aspetti concettuali legati alle tecnologie, alle tecniche e agli strumenti utilizzati e studiati durante l'attività di stage. Sarà quindi descritta l'architettura di Microsoft .NET Framework, le funzionalità e i miglioramenti aggiunti dalla versione 3.0, in particolare Windows Presentation Foundation e Windows Communication Foundation.

1 .NET Framework

Il .NET Framework è una piattaforma Microsoft per la creazione di applicazioni in ambiente Windows e per un rapido sviluppo di applicazioni e servizi orientati al Web.

Figura 3.1: Logo tecnologia .NET

Il .NET Framework è progettato per ottenere i seguenti obiettivi:

- fornire un ambiente di programmazione orientato agli oggetti coerente ed estensibile;
- fornire un ambiente di esecuzione del codice che minimizzi la distribuzione del software e i conflitti di versioni;
- fornire un ambiente di esecuzione del codice che permetta un'esecuzione sicura anche dei codici creati da produttori sconosciuti;
- rendere coerente l'esperienza dello sviluppatore attraverso tipi molto vari di applicazioni, quali applicazioni basate su Windows e applicazioni basate sul Web;
- generare tutte le comunicazioni in base agli standard industriali per assicurare che il codice basato su .NET Framework possa integrarsi con qualsiasi altro codice.

L'architettura .NET, a differenza di quanto accade per Java, non è legata ad un linguaggio specifico ma è multilinguaggio: si può scrivere un programma in un qualunque linguaggio supportato, compilarlo in un linguaggio intermedio denominato MSIL ed eseguirlo su di una piattaforma .NET.
Gli oggetti scritti in un linguaggio, infatti, possono essere compilati e riutilizzati come moduli in programmi scritti in un altro linguaggio. I programmatori

possono, quindi, adottare i linguaggi con cui hanno maggiore dimestichezza mantenendo comunque la possibilità di eseguire debugging tra servizi o componenti scritti in linguaggi di programmazione differenti.

Poiché la piattaforma .NET Framework supporta l'integrazione di molti linguaggi di programmazione, gli sviluppatori hanno la possibilità di scegliere il linguaggio di programmazione più adatto alle diverse attività.
.NET nella sua versione per sviluppatori (SDK) contiene i compilatori per C#, Visual Basic .NET , JavaScript, J# e oltre a questi linguaggi forniti da Microsoft, sono utilizzabili altri linguaggi come Delphi e Lisp.

Se si sviluppa e pubblica un servizio Web XML, .NET Framework fornisce, inoltre, un insieme di classi conformi a tutti gli standard di comunicazione sottostanti, quali SOAP, WSDL e XML. Mediante queste classi, è possibile individuare la logica del servizio, senza preoccuparsi delle infrastrutture di comunicazione richieste dallo sviluppo di software distribuiti.

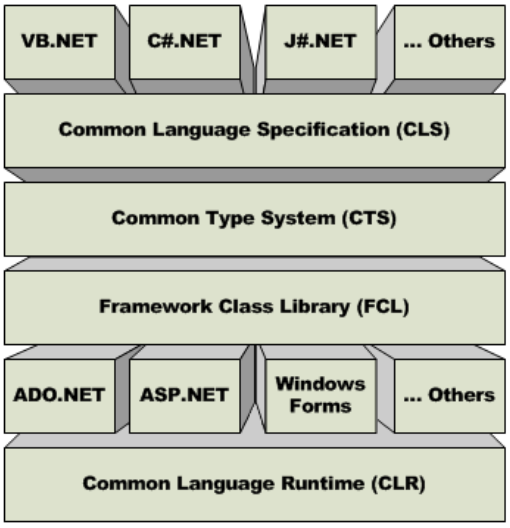


Figura 3.2: Architettura .NET Framework 2.0

.NET Framework presenta due componenti principali:

- *Common Language Runtime (CLR)*
- *libreria di classi .NET Framework*

1.1 Common Language Runtime

Il *Common Language Runtime* gestisce la memoria (garbage collection), l'esecuzione di *thread*, l'esecuzione del codice, la verifica della sicurezza del codice, la compilazione e altri servizi di sistema.
I compilatori di linguaggio destinati al .NET Framework producono come risultato un codice eseguibile dal *runtime*, al quale deve essere ovviamente conforme.
Questo tipo di codice aderente alle regole e alle strutture dati offerte dal CLR è chiamato managed code (codice gestito) ed ha, quindi, accesso al *runtime*. Codice macchina insicuro (perché scavalca il *runtime*) può essere generato dai compilatori, e in questo caso si parla di unmanaged code (codice non gestito) che però non può appoggiarsi al *runtime*.

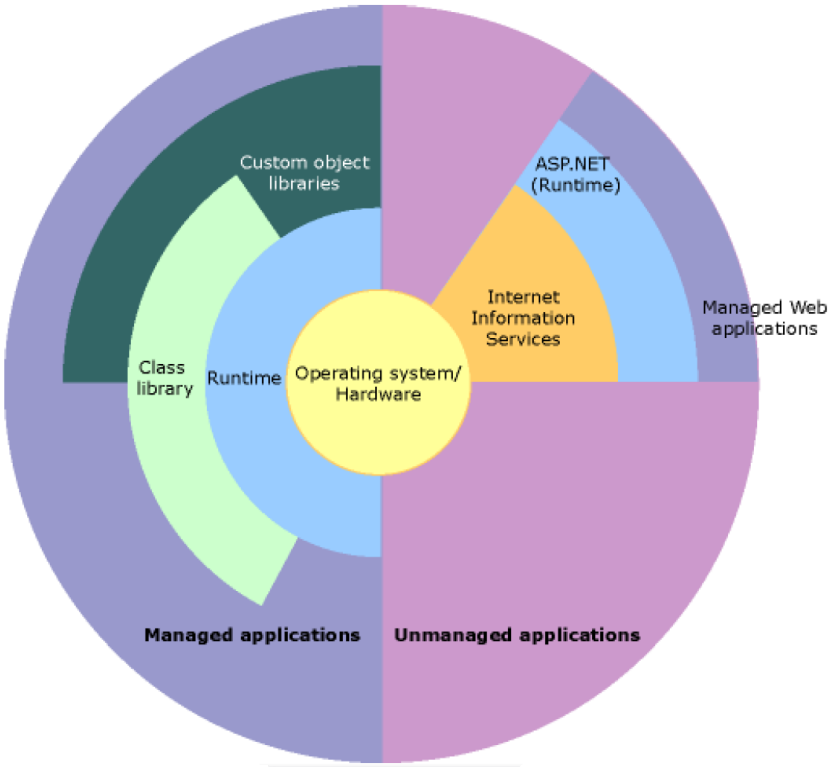


Figura 3.3: Rappresentazione a torta della struttura .NET Framework

Il ruolo critico del *Common Language Runtime*, e ciò che realmente lo contraddistingue, è il fatto che fornisce un *ambiente di runtime unificato comune a tutti i linguaggi di programmazione*. I linguaggi la cui destinazione è il CLR sono compilati in Microsoft Intermediate Language (MSIL) (detto anche CIL,

Common Intermediate Language). Le applicazioni vengono compilate come file PE (Portable Executable) e DLL in modo da risultare simili a qualsiasi altra applicazione basata su Windows. Il codice MSIL contenuto in questi file, viene poi compilato localmente tramite **JIT** (Just In Time) in base alle istruzioni del computer al momento dell'esecuzione. Ciò significa che i linguaggi compatibili con le specifiche CLS (Common Language Specification: definisce gli elementi che un linguaggio compatibile con .NET deve fornire al sistema) la cui destinazione è il CLR appariranno al *runtime* come qualsiasi altro linguaggio e quindi si comporteranno e verranno recepiti come linguaggi di pari livello. Ad esempio, un programma Visual Basic disporrà delle medesime funzionalità di base di un programma Visual C# e perfino di un programma C++ gestito.

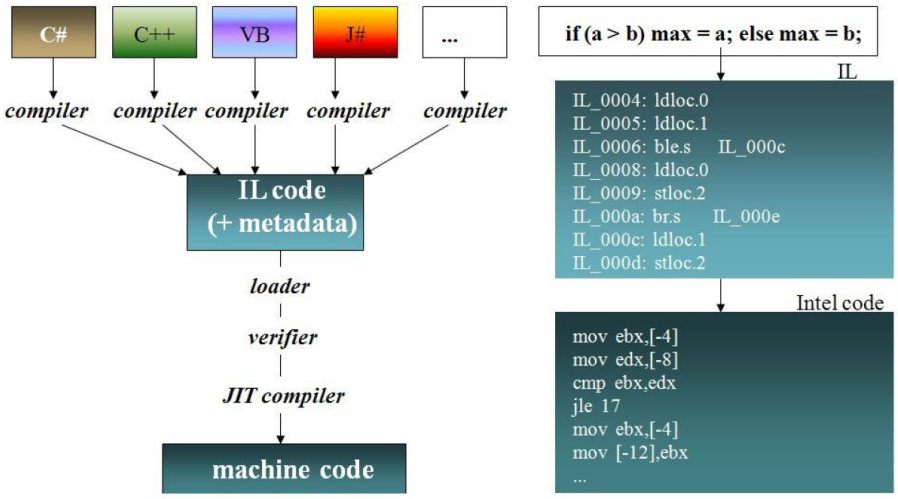


Figura 3.4: Esecuzione del codice in .NET

Classico programma "ciao mondo" scritto in istruzioni CIL.

```
.assembly CiaoMondo{} .method public hidebysig static void Main()
cil managed {
   .entrypoint
   ldstr ''Ciao mondo.''
   call void [mscorlib]System.Console::WriteLine(string)
   ret
}
```

1.2 Libreria di classi .NET Framework

La libreria di classi .NET Framework è un insieme di tipi riutilizzabili che si integrano strettamente con il *Common Language Runtime*. La libreria di classi è orientata agli oggetti e fornisce tipi dai quali il codice gestito può derivare le funzionalità. In questo modo non solo viene semplificato l'utilizzo dei tipi .NET Framework ma viene anche ridotto il tempo necessario all'apprendimento delle nuove funzionalità di .NET Framework in quanto tutti i linguaggi hanno a disposizione la stessa libreria di classi.

La libreria di classi .NET Framework è un insieme di tipi riutilizzabili che si integrano strettamente con il *Common Language Runtime*. La libreria di classi è orientata agli oggetti e fornisce tipi dai quali il codice gestito può derivare le funzionalità. In questo modo non solo viene semplificato l'utilizzo dei tipi .NET Framework ma viene anche ridotto il tempo necessario all'apprendimento delle nuove funzionalità di .NET Framework in quanto tutti i linguaggi hanno a disposizione la stessa libreria di classi.

Nei tipi di .NET Framework è utilizzato uno schema di denominazione con sintassi a punti che definisce una gerarchia. Secondo questa tecnica i tipi correlati vengono raggruppati in spazi dei nomi, in modo da semplificarne le ricerche e i riferimenti. La prima parte del nome completo, fino al primo punto da destra, è il nome dello spazio dei nomi. L'ultima parte del nome è il nome del tipo. *System.Collections.ArrayList*, ad esempio, rappresenta il tipo *ArrayList*, appartenente allo spazio dei nomi *System.Collections*. I tipi di *System.Collections* possono essere utilizzati per manipolare insiemi di oggetti.

2 .NET Framework 3.0

Microsoft .NET Framework versione 3.0 (precedentemente noto come WinFX) è il nuovo modello di programmazione a codice gestito per Windows. La nuova versione combina tutte le potenzialità di .NET Framework 2.0 con le nuove tecnologie per la creazione di applicazioni caratterizzate da funzionalità grafiche avanzate per l'utente, perfetta interoperabilità tra le diverse tecnologie e supporto per un'ampia gamma di processi aziendali.

Alcune parti di .NET Framework 2.0 risultano decisamente obsolete rispetto ai nuovi componenti della versione 3.0, ma comunque le tecnologie della versione 2.0 rimangono un aspetto fondamentale anche in questa nuova release.

Utilizzando .NET 3.0 gli sviluppatori possono utilizzare Windows Presentation Foundation anziché Windows Form per creare un'interfaccia grafica Windows nativa e in genere preferiranno Windows Comunication Foundation ai Web Service ASP.NET, .NET Remoting o Enterprise Services.

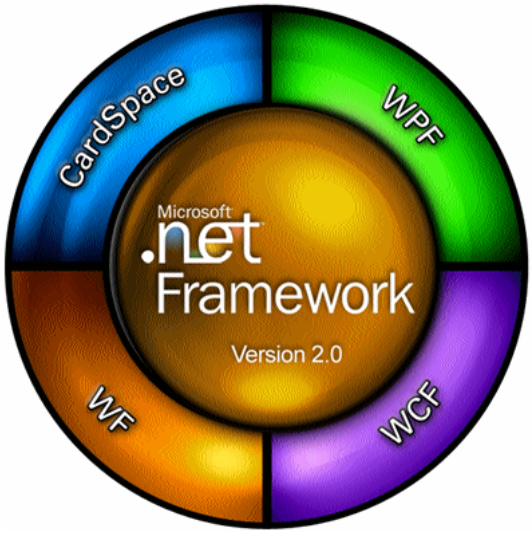


Figura 3.5: L'architettura del Framework .NET 3.0

Come illustrato nella figura, la versione di .NET Framework 3.0 è basata sulla release precedente 2.0, ma introduce quattro nuovi componenti: *Windows Communication Foundation*, *Windows Presentation Foundation*, *Windows Workflow Foundation* e *Windows CardSpace*.

Visto che .NET Framework 3.0 condivide numerosi componenti con la versione precedente, il *Common Language Runtime* e le *Base Class Library* rimangono, quindi, alla versione 2.0, mentre il numero di versione 3.0 si applica a tutti gli *assembly runtime*, e di riferimento, per Windows Communication Foundation, Windows Presentation Foundation, Windows Workflow Foundation e Windows CardSpace.

Per il progetto di stage sono stati studiati i primi 2 componenti (*Presentation* e *Communication*) e per questi nel seguito della tesi verrano usati spesso gli acronimi "WPF" e "WCF". Segue una descrizione di essi.

2.1 Windows Communication Foundation

Esiste un gran numero di tecnologie per la comunicazione tra applicazioni come Web service ASP.NET, .NET Remoting, Enterprise service ecc. ugualmente valide e ognuna riveste un ruolo specifico.

L'obbiettivo di WCF, è fornire un'unica piattaforma per la comunicazione tra applicazioni basata su servizi interoperabili. Nell'ambiente .NET Framework 3.0, la maggior parte delle applicazioni comunicherà tramite WCF, invece di utilizzare una delle tecniche elencata precedentemente.

WCF segue un approccio alle comunicazioni esplicitamente orientato ai servizi, anziché fornire un contesto trasparente alle comunicazioni tra gli oggetti, WCF astrae la complessità interponendo il modello a servizi tra le parti comunicanti. Uno degli obiettivi di questo approccio è sciogliere alcuni degli accoppiamenti rigidi che possono presentarsi nei sistemi a oggetti distribuiti, riducendo la possibilità di errori e facilitando la modifica delle interazioni.

2.2 Windows Presentation Foundation

WPF è rivolto alla gestione dell'interfaccia utente. Consente di disaccoppiare la progettazione della parte grafica vera e propria da quella della parte funzionale, in modo da consentire ai designer di occuparsi della presentazione, svincolati dagli eventuali limiti intrinseci dell'ambiente di esecuzione, e agli sviluppatori software di concentrarsi sugli elementi legati alla logica da implementare nell'applicazione. Per risolvere questo problema, WPF si fonda su XAML (eXtensible Application Markup Language), un linguaggio basato su XML che consente di specificare l'interfaccia utente in modo dichiarativo, anziché a livello di codice procedurale.

3 Visual Studio 2005

Visual Studio è un insieme completo di strumenti di sviluppo per la creazione di applicazioni Web ASP.NET, Web Service XML, applicazioni desktop e applicazioni mobili. Visual Basic, Visual C++, Visual C# e Visual J# utilizzano lo stesso ambiente di sviluppo integrato (IDE), che consente la condivisione di strumenti e facilita la creazione di soluzioni in linguaggio misto. Questi linguaggi sfruttano le funzionalità di .NET Framework, che fornisce l'accesso a tecnologie chiave in grado di semplificare lo sviluppo di applicazioni Web ASP e Web Service XML.
Gli strumenti a disposizione comprendono i Web Form che sono una tecnologia ASP.NET che consente di creare pagine Web programmabili. Queste danno luogo a codice HTML e script compatibili con i browser per consentire la visualizzazione delle pagine con qualsiasi browser e su qualsiasi piattaforma. Con i Web Form è possibile creare pagine Web trascinando e rilasciando controlli nella finestra di progettazione e aggiungendo in seguito il codice, analogamente a quanto accade per la creazione dei form di Visual Basic.

Per utilizzare le componenti aggiunte dal .Net Framework 3.0 con Visual Studio 2005 bisogna scaricare ed installare un apposito modulo messo a disposizione sul sito di Microsoft. Il motivo risiede nella data di rilascio della piattaforma: Visual Studio è uscito prima del rilascio di .NET Framework 3.0.

Capitolo 4

Attività di Stage

In questo capitolo verrà trattata l'attività di stage vera e propria. Dopo un'analisi delle tecnologie apprese e necessarie allo sviluppo del progetto verrà illustrato l'applicativo sviluppato in forma prototipale di *Business Console*.

1 Windows Presentation Foundation

1.1 Descrizione generale

Windows Presentation Foundation, nome in codice Avalon, è una libreria di classi del Framework .NET (introdotta con la versione 3.0) per lo sviluppo dell'interfaccia grafica delle applicazioni. WPF rappresenta la nuova *API* Microsoft presente nel .NET Framework 3.0.
Inizialmente era stata pensata come base per tutte le applicazioni di Windows Vista, ma nel 2004, con il "reset" dello sviluppo della nuova versione di Windows, è stato deciso di renderla disponibile anche per Windows XP e le applicazioni di Windows Vista utilizzano ancora le classiche API.

1.2 Caratteristiche

WPF consente la creazione di interfacce grafiche accattivanti e più moderne rispetto alle GUI conosciute fino a Windows XP. L'innovazione principale di WPF è la rimozione di ogni legame con il modello di sviluppo tradizionale di Windows, introdotto con la versione 1.0 del sistema operativo. Tutti i controlli sono stati riscritti (non si appoggiano più a quelli della libreria "user") e lo stesso meccanismo basato su scambio di messaggi, cuore del modello di programmazione di Windows, è stato abbandonato.
WPF è basato su un sistema di grafica vettoriale che può anche sfruttare le potenzialità tridimensionali delle moderne schede grafiche. Inoltre il meccanismo per la creazione dell'interfaccia utente è il linguaggio XAML (eXtensible Application Markup Language).

1.3 XAML

XAML (acronimo di eXtensible Application Markup Language, e pronunciata "Zammel") è un linguaggio di markup basato su XML, utilizzato per descrivere l'interfaccia grafica delle applicazioni basate sulla libreria WPF.
Il linguaggio XAML si basa sugli oggetti contenuti nel *Common Language Runtime* e sulle loro proprietà o eventi; tramite esso è possibile descrivere gli oggetti in maniera molto più semplice e intuitiva di quanto si sia fatto con l'utilizzo delle classiche API e con codice dichiarativo.

Esempio di linguaggio di mark up, in generale ogni elemento corrisponde ad un oggetto, ogni attributo corrisponde ad una proprietà:

```
<object>
    <child property="x" property="y">
        <child.property>
            <class property="u" property="v"/>
        </child.property >
    </child>
    <child>
    </child>
</object>
```

1.4 Nuovo concetto di GUI

Quanto concerne la creazione delle nuove GUI è il punto forte di WPF. Poiché l'interfaccia grafica viene definita tramite XAML, è permessa una scissione tra il "*business logic*" e il "*presentation logic*". Cio è facilmente capibile facendo un raffronto con il mondo Web 2.0, nel quale la parte di contenuto viene resa tramite il linguaggio HTML, mentre quella di presentazione tramite CSS. Con WPF lo strato logico viene reso con un qualsiasi linguaggio messo a disposizione dal Framework (C#, VB.NET, J#...), mentre lo strato grafico viene reso tramite linguaggio XAML.

Il paradigma innovativo per lo sviluppo di applicazione si può riassumere nella seguente locuzione:

"**Application = Code + Mark Up**"

1.5 Compromesso tra Web e WinForm

Con WPF si riduce notevolmente il divario tra applicazioni e/o pagine Web e applicazioni *WinForm*, in quanto vi è un tentativo sempre maggiore di fondere l'interfaccia grafica tra i due mondi, indipendentemente dal codice usato per il code-behind. Tale concetto è espresso in figura 4.2:
Inoltre è consentito facilmente eseguire il porting di una applicazione scritta per essere eseguita stand-alone su ambiente Web.

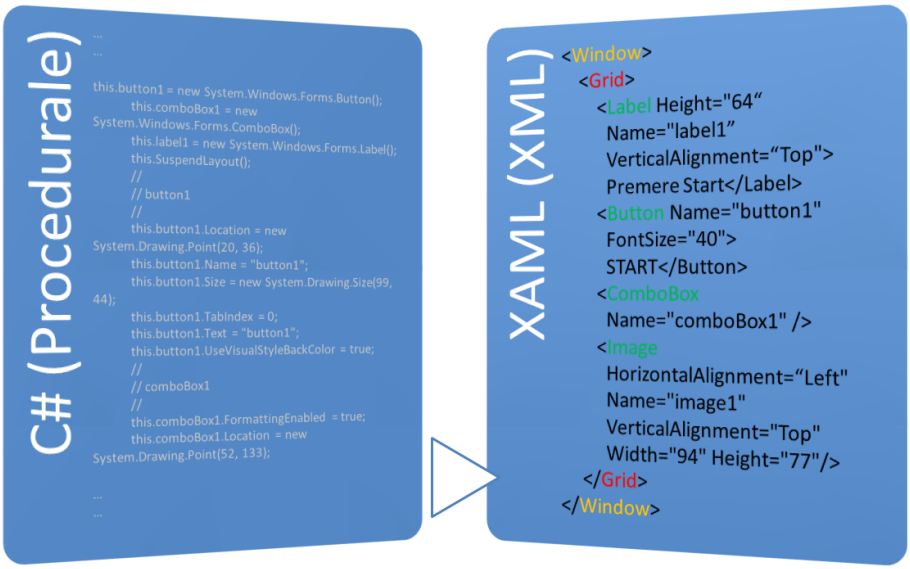


Figura 4.1: Evoluzione dalla descrizione dell'interfaccia grafica tramite codice procedurale a codice XML

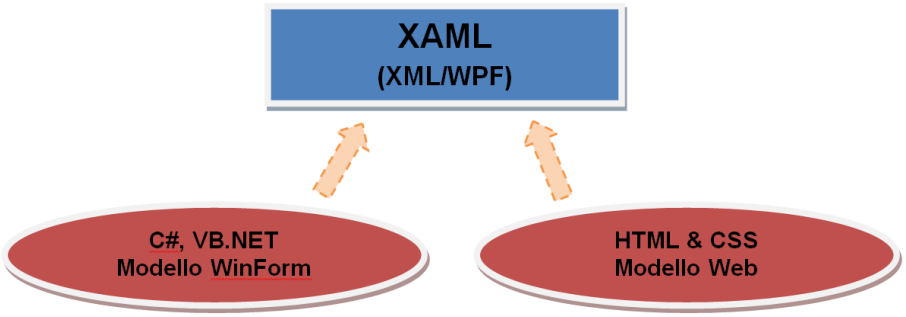


Figura 4.2: Concetto di fusione tra WinForm e Web

1.6 WPF/E (Silverlight)

WPF/E è un sottoinsieme di WPF e consente la realizzazione di applicazioni web dotandole di immagini vettoriali, animazioni, trasformazioni e che supportino gli attuali formati audio e video. WPF/E fonda le basi sull'utilizzo di JavaScript e l'architettura di WPF. La definizione del layout è infatti affidata a XAML, grazie al quale è facile la sua comprensione da parte dello sviluppatore e una sua semplice manipolazione con strumenti di disegno. Molti concetti di XAML sono infatti validi ad eccezione delle markup extension, che al momento non sono supportate e gli oggetti utilizzabili sono limitati al subset di WPF messo a disposizione.

Silverlight è il nome in codice di WPF/E. Lo scopo commerciale di questa tecnologia è quello di competere con le applicazioni web create con il software grafico *Adobe Flash*®, allo stato dell'arte il più diffuso per lo scopo. Uno dei più grandi limiti nello sviluppo di applicazioni web è dovuto al linguaggio (X)HTML che non si è evoluto negli anni ed è rimasto quasi del tutto invariato non allineandosi alle esigenze di mercato. Da qui l'idea di Microsoft di introdurre una nuova tecnologia legata al mondo web che permettesse di abbattere i limiti legati all'HTML e legarla ad una tecnologia già consolidata come il .NET Framework. Silverlight si presenta come plugin per browser internet cross-platform e cross-browser.
L'architettura interna fa utilizzo di molti linguaggi nuovi e vecchi per rendere estendibile questa nuova metodologia di sviluppo, tra i quali Javascript, XAML, WCF, ASP.NET, AJAX e HTML. Importante notare che tali applicazioni eseguono in una SandBox che impedisce al codice di accedere alle risorse del client, abbattendo totalmente i rischi di creare e ricevere codice maligno in quanto viene fornito un Isolated Storage, che permette di salvare sul client informazioni accessibili dall'applicazione Silverlight.

Il ciclo di vita di una richiesta in un'applicazione parte da una pagina HTML contenente Il codice Javascript necessario ad istanziare il plug-in che mostrerà a video il render del contenuto presente all'interno del file XAML, in maniere molto simile a come avviene per i file sviluppati con *Adobe Flash*.
Come per le applicazioni AJAX, gli eventi vengono gestiti tramite Javascript, così da comunicare facilmente con il server. Il codice seguente mostra una pagina HTML con il codice Javascript necessario ad instanziare l'applicazione e a settarne il focus.

```
<body
onload="document.getElementById('SilverlightControl').focus()">
    <div id="SilverlightControlHost" align="center">
        <script type="text/javascript">
            createSilverlight();
        </script>
    </div>
</body>
```

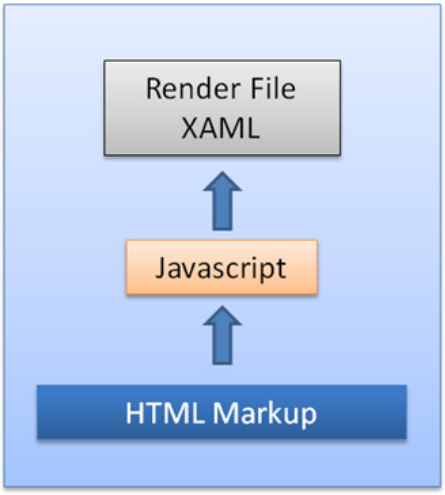


Figura 4.3: Schematizzazione ciclo di vita applicazione web

La funzione Javascript *CreateSilverlight()* ci viene creata da Visual Studio, ed crea l'object per il browser e imposta proprietà di base.

1.7 Progetto di tipo Windows Application

Un progetto di tipo Windows Application (WPF) si presenta come in fig. 4.4. Nella scheda "Esplora soluzioni" si può notare che ogni file *.xaml* ha un corrispettivo file *.xaml.cs* che viene visualizzato ramificato dal primo. Si può pensare al file *.xaml* come corrispondente del file *.Designer.cs* dei progetti Windows Application classici, riscritto in linguaggio di marcatura.
Cider permette dalla scheda "Design" di trascinare e disegnare oggetti, e dalla scheda "Xaml" di editare e scrivere nel file *.xaml* il codice Xaml che rappresenta gli oggetti. Premendo sul pulsante in basso a destra possiamo passare all'editor XML e vederne il codice generato:

```
<Window x:Class="WindowsApplication1.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="WindowsApplication3" Height="300" Width="300"
    >
    <Grid>

    </Grid>
```

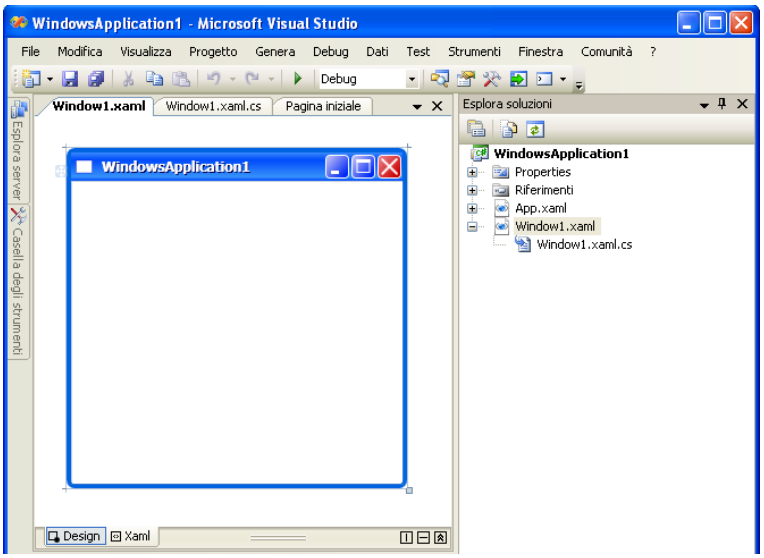


Figura 4.4: Nuovo progetto di tipo Windows Application in Visual Studio 2005

```
</Window>
```

Root dell'albero XML è l'elemento principale *Window*, la finestra, e vengono specificati come attributi il titolo (*Title*), l'altezza (*Height*) e la larghezza (*Width*). Come figlio troviamo un *Grid*, che rappresenta una griglia simile alle tabelle HTML.

Essendo un documento XML è necessario che questo sia ben formato (*well-formed*) e validato, perciò nella root sono specificati due namespace: uno di default ed uno con prefisso "*x*". Il primo va ad indicare che i tags che andiamo a specificare appartengono ad un set di namespace contenuti nell'assembly *PresentationFramework.dll*. Il secondo invece va ad indicare informazioni necessarie al parser XAML.

L'attributo *x:Class* è utile al generatore di codice in fase di compilazione, poiché indica il code-behind di questo file. L'elemento root che specifichiamo per ogni file XAML è il tipo dal quale la classe autogenerata eredita. Il file *.xaml.cs* contiene infatti il seguente codice:

```
public partial class Window1 : Window {
    public Window1()
```

```
    {
        InitializeComponent();
    }
}
```

1.8 Compilazione dei file XAML

Si noti che la classe eredita da Window, dal momento che da essa prendiamo tutte le sue caratteristiche, ma si estende con controlli e codice personalizzato. Il metodo *InitializeComponent*, definito nel secondo file di questa classe, di norma non viene visualizzato e modificato perché autogenerato dal compilatore. All'interno della cartella *obj/debug* del progetto è possibile trovare un file con estensione *.g.cs*, che depurato di informazioni supplementari contiene questo codice:

```
public partial class Window1 : System.Windows.Window {

    public void InitializeComponent()
    {
        System.Uri resourceLocater = new System.Uri("window1.baml",
        System.UriKind.RelativeOrAbsolute);
        System.Windows.Application.LoadComponent(this, resourceLocater);
    }
}
```

La definizione della classe è uguale a quella del *code-behind* e definisce il metodo *InitializeComponent*, richiamato nel costruttore; è importante notare, come si poteva supporre, che non vi è nessun codice per la creazione e la definizione dei controlli specificati nel file XAML; viene richiamato *LoadComponent*, passando come uri il file *.baml*. Questo file è visibile sempre nella stessa directory ed è una versione binaria, compattata ed ottimizzata al caricamento, del file *.xaml* contenente le definizioni dei controlli. Il metodo statico *LoadComponent* carica questo file ed istanzia a runtime i controlli.

Così facendo si sposta parte del lavoro da compiletime a runtime. Il file *.baml* non è esterno all'applicazione, ma è incluso nelle risorse dell'assembly finale. Il file *App.xaml* contiene, nello stesso modo di Window1, il codice XAML,

ma con root element di nome *Application*. Tale classe rappresenta il motore dell'applicazione.

1.9 Progetto di tipo Browser Application (XBAP)

Quanto detto per i progetti di tipo Windows Application, vale anche per quelli di tipo *Browser Application* XBAP (Xaml Browser Application). La sostanziale differenza è che essi vengono eseguiti all'interno del browser Internet Explorer.

Le applicazioni XAML Browser vengono eseguite in una sandbox con permessi Internet specifici. Solo le caratteristiche che soddisfano determinate condizioni di sicurezza possono essere eseguite all'interno della *sandbox*. Questo permette di evitare problemi di sicurezza già riscontrati con componenti *ActiveX* che richiedono il controllo quasi totale del sistema, piuttosto che *applet Java* (anch'esse eseguite in sandbox).

1.10 Navigation Service

Un aspetto innovativo di WPF è la possibilità di svilluppare applicazioni non soltanto secondo il paradigma a finestre (windows), ma anche secondo il paradigma di navigazione tra pagine stile web. Ai progetti si possono infatti aggiungere elementi come Window o Page. Window (*System.Windows.Window*) rappresenta una classica finestra, mentre Page (*System.Windows.Controls.Page*) rappresenta una pagina.
Questa innovazione cerca ancora una volta di avvicinare i mondi WinForm e Web. La vera differenza tra Window e Page si nota attraverso l'utilizzo del *NavigationService*, ovvero quell'oggetto che, visualizzato in genere in alto nella finestra dell'applicativo, consente la navigazione avanti e indietro tra le pagine che compongono il progetto secondo quanto scritto nel codice.

1.11 Componenti di terze parti

Sebbene WPF permetta la creazione di applicazioni moderne dal design innovativo, non mette a disposizione una gamma completa di componenti e controlli. Le mancanze più sentite riguardano controlli quali tabelle e grafici,

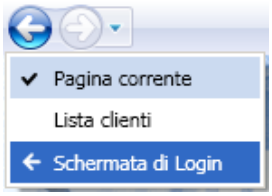


Figura 4.5: Esempio di NavigationService

per i quali la Microsoft non ha sviluppato ad oggi nessuna soluzione. Casualmente proprio questi componenti sono risultati fondamentali per lo sviluppo di *Business Console*.
Alcune software house propongono a pagamento suite di componenti di diversa natura e per diversi utilizzi. I più pubblicizzati soprattutto per ambienti di produzione sono *DataGrid*, *Chart* e *Calendari*, sebbene ancora in versioni *beta*. Non mancano controlli di natura più ludica più spesso distribuiti gratuitamente rispetto ai primi.

1.12 Porting

Porting su web di un'applicazione WPF esistente

Come discusso, WPF permette la creazione di applicazioni web visualizzabili tramite browser (*XBAP*).

Una volta terminata la progettazione di un applicativo di tipo Windows Application risulta abbastanza semplice il porting del progetto in XBAP, ma pertanto possibile solo la progettazione parallela dei 2 progetti. I progetti XBAP, operano in *sandbox*, e hanno accesso limitato alle risorse della macchina client.

Sebbene sia possibile e più semplice di quanto sia stato nelle precedenti versioni di Visual Studio e del Framework .NET, il porting su web di una soluzione software presenta alcuni aspetti da considerare:

- funzionalità che richiedono specificate azioni, come accesso a database, operazioni su filesystem o chiamate di sistema operativo non sono garantite funzionare in sandbox senza opportune modifiche al codice o alle impostazioni di sicurezza del browser (o IIS)

- la visualizzazione delle pagine nel browser può risultare leggermente diversa a livello grafico se non pensata e progettata appositamente
- XBAP è una tecnologia non matura e non ancora diffusa
- la visualizzazione di una applicazione XBAP è possibile soltanto con il browser Internet Explorer.

Per effettuare un porting si seguano i seguenti punti:

- creare un nuovo progetto XBAP Application in Visual Studio
- importare tutti i file del progetto
- aggiungere tutti i riferimenti necessari
- apportare opportune modifiche al caso

Porting su WPF di applicazioni Windows Application

La necessità di effettuare un porting su WPF da una Windows Application non è scontato. Non è pensabile e sostenibile il costo di un porting per il solo scopo di aggiornarsi alla nuova tecnologia: il porting impiegherebbe molte ore di riprogrammazione dell'interfaccia grafica sebbene il *business logic* rimanga invariato.

2 Windows Communication Foundation

2.1 Descrizione generale

Windows Communication Foundation, nome in codice Indigo, è una libreria di classi del Framework .NET (introdotta con la versione 3.0) per lo sviluppo di applicazioni distribuite. WPF rappresenta la nuova API Microsoft presente nel .NET Framework 3.0.

Allo stato dell'arte esistono diverse tecnologie per creare questo tipo di applicazioni, tutte supportate dal .NET Framework, quali:

- **WebService**: per esigenze di interoperabilità con il maggior numero di sistemi eterogenei
- **Remoting**: per mettere in comunicazione due sistemi .NET ed avere un forte controllo su entrambi
- **Microsoft Message Queue (MSMQ)**: per sviluppare applicazioni asincrone basate su messaggi

Poiché ogni tecnologia elencata ha un suo modello di programmazione e necessita quindi di una conoscenza specifica, da parte degli sviluppatori, Microsoft ha unificato il modello di programmazione di tutte le attuali tecnologie con WCF.

2.2 Caratteristiche

Un servizio WCF si basa sugli *Endpoints* che sono le porte attraverso le quali l'applicazione comunica con il mondo esterno; si può quindi affermare che un servizio WCF sia una collezione di *Endpoints*. A sua volta, un *Endpoint* è costituito da quelli che sono i pilastri di WCF: *Address*, *Binding*, *Contract*, ovvero il dove, come e cosa.

Address (Where)

È l'indirizzo al quale il servizio risponde, è composto da un URI, un Identity e una lista di Headers. L'informazione principale è l'URI che corrisponde all'indirizzo fisico del servizio (es. *http://localhost/ws/Service.svc*). Headers e Identity sono informazioni necessarie solo in casi particolari. Ad esempio

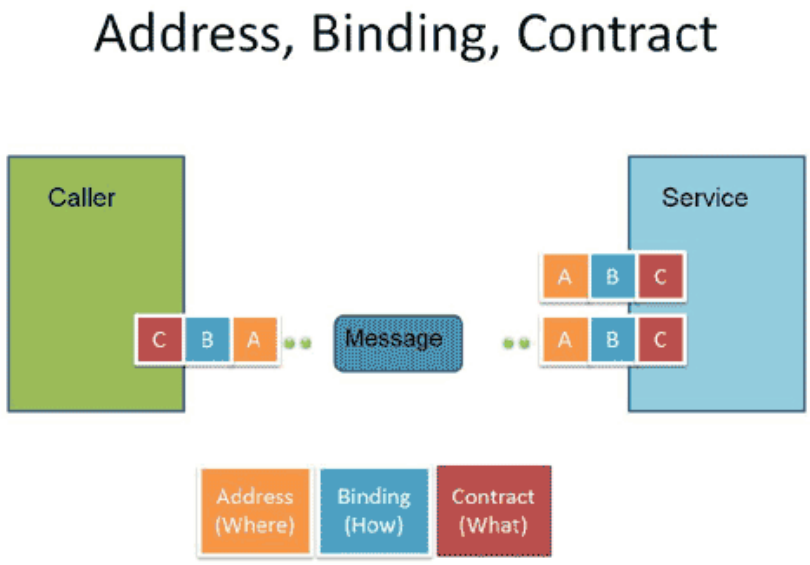


Figura 4.6: Schematizzazione grafica di un servizio WCF

quando ci sono più *Endpoints* può essere utile avere diversi Headers a seconda dell'*Endpoint* che il client utilizza.

Bindings (How)

I *Bindings* permettono di programmare in maniera svincolata dall'infrastruttura di trasporto, si occupano di quello che avviene tra il momento in cui il servizio spedisce logicamente il messaggio ed il momento in cui viene fisicamente messo in rete. In questo lasso di tempo vengono eseguiti numerosi passi che seguono una precisa pipeline di cui i *Bindings* sono responsabili.

Durante l'esecuzione della pipeline il messaggio deve attraversare due layer:

- **Behaviour**: trasformazioni da applicare al messaggio
- **Channels**: instradamento verso il canale fisico di trasporto

Il *Behaviour* si occupa della conversione dei dati dal formato codice al formato messaggio, come ad esempio vengono trasformate le informazioni da una classe al formato xml di un messaggio SOAP. I *Behaviours* si occupano anche della sicurezza, del criptaggio delle informazioni e di tutte le features relative ai dati. Durante la seconda fase il messaggio viene messo sul canale

di trasporto secondo quanto specificato in fase di configurazione. Quindi è in questa fase che si istanzia il canale (HTTP, HTTPS, MSMQ, TCP) su cui viaggeranno le informazioni. Poiché a livello di protocollo si possono controllare alcuni dettagli, in questo layer si possono aggiungere informazioni sulla modalità di trasmissione, protetta o meno, oppure sul Reliable Messaging.

La gestione dei *Bindings* viene gestita tramite la configurazione di un apposito file *.config* dalla soluzione di Visual Studio, tale file è denominato *app.config* per le soluzioni WinForm e *web.config* per le soluzioni Web.

Contracts (What)

I *Contracts* rappresentano l'interfaccia software di scambio che il nostro servizio pubblica.

I contratti non stabiliscono solo quali operazioni si possono invocare su un servizio, ma anche come e quali dati si debbano scambiare. Le tipologie di Contracts sono:

- ServiceContract
- DataContract
- MessageContract

ServiceContract

Definiscono il servizio e tutte le API che mette a disposizione. La definizione di un simile contratto vede la sua naturale definizione in un'interfaccia .NET da implementare. Di per sé, un'interfaccia non ha alcun senso parlando in termini di Service Oriented Architecture; infatti, pur definendo metodi e proprietà, non stabilisce se questi debbano essere resi pubblici o meno. Con SOA la visibilità di un metodo o di una proprietà all'interno di una classe non ha alcun collegamento con la visibilità che può avere all'interno del servizio. A seconda dei casi, un metodo privato di una classe potrebbe diventare un'API del servizio, mentre un metodo pubblico della stessa classe potrebbe non essere pubblicato dal servizio.

In WCF, per definire i metodi da pubblicare, bisogna decorarli con opportuni attributi dichiarati nella definizione dell'interfaccia. Gli attributi da utilizzare

sono *ServiceContract* con la quale marcare l'interfaccia e *OperationContract* per ogni metodo da pubblicare. Una volta che è stato stabilito quali operazioni mettere a disposizione dei client, occorre definire quali informazioni debbano essere pubblicate.

DataContract
Supponiamo che un servizio pubblichi un metodo che dato il codice fiscale ritorni i dati della persona associata. Nel *DomainModel* l'oggetto Persona contiene informazioni come Nome, Cognome, Data di Nascita e altro ancora, tuttavia può nascere l'esigenza di non pubblicare tutti i dati della persona ma soltanto quelli che si ritengono opportuni. Per ottenere un risultato del genere bisogna decorare sia la classe sia le sue proprietà con attributi; più approfonditamente alla classe va applicato l'attributo DataContract e alle proprietà l'attributo *DataMember*.

Per default, tutte le proprietà di una classe vengono mappate all'interno del corpo di un messaggio. A seconda delle esigenze, può capitare che si debbano piazzare delle informazioni nelle intestazioni invece che nel corpo; questo mapping tra le classi ed il formato del messaggio viene impostato tramite *MessageContract*.

MessageContract
Anche in questo caso l'impostazione di queste opzioni avviene tramite attributi e, più precisamente, *MessageContract* per la classe e *MessageBody* o *MessageHeader* per le proprietà.

Esempio di utilizzo WCF

Come esempio verrà creato un servizio che riceve in input un codice prodotto e restituisce tutte le informazioni legate al prodotto con quel codice.

La prima cosa da fare quando si costruisce un servizio con WCF è stabilire il contratto di comunicazione tramite interfaccia che nel nostro caso è la seguente:

```
[ServiceContract] public interface IContract {
    [OperationContract]
```

```
    Product GetProductById(string IdProduct);
}
```

Come detto in precedenza, è importante notare l'uso degli attributi. Una volta stabilito quali servizi mettere a disposizione, si determina quali dati l'entità Product debba pubblicare e quali invece debba mantenere privati.

Da notare in questo caso la possibilità di pubblicare la proprietà Price nonostante questa sia privata. Infatti, quando il client riceverà le informazioni i dati dell'entità Product saranno tre: IdProduct, Name e Price e non Category nonostante sia pubblico. Poiché gli attributi vengono utilizzati dal runtime di WCF per serializzare i dati, questi sono il naturale contenitore di alcune proprietà che permettono di controllare questo processo. Infatti è possibile definire come chiamare l'elemento che contiene la classe o come chiamare i singoli attributi che espongono i valori delle proprietà. Possono essere definite anche altre proprietà come l'ordine, l'obbligatorietà di un dato e altro ancora.

Nonostante questo buon livello di controllo a volte si ha bisogno di un controllo totale sul processo di serializzazione, in questo caso la cosa migliore è marcare l'interfaccia con l'attributo XmlSerializerFormat che invece di sfruttare il serializzatore di default utilizza il serializer standard del .NET Framework che viene utilizzato anche per i WebService Asmx.

L'interfaccia è ora pronta e le classi stabilite. Segue l'implementazione del codice per restituire i dati:

```
[DataContract] public class Product{
    [DataMember]
    public string IdProduct{
        get { return _idProduct; }
        set { _idProduct = value; }v }
    [DataMember]
    public string Name{
        get { return _name; }
        set { _name = value; }
    }
    [DataMember]
```

```
    private decimal Price{
        get { return _price; }
        set { _price = value; }
    }
    public string Category{
        get { return _category; }
        set { _category = value; }
    }
}
```

Per realizzare un servizio WCF questo è tutto il codice di cui abbiamo bisogno. Come si vede, non c'è alcun riferimento al tipo di tecnologia di trasmissione dati o codice di serializzazione o altro ancora.

Erogazione del servizio (hosting server)

Un servizio WCF non è una vera applicazione, quindi ha bisogno di appoggiarsi ad una struttura che ne permetta la pubblicazione; in pratica il servizio ha bisogno di un Host. Si possono utilizzare diversi tipi di Host: ad esempio possiamo scrivere una Console Application, un Servizio Windows o, più comunemente, utilizzare IIS. Quando si vuole utilizzare una propria applicazione per ospitare un servizio, occorre scrivere del codice per preparare l'infrastruttura necessaria, quindi istanziare le classi che si occupano della gestione dell'endpoint.

Come già accennato, la forza di WCF sta in parte nel codice semplificato ed unificato per ogni modalità di comunicazione, e in parte nel fatto che semplicemente cambiando la configurazione si può passare da http a tcp, da messaggi in chiaro a messaggi criptati, ecc...

Il posto naturale dove inserire le informazioni di configurazione è il file .config all'interno del quale va inserito un nodo *system.serviceModel* che è il contenitore di ogni impostazione:

```
<configuration>
    <system.serviceModel>
        <services>
            <service name="Service.Library.ProductRetriever">
```

```
                <endpoint
                    address="http://localhost/ws/Service.svc"
                    binding="basicHttpBinding"
                    contract="Service.Library.IContract" />
            </service>
        </services>
    </system.serviceModel>
</configuration>
```

Dal file di configurazione possiamo ricavare tutte le informazioni necessarie per il funzionamento del servizio e utilizzarle da codice per istanziare il servizio utilizzando l'oggetto ServiceHost in una semplice console application:

```
using (ServiceHost serviceHost = new
ServiceHost(typeof(Service.Library.ProductRetriever),
"http://localhost/ws/Service.svc")) {

serviceHost.Open(); }
```

Una volta avviata l'applicazione questa resterà in ascolto in base alle informazioni specificate nel .config relative all'address, binding e contract. Nel caso del nostro servizio, la creazione manuale di un Host è alquanto inutile in quanto possiamo sfruttare IIS. Con l'installazione di delle estensioni WPF e WCF, viene inserito un nuovo mapping nel metabase di IIS che associa l'estensione .svc alla pipeline di asp.net. Grazie a questa estensione, il runtime capisce che, quando arriva una richiesta con estensione SVC, questa deve essere passata al motore di WCF che istanzia il servizio per soddisfare la richiesta. La tecnica è molto simile ai file *.asmx* e consiste nel creare un file .svc contenente la dichiarazione del servizio da utilizzare:

```
<\%@ServiceHost language="c#" Debug="true"
Service="Servioce.Library.ProductRetriever" %>
```

Utilizzo del servizio (client)

Quando si parla di servizi, ci sono sempre due entità: una è il servizio che, una volta avviato, rimane in attesa di essere invocato; l'altra è il client che invia messaggi al servizio. Il client può essere una qualsiasi applicazione, sia

essa WinForm, Web, un secondo servizio WCF o ovviamente una qualsiasi altra piattaforma. Per poter invocare un servizio, il client deve conoscere l'interfaccia di comunicazione che in WCF è il contratto, quindi, analogamente a quanto avviene ora per i WebService, è necessario creare un proxy che si interfacci al servizio.

Attualmente, Visual Studio non ha un'interfaccia grafica per costruire in automatico il proxy, quindi la sola possibilità è utilizzare il tool *SvcUtil.exe*, da riga di comando. Questo tool risulta molto comodo perché non solo crea automaticamente il proxy, ma anche il file di configurazione che il client può utilizzare per la connessione. Quest'ultima frase lascia capire che anche il client può essere pilotato da configurazione senza bisogno di codice esattamente come si fa con il servizio.

Avendo anche il client un EndPoint, le informazioni da pubblicare sono le stesse del servizio: l'Address, il Binding ed il Contract. I primi due corrispondono ai dati dell'endpoint utilizzato dal servizio, mentre il contract corrisponde alla classe automaticamente generata nel proxy. L'altra differenza rispetto alla configurazione del servizio è il nome del nodo principale che contiene i dati: invece di avere un nodo ¡services¿ con diversi ¡service¿ innestati, esiste un nodo client con tutti gli endpoint figli:

```
<configuration>
    <system.serviceModel>
        <client>
            <endpoint
                address="http://localhost/ws/Service.svc"
                binding="basicHttpBinding"
                contract="Service.Library.IContract"/>
        </client>
    </system.serviceModel>
</configuration>
```

Anche qui, una volta stabilita l'interfaccia, si passa al codice per invocare il servizio e ricevere i dati. Nel nostro caso il client sarà una classica Windows Application che al click di un bottone invoca il servizio.

```
using (ContractProxy proxy = new ContractProxy()) {
```

```
Service.Library.Product p = proxy.GetProductById("1"); }
```

Questo snippet illustra come il codice sia in realtà molto banale e simile a quello che si fa ora per i WebService: si crea un'istanza del proxy e si invoca il metodo. Anche per il client, qualunque sia il protocollo, i behaviour e altro ancora, sarà sempre compito di WCF eseguire la pipeline che porta alle trasformazioni e all'invio fisico del messaggio e viceversa alla ricezione della risposta, ripercorrendo la pipeline al contrario.

3 Business Console

Come anticipato nel capitolo 2, il progetto *Business Console* in forma prototipale ha lo scopo di reperire da un database aziendale chiamato "**PVS05DAT**" le informazioni dei clienti e presentarle in maniera tabellare e il più *user-friendly* possibile. L'utente finale è il rappresentante ASI. *Business Console* deve inoltre permettere l'accesso ai dati tramite internet. Per questo secondo scopo, è stato sviluppato un Web Service apposito.
Tutto il codice è stato ampiamente commentato rendendo di fatto il progetto facilmente capibile in breve tempo e manutenibile dal personale ASI.

3.1 Requisiti di prototipo

In questa sezione verranno brevemente documentati tutti i requisiti richiesti per il progetto *Business Console*, che sarà valutato dall'azienda per sviluppi futuri. I requisiti che seguono esplicano formalmente quali sono i bisogni software e di sistema, che devono soddisfare il prototipo attraverso l'uso esclusivo di tecnologie Microsoft. Vengono classificati in *funzionali*, di *qualità* e di *interfacciamento*. Si sono inoltre attribuite ad ogni requisito, le seguenti priorità: *obbligatorio*, *desiderabile* e *opzionale*.

Requisiti Funzionali

- **Obbligatori**:
  - BC deve accedere al database aziendale PVS05DAT;
  - BC deve recuperare le informazioni dei clienti e presentarle in maniera tabellare;
  - BC deve mostrare il fatturato, la situazione finanziaria, l'estratto conto e le note commerciali dei clienti);
  - BC deve essere sviluppato secondo il paradigma WPF e WCF .NET 3.0;
  - BC deve interagire con un Web Service (WCF) per la sicurezza dei dati su database;
- **Desiderabili**:
  - portabile su web con facilità senza riscrittura di codice;
- **Opzionali**:
  - permettere all'utente di selezionare diversi stili e grafici per la rappresentazione dei dati;

Requisiti di Qualità

- **Obbligatori**:
  - utilizzare esclusivamente tecnologie Microsoft;
- **Desiderabili**:
  - fare in modo che l'interfaccia grafica di BC sia immediata e semplice da utilizzare, adottando una grafica essenziale ed amichevole per l'utente;

Requisiti di Interfacciamento

- **Obbligatori**:
  - BC deve essere il più semplice possibile da installare
- **Desiderabili**:
  - il codice di BC deve essere il più commentato e manutenibile possibile

A stage terminato tutti i tipi di requisiti (*Funzionali, di Qualità, di Interfacciamento*) sono stati soddisfatti.

3.2 Standard adottati

Per quanto riguarda gli standard al fine di rendere chiare le modifiche effettuate e produrre un codice più facilmente leggibile si è deciso di adottare i seguenti standard di documentazione del codice:

- breve commento che spiega il comportamento di ogni nuovo metodo inserito prima del codice del metodo;
- informazioni nell'intestazione dei file
  `//Autore:`
  `//Data:`
  `//Descrizione modifica:`
- tabulazione per ottenere l'indentazione di ogni blocco funzionale;
- linee di codice non più lunghe di 80 caratteri;
- linee vuote di separazione tra diversi blocchi funzionali;
- una spaziatura dopo la virgola e prima e dopo gli operatori +, -, *, /, =, >, < ;
- la parentesi graffa di apertura del blocco funzionale segue l'intestazione del blocco ed è da essa separata da uno spazio;

- la parentesi graffa di chiusura del blocco viene posta nella riga successiva all'ultima istruzione del blocco, e allineata verticalmente con la sua intestazione; utilizzo di nomi significativi delle variabili;

L'adozione di questi standard non è stata imposta dall'azienda, ma è stata una mia scelta seguendo le nozioni apprese nei vari corsi di programmazione e soprattutto nel corso di Ingegneria del Software.

3.3 Componenti di terze parti

Per lo sviluppo non si è optato per l'utilizzo di componenti appartenenti alle vecchie API come il *DataGridView*, in quanto sarebbe stato fuorviante per il caso di studio WPF.
Sebbene WPF permetta la creazione di applicazioni moderne dal design innovativo, non mette a disposizione una gamma completa di componenti e controlli. Le mancanze più sentite riguardano controlli quali tabelle e grafici, per i quali la Microsoft non ha sviluppato ad oggi nessuna soluzione. Si è scelto di effettuare una valutazione comparativa tra vari prodotti di terze parti.
Per *Business Console* sono stati testati i seguenti componenti *DataGrid* a pagamento in versione dimostrativa:

- **DataGrid for WPF** (Xceed) Licenza free, ma funzionalità aggiuntive con licenza Vanguard per 1 anno [$ 499]
- **NetAdvantage Volume 1** (Infragistics) Tutti i controlli della suite, sottoscrizione al servizio, updates, e nuove release per 1 anno [$ 395]
- **GridFX** (SoftwareFX) Licenza Test/Development (più economica) [$ 799]

Come esito di una valutazione comparativa il componente DataGrid più adatto è risultato *NetAdvantage* di Infragistics.
Per quanto concerne il componente Chart (grafico) non sono stati trovati fino al momento della stesura del documento componenti WPF. Sono state reperite in rete solo versioni Beta che ovviamente quindi non sono state testate (come Chart FX for WPF di SoftwareFX). Sono stati testati i seguenti componenti Chart .NET a pagamento in versione dimostrativa:

- **Chart for .NET 4.1** (Xceed)

- **Chart FX** (SoftwareFX)
- **Dundas Chart** (Dundas)
- **Manco.Chart** (Manco Software)
- **.NET Charting**

Si è optato per il componente *Chart FX* di SoftwareFX.

3.4 WCF Service

Business Console si appoggia ad un WCF Service per recuperare i dati dal database "**PVS05DAT**" SQL Server. Il WCF Service è denominato *BC_Service*. L'interfaccia pubblica di *BC_Service* è molto semplice ed è composta da 2 metodi quali *CreateConnection* e *ExecQuery*. Il primo crea la connessione e la imposta per il servizio utilizzato dal client, il secondo esegue una query prima aprendo una connessione e poi chiudendola. Entrambi i metodi sono commentati per una miglior manutenibilità.
Poiché il prodotto per l'utente finale dovrà appoggiarsi su questo web service tramite http, è necessario attivare il *web service* su un computer aziendale per renderlo disponibile anche all'esterno della rete LAN, configurando con i parametri necessari al caso il file di configurazione.

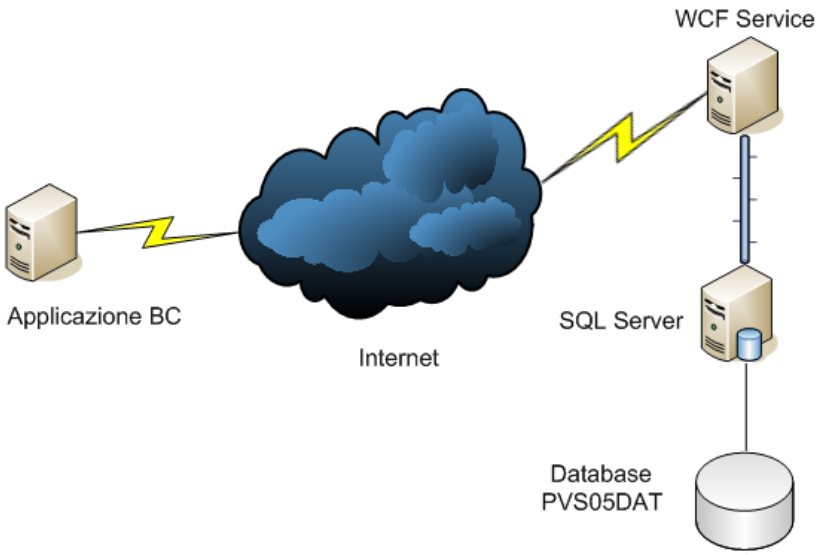


Figura 4.7: Schematizzazione dell'utilizzo del Web Service per BC

3.5 Codifica

Poiché *Business Console* viene sviluppata con WPF, molto codice riguarda l'interfaccia grafica scritta in XAML. Ricordo infatti il paradigma alla base di WPF, ovvero "**Application = Code + Mark Up**". Quasi tutto il codice XAML è stato scritto a mano senza l'ausilio di *Cider*; ciò è stato fatto per capire meglio il funzionamento ed eventuali anomalie, e soprattutto per poter apprendere meglio l'argomento e stilare il documento aziendale relativo a WPF in maniera più dettagliata.

Per quanto riguarda il *Business logic* invece, sono risultate necessarie le classi presenti in figura 4.8, tale figura è stata autogenerata dall'IDE Visual Studio.

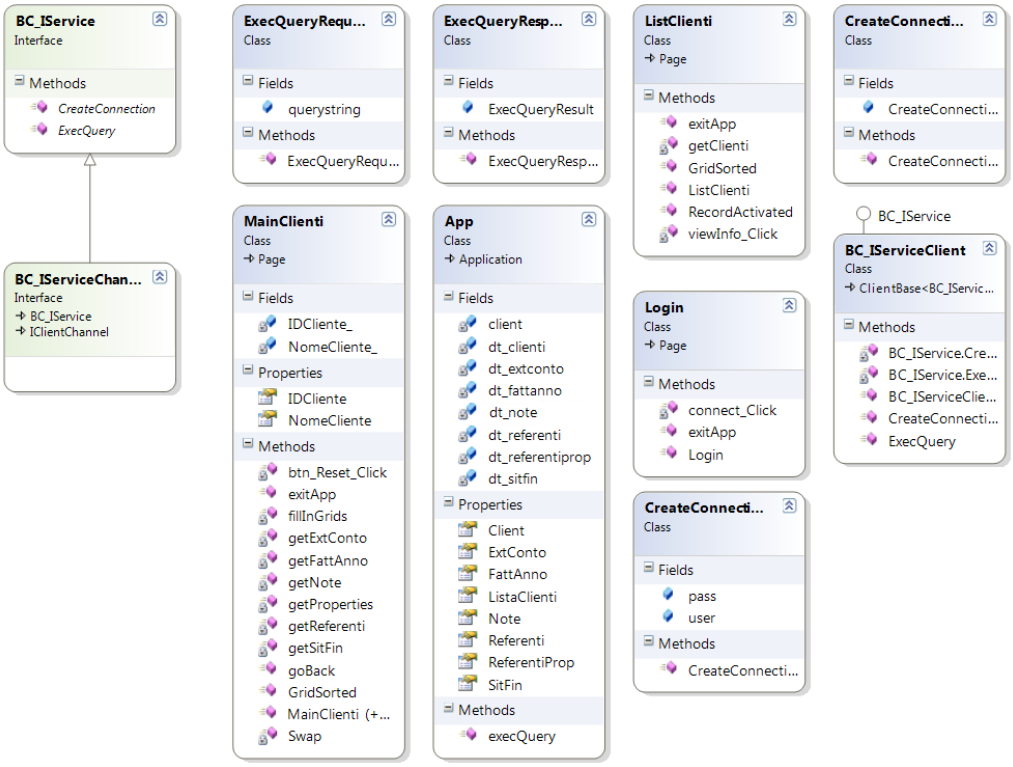


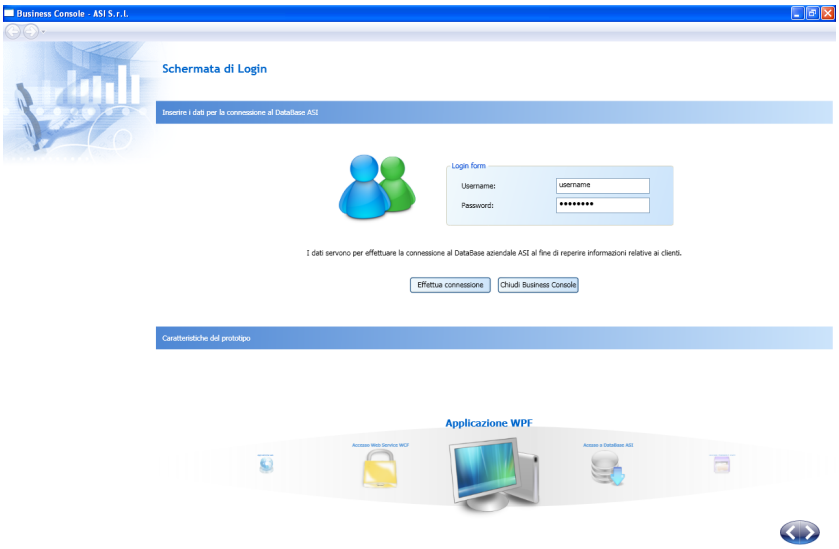
Figura 4.8: Diagramma classi Business Console

Eseguire il porting su XBAP è stato immediato, non vi è stata nessuna riscrittura di codice come preventivato dallo studio di WPF. Il browser *Internet Explorer* non ha riscontrato problemi, i programmi XBAP, girano in *sand-box* ed hanno accesso ad un numero limitato di risorse, ciò non è comunque stato

un problema, e anche l'interrogazione del database tramite *web service* non ha presentato anomalie.

4 Manuale Business Console

La prima pagina del programma mostra una schermata di *login* come in figura.

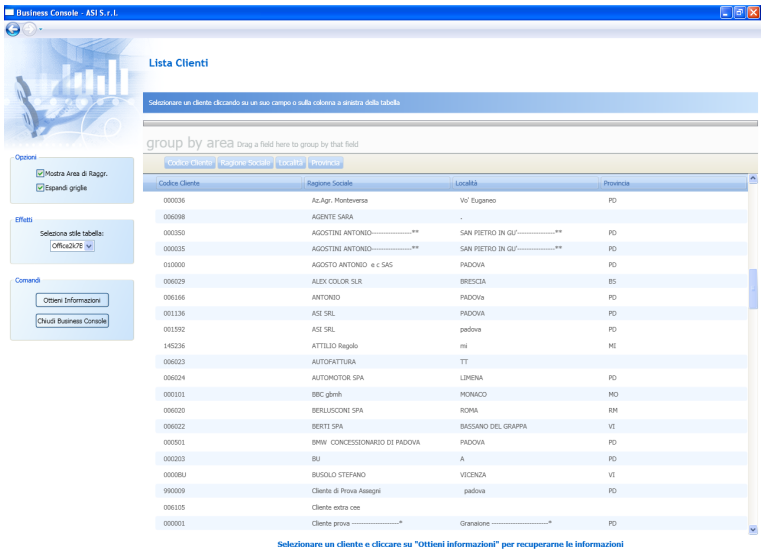


La parte alta della finestra visualizza una login form per l'immissione dei dati per la connessione al database al fine di reperirne le informazioni. Una volta inseriti i dati personali per la connessione cliccare sul pulsante "Effettua connessione" per effettuare una connessione e recuperare la lista dei clienti. In caso i dati di login risultino essere sbagliati verrà visualizzata la scritta "Errore di connessione. Username o password errati." In caso il web service al quale si appoggia Business Console non sia attivo verrà visualizzato un Message Box che segnala l'errore di servizio non disponibile.

La parte bassa della finestra illustra le caratteristiche del prototipo e ha puro scopo presentativo. Con i pulsanti presenti in basso a destra è possibile ruotare le icone presenti sopra di essi. Tale controllo è stato aggiunto solamente come prova di un componente presente nella suite NetAdvantage di Infragistics, ne è pertanto probabile la rimozione in una successiva versione di Business Console.

Una volta effettuata la connessione correttamente dalla pagina di login, verrà

presentata la pagina Lista Clienti.

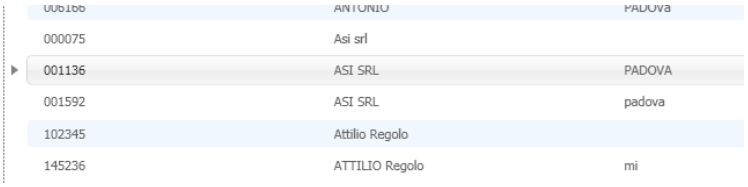


La pagina presenta un menù a sinistra con 3 group box quali: Opzioni, Effetti, Comandi.

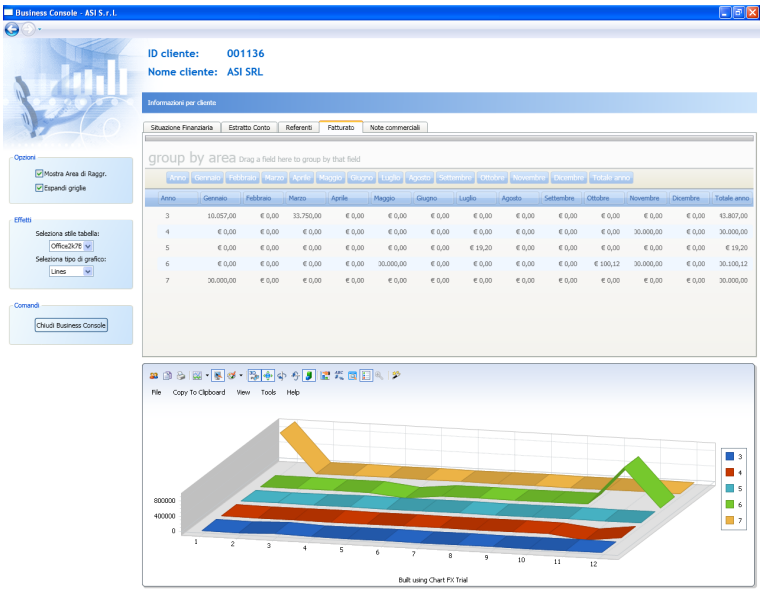
Nel group box Opzioni è possibile scegliere se mostrare le aree di raggruppamento e espandere le griglie presenti nella parte destra della pagina tramite una check box. In Effetti è possibile selezionare lo stile grafici preferito della tabella e in Comandi sono presenti 2 bottoni: il primo (Ottieni Informazioni) permette di recuperare le informazioni del cliente selezionato nella griglia presente a destra della pagina, il secondo (Chiudi Business Console) termina l'applicazione.

La parte destra della pagina presenta il titolo della pagina e l'azione da fare ovvero: Selezionare un cliente cliccando su un suo campo o sulla colonna a sinistra della tabella. Facendo ciò, verrà evidenziato il cliente desiderato come in figura.

È possibile raggruppare i clienti per Provincia, Località, Codice cliente e Ragione Sociale. Nell'esempio successivo si mostra la tabella raggruppata per Provincia. Per fare ciò trascinare il tasto Provincia presente sopra la griglia nella zona con scritto Group by Area. Una volta selezionato il cliente desider-



ato, cliccare sul pulsante Ottieni Informazioni per reperirne le informazioni che verranno visualizzate nella nuova pagina Informazioni cliente.
La pagina Informazioni cliente si presenta come segue.



La parte sinistra è simile a quella della pagina Lista clienti. È presente una combo box in più dove poter selezionare il tipo di grafico che si desidera visualizzare e un pulsante Torna alla lista clienti per tornare alla pagina precedente. Il grafico utilizzato è il componente *Chart FX*.

Nella parte destra sono presenti più schede denominate Situazione Finanziaria, Estratto Conto, Referenti, Fatturato e Note commerciali. In Situazione Finanziaria è sono presenti i dati del cliente visualizzati come segue.
Nelle altre schede sono presenti griglie che hanno come contenuto le informazioni riportate in titolo alla scheda. Nella griglia Referenti in particolare, è possibile selezionare un referente tra quelli disponibili nella parte alta della scheda (nello stesso modo come si è fatto in precedenza per selezionare un cliente) per recuperare le informazioni aggiuntive del referente come Tele-

fono, Qualifiche, E-mail, Cellulare, Note.

Per muoversi tra le pagine visitate si utilizzi il Navigator. Il Navigator è presenta in alto a sinistra di tutte le pagine descritte e si presenta come in figura.

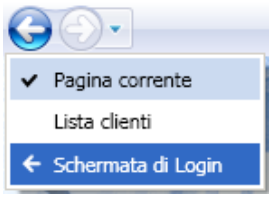


Figura 4.9: Navigation Service

Selezionando Lista Clienti si ritorna alla pagina Lista Clienti, selezionando Schermata di Login si ritorna alla prima pagina del programma per effettuare una nuova connessione.

5 Manualistica

Come da pianificazione (sez. 3) sono stati redatti dei documenti destinati all'azienda che rappresentano una parte importante dello stage. Dopo varie revisioni le forme definitive dei documenti risultano essere:

- Analisi WPF (43 pagine)
- Analisi WCF (25 pagine)
- Valutazione Comparativa componenti aggiuntivi (14 pagine)
- Manuale programmatore BC (4 pagine + codice commentato)
- Manuale utente BC (13 pagine)
- Consuntivo

I documenti di Analisi WPF e WCF sono stati i più importanti e sostanziosi. Uno scopo di questo stage infatti era quello di rendere disponibile al personale ASI della documentazione sulle nuove tecnologie che fosse chiara, immediata per l'apprendimento e lo sviluppo di software in azienda e il meno dispersiva possibile.

6 DB Compare

Come verrà illustrato nel consuntivo finale, sono rimaste a disposizione delle ore extra, durante le quali ASI mi ha chiesto di sviluppare un brevissimo applicativo, denominato *DB Compare* per il confronto tra 2 database. Lo scopo era quello di confrontare 2 database e rilevarne le differenze di struttura e popolamento. L'esigenza era nata dal fatto che ASI in alcuni ambiti utilizza due database, uno dei quali viene aggiornato con le modifiche fatte al primo. Di fatto i 2 database dovrebbero essere identici a differenza che uno viene usato come copia per prove e aggiornamenti, mentre l'altro è quello con dati affidabili aggiornato dal primo meno frequentemente.

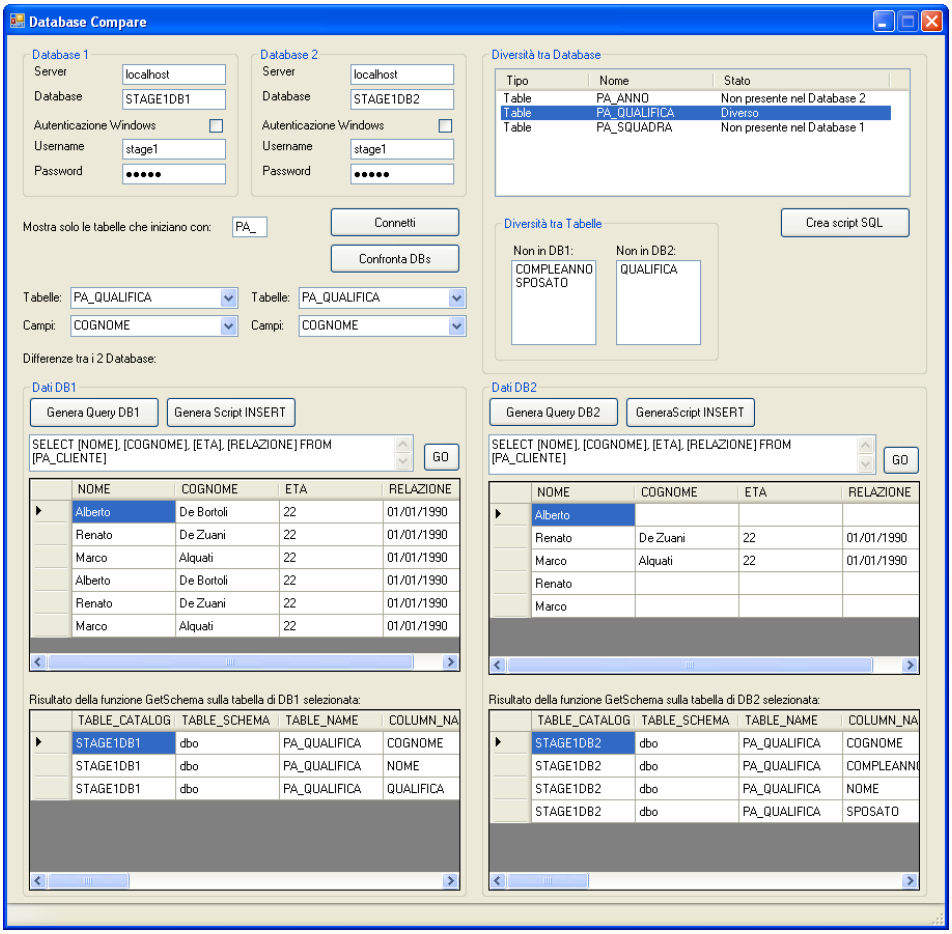


Figura 4.10: Interfaccia principale DB Compare

Senza andare nel dettaglio, poiché ciò non è oggetto principale della tesi, si può notare dalla figura 4.10 che *DB Compare* permette di reperire la strut-

tura di 2 database sapendo l'indirizzo di essi, il nome e in caso di dati di accesso. Una volta connessi ai database si può effettuare il confronto tra i due e reperire le diversità in una listbox. È possibile generare delle query per aggiornare un database con le strutture mancanti ma presenti nell'altro e viceversa. Stesso avviene con i dati tramite la generazione di uno *script*.

7 Tecniche di verifica

Per l'attività di verifica effettuata sul prodotto si sono seguiti due tipi di analisi: *analisi statica* e *analisi dinamica*. L'analisi statica è stata effettuata utilizzando le seguenti tecniche:

- **Analisi di flusso di controllo**: ha l'obiettivo di accertare che il codice sia ben strutturato ed esegua nella sequenza attesa;
- **Analisi di flusso dei dati**: controlla la correttezza degli accessi alle variabili rilevando possibili anomalie quali sovrascrittura di dati sensibili non ancora utilizzati. Si occupa anche della verifica sull'utilizzo di variabili non ancora assegnate che siano globali o locali dell'unità;
- **Walkthrough**: lettura critica del codice a largo spettro.

Si è utilizzato più frequentemente il metodo *Walkthrough* cercando quindi di controllare tutto il codice prodotto approfittandone anche per commentare delle parti di codice in modo da rendere più chiara la struttura dei possibili cammini percorribili dal flusso di controllo.
L'analisi dinamica consiste invece di prove, test di esecuzione del codice prodotto, e può essere effettuata su singole componenti, su gruppi di componenti tra loro in relazione o sull'intero sistema. Si parla quindi rispettivamente di *test di unità*, *test di integrazione* e *test di sistema*.
I test di unità possono avere due obiettivi:

- accertare la corretta relazione tra input forniti all'unità e output emessi al termine dell'esecuzione; in questo caso si parla di test funzionali, o black box, i quali non tengono conto della struttura interna del codice bensì solo delle uscite prodotto in relazione agli ingressi immessi;
- accertare la correttezza della logica interna dell'unità, cercando di attivare tutti i possibili cammini del flusso di controllo.

Una volta accertato il funzionamento delle singole componenti, i test di integrazione non hanno dato particolari problemi così come i test di sistema che sono un caso particolare di integrazione visto che coinvolgono tutte le unità. Ho testato tutte le funzionalità del prodotto usando anche i cosiddetti casi limite, con lo scopo di evitare errori imprevisti in fase di esecuzione.

8 Verifica & Validazione

La *fase di verifica*, che risponde alla domanda "Did I build the product right?", si occupa di come sia stato svolto un processo oppure solo una parte di esso. Grazie a questa fase è stato possibile accertare che gli errori fatti durante il percorso di stage fossero controllati ed eliminati ove possibile. Durante lo studio e la realizzazione del prototipo è stato molto importante riuscire a verificare il corretto svolgimento delle fasi. La verifica è stata attuata tramite le revisioni formali stabilite ad inizio stage e grazie ad incontri settimanali con il tutor aziendale.

La *fase di validazione* risponde alla domanda "Did I build the right product?", ovvero si occupa di cosa sia stato prodotto da un processo, in questa fase è stato importante capire se l'interfaccia corrispondesse alle attese. Per quanto possibile si è cercato di adottare il ciclo **PDCA** in modo tale da poter pianificare le attività da svolgere, eseguirle, controllarle ed agire di conseguenza in caso di errori.

9 Consuntivo

In questa sezione verranno mostrate le ore effettive impiegate rispetto alla pianificazione definita ad inizio stage.

Fasi	Dettagli	Ore previste	Ore effettive
Analisi dei requisiti primo progetto di stage	(terminato per motivi da attribuire all'azienda)	0	40
Formazione	studio Framework .NET 2.0 studio IDE Visual Studio 2005 studio Framework .NET 3.0 (WPF e WCF)	100	70
Analisi componenti aggiuntivi e Progettazione	analisi componenti aggiuntivi di terze parti progettazione interfaccia BC progettazione web service BC	60	64
Sviluppo	sviluppo interfaccia BC sviluppo web service BC	60	32
Documentazione	stesura documenti .NET 3.0 stesura manuale BC	60	58
Verifica & Validazione	verifica e validazione test	20	10
Attività extra	sviluppo DB Compare presentazione finale	0	30
TOTALE		**300**	**304**

Figura 4.11: Consuntivo finale

È da notare come la fase di sviluppo abbia subito un calo di ore. Ciò è dovuto ad una sovrastima iniziale, mentre in realtà questa fase si è rivelata più breve del previsto. Le ore impiegate nel progetto di stage iniziale sono state recuperate perlopiù nella fase di *Formazione*, che con un risparmio di 30 ore ha colmato in buona parte le ore spese (40) nel progetto iniziale. La fase di *Attività Extra* non era stata preventivata, ma poiché le fasi di *Formazione* e *Sviluppo* sono risultate più brevi, si è potuto impiegare delle ore in questa fase.

Qui di seguito è possibile notare il raffronto grafico delle percentuali delle ore impiegate per ogni fase. Il calo di ore totali effettive, come già detto, è stato portato, grazie anche ad un impegno maggiore del previsto, ad un totale di ore lavorative giornaliere talvolta pari a 10 e continuando le attività, per quanto possibile, anche nei weekend (ore non contate in figura 4.11). Così facendo sono riuscito a portare a termine tutti gli obiettivi prefissati nei tempi previsti a inizio primo progetto di stage, recuperando quindi le ore "perse". È stata fatta inoltre una presentazione in azienda sullo stage effettuato con la presenza dell'amministratore delegato, del tutor e del responsabile dello sviluppo. La cosa non ha avuto comunque un impatto rilevante sul consuntivo

in quanto la durata di essa è stata di un paio di ore, ma è stata estremamente utile per capire come si svolge una presentazione aziendale. Inoltre, dopo un breve periodo, a stage concluso, ASI ha chiesto agli stagisti se fossero stati disponibili disinteressatamente a ripetere la presentazione per altro personale aziendale direttamente interessato. Personalmente ho accettato la richiesta.

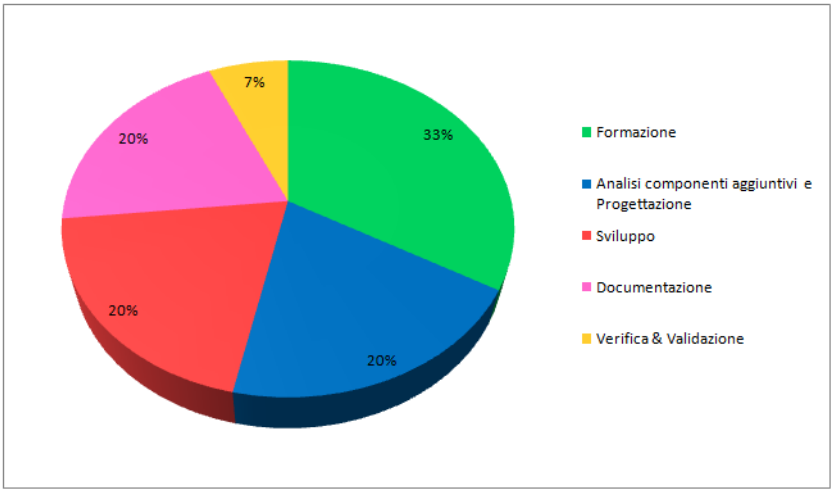


Figura 4.12: Percentuale ore previste per fase

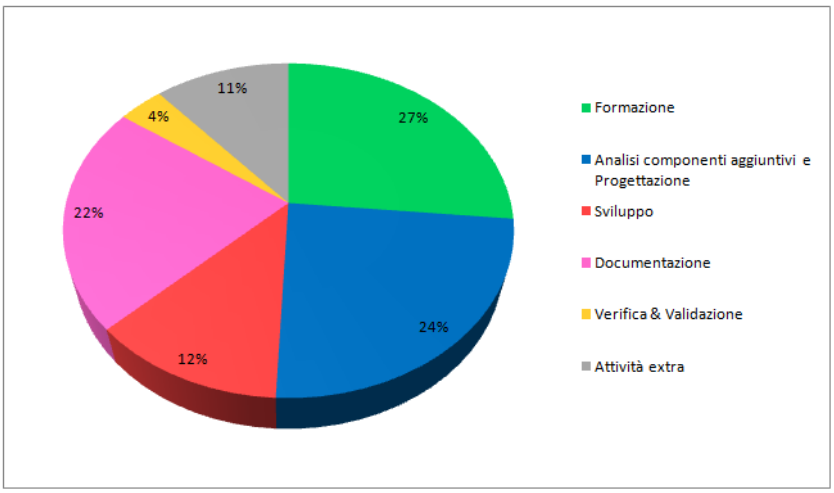


Figura 4.13: Percentuale ore impiegate per fase

Capitolo 5

Considerazioni finali

Questo capitolo contiene una valutazione finale dell'esperienza di stage. Verranno mostrate le critiche, le aspettative, i risultati ottenuti ed una breve valutazione del corso di studi.

1 Analisi Critica

A stage terminato è stato naturale un giudizio globale sul lavoro svolto. Non vi sono state difficoltà insormontabili, solo problemi da affrontare. Le difficoltà maggiori comunque hanno riguardato la prima settimana di stage, ovvero le ore che concernono il primo progetto di stage. Esso fin dall'inizio è risultato ostico nel capire effettivamente cosa l'azienda volesse, in quanto non aveva fornito punti di partenza solidi. Lo studio di fattibilità dunque è stato travagliato e oggetto di numerose modifiche in poco tempo. Col secondo progetto di stage invece non vi sono stati un vero e proprio Studio di fattibilità e una Analisi dei requisiti in quanto erano sottintesi da parte dell'azienda. Tutti i requisiti per il prototipo *Business Console* elencati nella sezione 3.1 sono stati soddisfatti.
Il consuntivo finale è stato molto soddisfacente e si è discostato dalla pianificazione iniziale in positivo, ovvero sono state risparmiate ore che erano già state impiegate nel progetto di stage abortito.

1.1 Vantaggi d'utilizzo di WPF

Si prevede che nei prossimi anni lo sviluppo di applicazioni in ambiente Microsoft verterà sempre più sulla tecnologia descritta in questo documento; è pertanto importante, se non d'obbligo, considerare seriamente il passaggio all'utilizzo di queste in ambito aziendale per lo sviluppo di soluzioni software competitive. Seguono i principali vantaggi derivati dall'utilizzo di WPF.

- Utilizzo di un linguaggio di mark up che semplifica la creazione di GUI e ne aumenta le possibilità grafiche;
- Presenza di nuove API più moderne per dotare le applicazioni di look accattivanti;
- Avvicinamento tra applicazioni stand alone e web-based grazie al linguaggio di marcatura;
- Benefici derivati dal property system e dal binding engine che consentono di limitare la lunghezza del codice;

Il passaggio all'utilizzo di WPF abbandonando le vecchie API prevede conoscenze riguardanti i linguaggi di marcatura (XML) e programmazione OO, concetti, nella maggior parte dei casi, già noti ai programmatori.

1.2 Vantaggi d'utilizzo di WCF

Si prevede che nei prossimi anni lo sviluppo di applicazioni in ambiente Microsoft verterà sempre più sulle tecnologie descritte in questo documento; è pertanto importante considerare seriamente il passaggio all'utilizzo di queste in ambito aziendale per lo sviluppo di soluzioni software competitive. Seguono i tre principali vantaggi derivati dall'utilizzo di WCF.

- Unificazione di svariati servizi (Remoting, Web Services, Miscrosoft Message Queue) in un unico modello di programmazione;
- Semplicità d'uso e di configurazione del servizio anche in vista di cambiamenti dello stesso;
- Più facile manutenibilità del servizio anche in vista di cambiamenti dello stesso;

Il passaggio alla creazione di sistemi distribuiti sviluppati secondo il paradigma WCF non risulta complesso, anzi, da quanto elencato nel paragrafo precedente si evince che porterà a un indubbio risparmio per la progettazione in termini di tempo/persona. Non è da considerare seriamente invece il problema di porting di un'applicazione esistente su tecnologia WCF. Come detto, WCF unifica il modello di programmazione dei vari sistemi di comunicazione e non ne introduce uno nuovo. Le applicazioni esistenti potranno continuare ad operare come progettate. Portare un'applicazione su WCF è ragionevole solo se l'azione è mossa da motivi considerevoli.

2 Confronto tra Java e .NET

A stage terminato è stato naturale chiedersi se la tecnologia .NET avesse qualcosa di simile alla struttura del framework Java di Sun. Ho riassunto qui una breve panoramica delle differenze e delle somiglianze tra le due tecnologie.

2.1 Architettura Java

"*Write Once, Run Anywhere*" è il motto che la Sun utilizza per identificare Java. Già da questa semplice frase si capisce qual è l'obbiettivo di questo linguaggio, rendere il codice scritto indipendente dalla piattaforma. In Java

è possibile scrivere un programma in ambiente Linux ed eseguirlo in ambiente Windows, senza apportare alcuna modifica al codice sorgente. Per ottenere questa indipendenza viene utilizzata una architettura particolare che è illustrata nell'immagine 5.1.

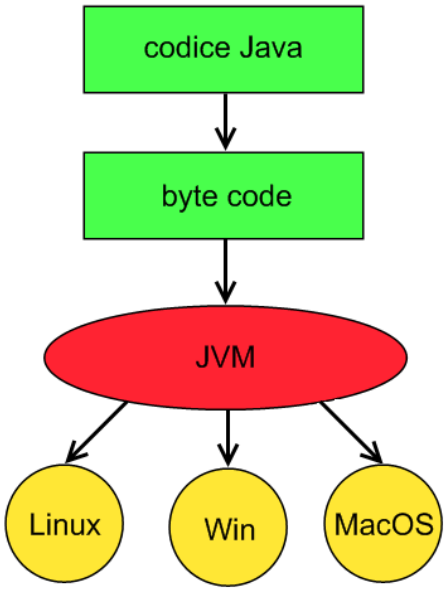


Figura 5.1: Architettura Java

L'architettura funziona nel modo seguente: l'utente scrive il proprio codice seguendo le regole grammaticali e sintattiche del linguaggio (Java). Il file prodotto, identificato dall'estensione .java, viene "*compilato*", in questa maniera viene creato un codice intermedio indipendente dalla macchina denominato byte code che è contenuto all'interno dei file .class. A questo punto entra in gioco la "*virtual machine*", il componente fondamentale della struttura; ne esiste una per ogni sistema operativo ed è essa che si occupa di rendere effettivamente portabile il codice. Quando un programma viene eseguito, prima viene caricata la "*virtual machine*" (se non è già stato fatto) la quale si occupa di chiamare ed eseguire le classi (byte code) di cui si ha bisogno ed utilizzando una compilazione *JIT* si preoccupa di tradurre in linguaggio macchina le istruzioni del programma. I compilatori *JIT* hanno la particolarità di effettuare la compilazione "al volo" ovvero durante l'esecuzione del programma stesso. Per eseguire il medesimo programma su di un sistema operativo diverso basta assicurarsi che sulla macchina di destinazione sia presente una "*virtual machine*" appropriata, copiare sulla macchina il byte code

del programma ed eseguirlo. In questa semplice maniera viene mantenuta la portabilità del codice.

2.2 Architettura .NET

Il prodotto Microsoft parte da un presupposto diverso, invece di puntare sulla portabilità del codice (anche se non è esclusa questa possibilità) punta sulla possibilità di lasciare la libertà al programmatore di usare il linguaggio di programmazione da lui preferito. Questa scelta secondo Microsoft permette di creare programmi molto velocemente perché per utilizzare questa piattaforma non è necessario imparare alcun nuovo linguaggio. Altri, invece, vedono questa libertà come una nota negativa che a lungo porta a scrivere codice più confuso, più difficile da leggere e da manutenere. Dal punto di vista architetturale il sistema può essere schematizzato come in figura 5.2.

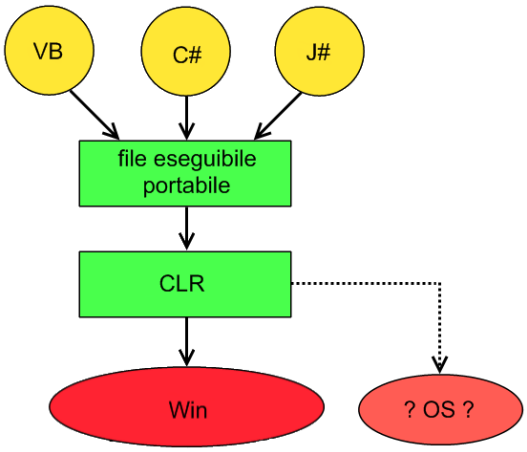


Figura 5.2: Architettura .NET

Una volta prescelto il linguaggio di programmazione preferito (n.b. C# è il linguaggio nativo della piattaforma) si scrive il proprio codice: il file così generato deve essere compilato con un compilatore per il linguaggio prescelto che sia compatibile con le specifiche .Net. Questa compilazione genera un codice intermedio molto simile al byte code di Java che in questo caso viene chiamato "*portable executable file*". La creazione di questo codice intermedio permette di nascondere il tipo di linguaggio utilizzato. Nel caso della programmazione ad oggetti si può anche passare come argomento di un metodo oggetti scritti in altri linguaggi. Questa interoperabilità tra linguaggi è pos-

sibile proprio perché tutti i compilatori generano codice coerente tra loro. Quando viene richiamato dal nostro programma un metodo esterno, non si ha la necessità di sapere in che linguaggio è stato scritto poiché sappiamo che sicuramente è stato compilato seguendo delle regole comuni, di conseguenza tutto il codice .Net è utilizzabile da tutti i linguaggi .Net. Una volta compilato il proprio programma, per poterlo eseguire, bisogna far entrare in gioco il "*common language runtime*" che concettualmente è molto simile alla "*virtual machine*". Questo componente deve essere eseguito prima di poter lanciare i programmi .NET; una volta in memoria esso si preoccupa di richiamare le classi di cui si ha bisogno e utilizzando un compilatore *JIT* compila in codice nativo le classi di cui ha bisogno. Teoricamente, implementando dei "*common language runtime*" per diversi sistemi operativi, si raggiungerebbe la stessa portabilità di Java; questa possibilità nella figura precedente è rappresentata dalla freccia tratteggiata.

2.3 Confronto

Dal punto di vista architetturale i due framework sono molto simili: Java ha enfatizzato l'indipendenza dalla macchina mentre .Net ha enfatizzato l'indipendenza dal linguaggio. Con piccole aggiunte queste due caratteristiche sono ottenibili da entrambi i concorrenti; infatti, anche nel caso di Java, esistono dei progetti funzionanti che permettono l'indipendenza dal linguaggio, ad esempio JPython, PERCobol, the TCL/Java project e Eifell-to-Java VM. Questi lavori mettono a disposizione dei compilatori adatti, che permettono di scrivere codice nei diversi linguaggi e alla fine della compilazione si ottiene come risultato del "*byte code*" che, come abbiamo visto, è utilizzabile dalla "*Java Virtual Machine*"; anche in queste implementazioni, come accade in .Net, si ha l'interoperabilità tra i diversi linguaggi. Dal punto di vista di .Net, è possibile ottenere l'indipendenza dalla piattaforma. Questa caratteristica è possibile implementando un "*common language runtime*" su ogni sistema su cui si vuole operare. In sostanza le caratteristiche generali offerte dai due sistemi sono abbastanza simili sebbene non identiche; la scelta sull'utilizzo di uno piuttosto che dell'altro dipende dalla familiarità che si ha con uno dei due framework e dalle caratteristiche proprie del progetto che si sta affrontando.

3 Risultati attesi

Ad inizio stage, i risultati attesi erano abbastanza chiari. Le più importanti aspettative riguardavano lo studio e la comprensione di nuove tecnologie mai viste durante il corso di studi quali .NET e i *Web Service*. Inoltre, lo stage era destinato a far parte di una realtà aziendale e ad affrontare autonomamente problemi mai visti prima mettendo in pratica quanto appreso nel corso "Ingegneria del Software".

4 Risultati ottenuti

Il corso "Ingegneria del Software" è stato utilissimo ad una crescita personale dal lato lavorativo e organizzativo, e ciò si è potuto mettere a frutto in azienda, la quale ha avuto un sicuro beneficio poiché, a costo praticamente zero, ha ottenuto il prototipo "Business Console" e "DB Compare", oltre alla numerosa documentazione riguardante la tecnologia .NET.
Facendo riferimento a quanto detto precedentemente e rispetto ai punti della sezione 1.4 si può dire che gran parte delle aspettative sono state raggiunte e questo stage ha avuto esito sicuramente positivo. Mi è stato permesso di fare la prima esperienza nel mondo del lavoro e di rapportarmi all'interno di un ambiente nuovo come quello aziendale. Infine, l'azienda ha avanzato proposte di lavoro nei miei confronti, cosa che, sicuramente gradita, ha confermato e apprezzato il lavoro svolto.

5 Ringraziamenti

Un ringraziamento speciale va alla mia famiglia, a nonna Bianca e zia Carmen per il supporto datomi in questi anni di studio universitario. Dopodiché ringrazio Jessica per il supporto morale, il mitico gruppo del corso di Ingegneria del Software "Wheelsoft", tutti i colleghi studenti ed amici dell'Università. Un ringraziamento anche a tutti i docenti del corso di studi e in particolare al Prof. Massimo Marchiori per avermi supervisionato l'attività di stage e questa tesi, al tutor aziendale Marino Della Puppa e al responsabile del settore sviluppo di ASI Carlo Venturella per la disponibilità avuta nei miei confronti durante il periodo di stage.

Appendice A

Glossario

A

API: acronimo di Application Programming Interface, insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per un determinato compito. È un metodo per ottenere un'astrazione, di solito tra l'hardware e il programmatore, o tra software a basso ed alto livello;

Assembly: unità elementari per la creazione delle applicazioni .NET. Sono gli elementi fondamentali di distribuzione, controllo delle versioni, riutilizzo, ambito di attivazione e autorizzazioni di protezione. Un assembly è una raccolta di tipi e risorse progettati per interagire tra loro e formare così un'unica unità funzionale. Con gli assembly si forniscono a Common Language Runtime le informazioni necessarie sulle implementazioni dei tipi. Per Common Language Runtime non esiste alcun tipo all'esterno del contesto di un assembly;

Avalon: nome in codice di Windows Presentation Foundation;

B

beta: versione di prova di un software non definitivo, già testato dagli esperti, che viene messo a disposizione anche dei meno esperti, confidando

proprio nelle loro azioni imprevedibili che potrebbero portare alla luce nuovi bug o incompatibilità del software stesso;

Browser: programma software che permette agli utenti utilizzatori la navigazione in Internet, visualizzando e interagendo con testi, immagini e altre informazioni. Questo navigatore interpreta il codice delle pagine HTML e lo visualizza in forma di ipertesto;

C

Ciclo di vita: principio teorico che indica il metodo da seguire nel progettare e nello scrivere un programma;

Cider: componente di Visual Studio che si occupa della creazione di interfacce grafiche in maniera immediata tramite operazioni di *drag&drop* e trascinamento;

code-behind: files di codice che contengono la parte di *business logic*;

D

DataBase: insieme di dati riguardanti uno o più argomenti correlati tra loro ed è strutturata in modo da permettere l'uso di questi dati da parte di applicazioni software;

debugging: attività che consiste nella individuazione della porzione di software affetta da errore (bug) rilevati nei software a seguito dell'utilizzo del programma;

Diagramma di Gantt: grafico illustrante il parallelismo e le dipendenze delle attività di un progetto nel tempo;

F

Framework .NET: ambiente per la creazione, la distribuzione e l'esecuzione di tutti gli applicativi che supportano .NET siano essi Servizi Web o altre applicazioni;

G

GUI: acronimo di *graphical user interface*, è un paradigma di sviluppo che mira a consentire all'utente di interagire col computer manipolando graficamente degli oggetti, svincolandolo dall'obbligo di imparare una sintassi come invece avviene con le interfacce testuali command line interface;

H

HTML: acronimo di HyperText Mark-up Language, linguaggio di marcatura per la creazione di pagine web statiche;

I

ICT: acronimo di *Information and Communications Technology*, cioè Tecnologie dell'Informazione e della comunicazione (TIC). Con questa sigla si intende la convergenza di informatica e telematica per nuovi modi di trasmettere l'informazione;

IDE: ambiente integrato di sviluppo. È un software che aiuta i programmatori nello sviluppo del software. In questo documento viene spesso usato come sinonimo di Visual Studio;

Incrementale (mod. di ciclo di vita): modello di ciclo di vita iterativo in cui analisi e progettazione si svolgono una sola volta e l'iterazione c'è tra progettazione di dettaglio e realizzazione;

Indigo: nome in codice di Windows Communication Foundation;

J

Java: linguaggio di programmazione orientato agli oggetti, derivato dal C++ (e quindi indirettamente dal C) e creato da James Gosling e altri ingegneri di Sun Microsystems;

JIT: tipo di compilazione che permette la traduzione del bytecode in codice macchina nativo in fase di run-time;

P

PDCA: modello studiato per il miglioramento continuo della qualità in un'ottica a lungo raggio. Serve per promuovere una cultura della qualità che è tesa al miglioramento continuo dei processi e all'utilizzo ottimale delle risorse;

Q

Query: interrogazione verso un database al fine di ottenere dei dati contenuti in esso;

S

sandbox: termine che sta ad indicare un ambiente controllato con privilegi limitati. Inizialmente era riferito all'ambiente controllato messo a disposizione dalla JVM (Java Virtual Machine) nel quale vengono eseguite le Java applet;

SDK: termine che in italiano si può tradurre come "pacchetto di sviluppo per applicazioni", e sta a indicare un insieme di strumenti per lo sviluppo e documentazione di software;

SOAP: protocollo leggero per lo scambio di messaggi tra componenti software, tipicamente nella forma di componentistica software. La parola object manifesta che l'uso del protocollo dovrebbe effettuarsi secondo il paradigma della programmazione orientata agli oggetti;

U

URI: stringa che identifica univocamente una risorsa generica che può essere un indirizzo Web, un documento, un'immagine, un file, un servizio, un indirizzo di posta elettronica, ecc;

V

Visual Studio: ambiente di sviluppo integrato sviluppato da Microsoft e atto ad aumentare la produttività aiutando il programmatore;

W

WCF: sottosistema applicativo proprietario di Microsoft, che offre la struttura API per la creazione di applicazioni distribuite in ambienti Windows;

Web Service: secondo la definizione data dal W3C un Web Service (servizio web) è un sistema software progettato per supportare l'interoperabilità tra diversi elaboratori su di una medesima rete;

WPF: libreria di classi del Framework .NET proprietarie Microsoft (introdotta con la versione 3.0) per lo sviluppo dell'interfaccia grafica delle applicazioni in ambienti Windows;

X

XAML: linguaggio di markup basato su XML, utilizzato per descrivere l'interfaccia grafica delle applicazioni basate sulla libreria Windows Presentation Foundation;

XBAP: acronimo di XAML Browser Applications, indica una applicazione per browser sviluppate con WPF;

XML: eXtensible Markup Language, linguaggio estensibile di marcatura strutturalmente simile ad HTML;

Bibliografia

[1] J. David, B. Ryan, R. DeSerranno, A. Young (2005), *Professional WinFX Beta* - Wilwy Publishing

[2] C. Anderson (2007), *Essential Windows Presentation Foundation* - Addison-Wesley

[3] M. MacDonald (2007), *Pro WPF* - APress

[4] Team Microsoft (2007), *Microsoft Virtual Labs: Building Windows Presentation Foundation Applications - C#* - Microsoft

[5] C. Andrade, S. Livermore, M. Meyers, S. Van Vliet (2007), *Professional WPF Programming* - Wilwy Publishing

[6] C. Sells, I. Griffiths (2005), *Programming Windows Presentation Foundation* - O'Really

[7] A. Nathan (2007), *Windows Presentation Foundation Unleashed* - Sams Publishing

[8] C. McMurtry, M. Mercury, N. Watling, M. Winkler (2007), *Windows Communication Foundation Unleashed* - Sams Publishing

[9] Team Microsoft (2007), *Microsoft Virtual Labs: The Fundamentals of Programming the Windows Communication Foundation* - Microsoft

[10] B. Johnson, C. Skibo, M. Young (2006), *Microsoft Visual Studio 2005 Guida all'uso* - Mondadori Informatica

[11] X. Liu (2004), *Accessing ODBC data sources using a web service* - Articolo tratto da http://www.codeproject.com

[12] G. Ferron, *WCF Windows Communication Foundation - Serializzazione* - Articolo tratto da http://www.dotnethell.it

[13] Microsoft, documentazione Microsoft sul Framework .NET http://msdn2.microsoft.com/en-us/netframework/default.aspx

[14] Team WinFXItalia.com, Articoli, script e tutorial tratti da http://www.winfxitalia.com/