Alma Mater Studiorum
Università di Bologna

seconda facoltà di ingegneria

Corso di Laurea Triennale in Ingegneria Informatica

Progettazione e Sviluppo di Applicazioni Web per Cloud Computing

Elaborata nel corso di
Sistemi Operativi

Relatore:
Ricci Alessandro

Presentata da:
Quarta Riccardo

Sessione I
Anno Accademico 2011/2012

PAROLE CHIAVE

Cloud Computing

Software-as-a-Service

SaaS Multi-Tenant

Windows Azure

Alla mia Je, che mi ha sempre supportato...
Alla mia famiglia, che mi dato una grande opportunità...

Introduzione

Negli ultimi anni si sente sempre più spesso parlare di cloud computing. L'idea di fondo di questo concetto è quella di pagare per il solo effettivo utilizzo di un servizio, disponibile sulla rete, avendo a disposizione la possibilità di poter variare le proprie risorse utilizzabili a seconda delle necessità, che potrebbero essere, per esempio, applicazioni standard oppure spazi di storage per i dati. Quando cominciò a diffondersi l'utilizzo del Web, la rete Internet veniva raffigurata come una nuvola (cloud) in modo tale che si rendesse l'idea di un'entità esterna rispetto alla nostra casa o al nostro posto di lavoro, un qualcosa cioè al di fuori dei luoghi abituali in cui vengono utilizzati i PC. Tale rappresentazione diventa ora utile per poter spiegare il concetto di cloud computing. Infatti, grazie a questa nuova tecnologia, dati e programmi normalmente presenti nei nostri computer potranno ora trovarsi sul cloud.

Molti reparti IT sono costretti a dedicare una parte significativa del loro tempo a progetti di implementazione, manutenzione e upgrade che spesso non danno un vero valore per l'azienda. I team di sviluppo hanno cominciato quindi a rivolgersi a questa nuova tecnologia emergente per poter minimizzare il tempo dedicato ad attività a basso valore aggiunto per potersi concentrare su quelle attività strategiche che possono fare la differenza per un'azienda. Infatti un'infrastruttura come quella cloud computing promette risparmi nei costi amministrativi che raggiungono addirittura il 50% rispetto ad un software standard di tipo client/server.

Questa nuova tecnologia sta dando inizio ad un cambiamento epocale nel mondo dello sviluppo delle applicazioni. Il passaggio che si sta effet-

tuando verso le nuove soluzioni cloud computing consente infatti di creare applicazioni solide in tempi decisamente più brevi e con costi assai inferiori, evitando inoltre tutte le seccature associate a server, soluzioni software singole, aggiornamenti, senza contare il personale necessario a gestire tutto questo.

L'obiettivo di questa tesi è quello di mostrare una panoramica della progettazione e dello sviluppo di applicazioni Web nel cloud computing, analizzandone pregi e difetti in relazione alle soluzioni software attuali.

Nel primo capitolo viene mostrato un quadro generale in riferimento al cloud, mettendo in luce le sue caratteristiche fondamentali, esaminando la sua architettura e valutando vantaggi e svantaggi di tale piattaforma.

Nel secondo capitolo viene presentata la nuova metodologia di progettazione nel cloud, operando prima di tutto un confronto con lo sviluppo dei software standard e analizzando poi l'impatto che il cloud computing opera sulla progettazione.

Nel terzo capitolo si entra nel merito della progettazione e sviluppo di applicazioni SaaS, specificandone le caratteristiche comuni ed elencando le piattaforme di rilievo allo stato dell'arte. Si entrerà inoltre nel merito della piattaforma Windows Azure.

Nel quarto capitolo viene analizzato nel particolare lo sviluppo di applicazioni SaaS Multi-Tenant, specificando livelli e caratteristiche, fino a spiegare le architetture *metadata-driven*.

Nel quinto capitolo viene operato un confronto tra due possibili approcci di sviluppo di un software cloud, analizzando nello specifico le loro differenze a livello di requisiti non funzionali.

Nel sesto capitolo, infine, viene effettuata una panoramica dei costi di progettazione di un'applicazione cloud.

Indice

Capitolo 1

Introduzione al Cloud Computing

Il termine *cloud* nasce nel mondo delle telecomunicazioni quando i provider di servizi cominciarono ad utilizzare le *Virtual Private Network* (VPN)[1] per le comunicazioni di dati.

Con Cloud Computing si intende computazione, software, accesso e memorizzazione di dati che non richiedono la conoscenza della locazione fisica e della configurazione del sistema che fornisce il servizio in questione da parte dell'utente finale. Il cloud computing è un tendenza recente nell'IT che sposta computazione e dati dai pc desktop o portatili che utilizziamo ogni giorno all'interno di grandi data center.

La definizione ufficiale di cloud computing è fornita dal *National Institute of Standards and Technology* (NIST) e afferma che "il cloud computing è un modello che favorisce un accesso di rete facilitato e on-demand ad un gruppo condiviso di risorse computazionali configurabili (reti, server, applicazioni di memorizzazione dati e servizi). Queste risorse possono essere fornite molto velocemente con un minimo sforzo nella gestione o nell'interazione con il provider di servizi". Grazie alla diffusione su larga scala di Internet, le

[1] Una VPN è una rete di telecomunicazioni privata, instaurata tra soggetti che utilizzano un sistema di trasmissione pubblico, come per esempio Internet.

applicazioni possono ora essere utilizzate come servizi sulla rete, riducendo così il costo generale drasticamente.

L'obiettivo principale del cloud computing è quello utilizzare al meglio le risorse distribuite, combinandole in modo tale da raggiungere un rendimento più elevato e da essere abili nel risolvere problemi di computazione su larga scala.

1.1 Un po' di storia

Il concetto che sta alla base del cloud computing fu introdotto negli anni 60 da John McCarthy. Egli riteneva che in futuro la computazione sarebbe stata alla portata di tutti. Le caratteristiche del cloud computing vennero esplorate per la prima volta nel 1966 da Douglas Parkhill nel suo libro, *The Challenge of the Computer Utility*.

Come detto precedentemente, il termine cloud proviene dal mondo delle telecomunicazioni, nel quale le aziende cominciarono ad offrire servizi di VPN che possedevano una qualità del servizio paragonabile alle soluzioni alternative del tempo ma con un costo assai inferiore. Inizialmente venivano forniti dei circuiti *point-to-point* di dati nei quali si poteva tuttavia rilevare un enorme spreco di banda. Con l'avvento però dei servizi VPN si poteva trasferire il traffico di rete in modo da bilanciare l'utilizzo generale della rete stessa. Al giorno d'oggi il cloud computing estende questo concetto in modo tale da inglobare server e infrastrutture di rete.

Molti esponenti dell'industria si sono cimentati nel cloud computing e hanno implementato alcune soluzioni. Un ruolo chiave è stato ricoperto da Amazon, che nel 2006 ha lanciato *Amazon Web Service* (AWS), ma anche Google e IBM hanno cominciato importanti progetti di ricerca in questo ambito.

1.2 Caratteristiche del Cloud

- Il cloud computing si basa principalmente sulla nozione di servizio. I servizi sono componenti computazionali autonomi, indipendenti dalla piattaforma, che possono essere pubblicati e composti ottenendo reti di applicazioni distribuite. L'applicazione così ottenuta diventa a sua volta un servizio disponibile sulla rete. A questo punto i fornitori non venderanno più software da installare su sistemi proprietari del cliente, ma metteranno a disposizione online determinate risorse, che possono essere o vere e proprie applicazioni o addirittura risorse fisiche come CPU o dischi fissi per storage. Non a caso l'essenza del cloud computing è *X as a Service*, che può essere tradotto con "Tutto è Servizio". Infatti tale acronimo si riferisce al fatto che un numero sempre più alto di servizi viene rilasciato su Internet piuttosto che sui PC in locale.

- Oltre a questo, molto importante è il concetto di virtualizzazione. Con tale termine si intende la possibilità di astrarre le componenti fisiche, come l'hardware, degli elaboratori al fine di renderle disponibili al software in forma di risorsa virtuale. Uno dei principali vantaggi è la razionalizzazione e l'ottimizzazione delle risorse hardware grazie ai meccanismi di distribuzione delle risorse disponibili di una piattaforma fisica. Si può quindi ottenere che più macchine virtuali possono girare contemporaneamente su un sistema fisico condividendo le risorse della piattaforma. Le eventuali contese di risorse vengono gestite dai software di virtualizzazione che si occupano della gestione dell'ambiente. Grazie a queste tecniche è quindi possibile allocare dinamicamente risorse sulla base delle richieste dei clienti, aumentando o diminuendo il loro quantitativo a seconda di carichi computazionali maggiori o minori.

- Nel cloud computing gli utenti accedono a dati, applicazioni o qualsiasi altro servizio con l'aiuto di un browser a prescindere dal dispositivo utilizzato e dal luogo in cui si trova l'utente. L'infrastruttura, che è

generalmente fornita da terze parti, è accessibile con l'aiuto di Internet. Per questo motivo il costo è molto ridotto rispetto alle soluzioni tradizionali. Infatti non è necessario acquistare nuove risorse per far fronte a carichi computazionali occasionali.

- Per l'implementazione sono necessarie molte meno abilità nell'IT rispetto ad altre soluzioni.
- Un servizio affidabile può essere ottenuto attraverso l'utilizzo di più siti che possono rendersi indispensabili affinchè tutto sia sempre funzionante e pronto per l'eventuale recupero a causa di problemi insorti, ad esempio il down dei server. Tuttavia capita che alcuni provider in questo ambito soffrano di improvvisa interruzione dei loro servizi di cloud computing e in questi momenti difficilmente i loro utenti possano fare qualcosa.
- La condivisione di risorse e costi attraverso un vasto insieme di utenti permette un efficiente utilizzo dell'infrastruttura.
- La manutenzione è più facile poiché le applicazioni di cloud computing non necessitano di installazione sui pc degli utenti.
- La modalità *pay-per-use* permette di determinare l'utilizzo di un'applicazione a client.
- Le performance possono essere monitorate.
- Il livello di sicurezza può essere addirittura migliore rispetto ai sistemi tradizionali perché i provider sono capaci di destinare risorse per risolvere problemi di sicurezza che molti clienti non potrebbero affrontare. Tuttavia questo aspetto lascia ancora molta preoccupazione quando i dati sono abbastanza confidenziali, portando quindi ad un ritardo nell'adozione del cloud computing su larga scala.

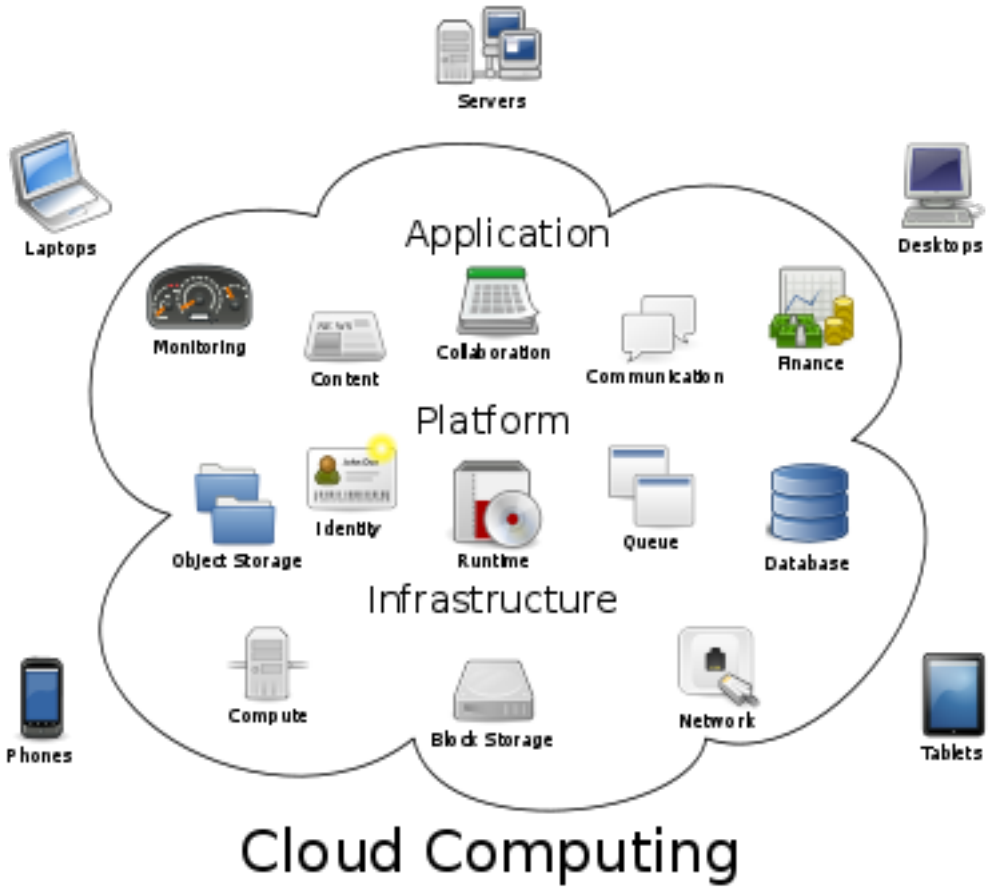

Figura 1.1: Il Cloud Computing

1.3 Architettura del Cloud

Il sistema del cloud computing può essere suddiviso in due parti: il *frontend* e il *backend*. Entrambi sono connessi tra di loro attraverso la rete, solitamente Internet. Il **frontend** è ciò che l'utente vede e, per essere visualizzato, è necessario che il computer dell'utente e l'applicazione possano accedere al cloud. Il **backend** è invece il cloud vero e proprio ed è quindi composto da tutti quei dispositivi come computer, server e dispositivi di memorizzazione dei dati.

Il monitoraggio del traffico di rete, l'amministrazione del sistema e le richieste degli utenti sono gestite da un server centrale. Questo segue determinate regole (i cosiddetti protocolli) e sfrutta un software speciale chiamato middleware che dà la possibilità ai computer nella rete di comunicare tra di loro.

1.3.1 Livelli e servizi

Il cloud computing è articolato secondo tre livelli fondamentali, che determinano come il concetto di servizio venga di fatto erogato. Tali livelli sono *Software as a Service* (SaaS), *Platform as a Service* (PaaS), *Infrastructure as a Service* (IaaS), dove quest'ultimo è il più basilare ed ogni livello più alto astrae dai dettagli di quelli inferiori.

Esistono inoltre altri livelli secondari come *Data as a Service* (DaaS), attraverso il quale vengono messi a disposizione via web solamente i dati ai quali gli utenti possono accedere tramite qualsiasi applicazione come se fossero residenti su un disco locale, e *Hardware as a Service* (HaaS), attraverso il quale l'utente invia dati ad un computer e questi vengono elaborati da altri computer messi a disposizione e restituiti infine all'utente iniziale.

SaaS	**PaaS**	**IaaS**
Software as a Service	**Platform as a Service**	**Infrastructure as a Service**
Gov-Apps	Application Development	Servers
Communication (email)	Security Services	Network
Collaboration	Database Management	Storage
Productivity tools		Management
ERP		Reporting
	Examples	
SalesForce.com	GAE	GoGrid
NetSuite	Microsoft's Azure	Flexiscale
Oracle	Amazon EC2	Joyent
IBM		
Google Apps		

Tabella 1.1: Riassunto delle piattaforme SaaS, PaaS, IaaS

Il livello *cloud application* implementa il concetto di *Software as a Service* (SaaS) su Internet, in modo tale da rimuovere la necessità di installare e far girare l'applicazione sui sistemi degli utenti. In questo ambito caratteristiche importanti sono l'accesso (basato sulla rete) e il controllo del software disponibile in commercio, entrambi monitorati da postazioni centralizzate.

È importante non dimenticare la possibilità data ai clienti di accedere a queste applicazioni da remoto attraverso Internet. In questa categoria esempi di provider per eccellenza sono Oracle, IBM e Microsoft; inoltre Google Apps è uno dei SaaS più usati in tutto il mondo.

Il livello *cloud platform*, il cosiddetto *Platform as a Service* (PaaS), fornisce una piattaforma di computazione usando l'infrastruttura cloud. Questa contiene tutta l'applicazione generalmente necessaria al client che ne usufruisce. In questo modo l'utente non ha bisogno di affrontare l'eventuale seccatura di comprare e installare software o hardware richiesti dall'applicazione.

Attraverso questo servizio gli sviluppatori possono tenere sott'occhio tutti i sistemi e gli ambienti richiesti per la life cycle del software: sviluppo, testing, deploying e hosting delle applicazioni web. Famosi esempi sono Google App Engine e Microsoft Azure.

Il livello *cloud infrastructure*, anche chiamato *Infrastructure as a Service* (IaaS), fornisce, come suggerisce il nome stesso, l'infrastruttura richiesta come un servizio. L'utente non deve quindi acquistare eventuali server richiesti, data center o risorse di rete.

Inoltre il vero vantaggio in questo caso è che i clienti devono pagare solamente per il tempo effettivo di utilizzo del servizio. In questo modo essi possono ottenere il servizio molto più velocemente e con costi minori. Esempi in questo ambito sono GoGrid e Flexiscale.

1.3.2 Deployment di un servizio

Per effettuare il deployment di una soluzione cloud computing, il primo passo fondamentale è quello di decidere su quale tipo di cloud implementarla. Ad oggi vi sono tre possibilità: cloud pubblico, cloud privato e cloud ibrido.

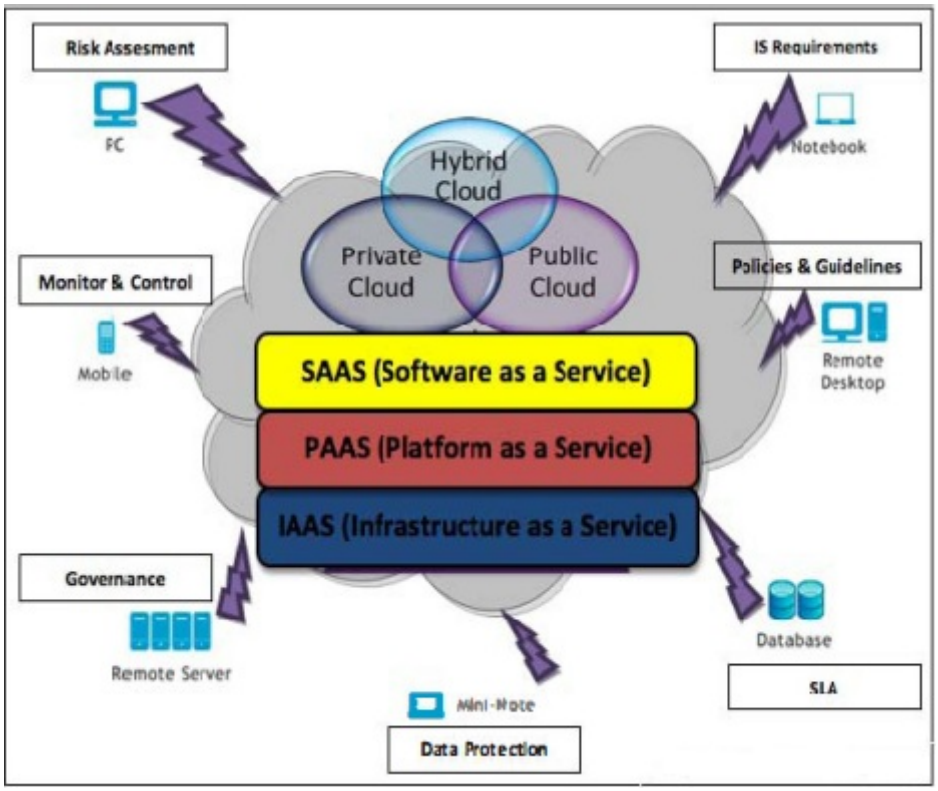

Figura 1.2: Mappa del Cloud Computing

Cloud Pubblico

Il **cloud pubblico** permette l'accesso degli utenti al cloud attraverso delle interfacce disponibili direttamente con l'utilizzo di un browser web. Coloro che usufruiscono del servizio devono quindi pagare solamente per il tempo di utilizzo vero e proprio, come accade nel *pay-per-use*. Per fare un

esempio pratico, possiamo confrontare questo procedimento con l'elettricità nelle nostre case: ogni utente paga solamente per quanta energia consuma.

Tuttavia il cloud pubblico è meno sicuro se confrontato con altri modelli poiché tutte le applicazioni e i dati su questa tipologia di cloud sono più soggetti ad attacchi informatici. Per poter porre rimedio a tutto ciò si potrebbe fare in modo tale che i controlli di sicurezza siano implementati attraverso un processo di validazione esteso ad entrambe le parti, sia lato provider del servizio cloud sia lato utente.

Cloud Privato

Nei **cloud privati** le operazioni avvengono all'interno dei data center propri dell'organizzazione. Il vantaggio principale in questo caso è che è molto più semplice controllare la sicurezza, la manutenzione e gli upgrade e inoltre fornisce un maggior controllo su deployment ed utilizzo. Il cloud privato può essere di fatto comparato ad una intranet.

Se confrontato con il cloud pubblico, dove tutte le risorse e le applicazioni erano controllate dal provider di servizi, nel cloud privato questi servizi sono riuniti insieme e resi disponibili agli utenti a livello dell'organizzazione. Risorse e applicazioni sono quindi controllate direttamente dalla stessa: la sicurezza qui è potenziata poiché solamente gli utenti dell'azienda medesima hanno accesso al cloud.

Cloud Ibrido

È una combinazione di cloud pubblico e privato. In questo modello un cloud privato è collegato a uno o più servizi cloud esterni. È un modo molto più sicuro per controllare dati e applicazioni e permettere agli utenti di accedere alle informazioni sulla rete. Tutto ciò à l'opportunità all'organizzazione di offrire i suoi servizi nel cloud privato e se occasionalmente ci fosse bisogno di maggiori risorse computazionali si fa domanda al cloud pubblico.

Community Cloud

Esiste infine una quarta modalità: quando molte organizzazioni decidono congiuntamente di costruire e condividere un'infrastruttura cloud, i loro requisiti e le loro politiche allora si parla di modello *community cloud*. In questo caso l'infrastruttura del cloud può risiedere presso un provider di terze parti o all'interno di una delle organizzazioni nella community.

1.4 Vantaggi e svantaggi

Andiamo ora a fare una panoramica di confronto sul cloud, in modo da poter analizzare pregi e difetti.

1.4.1 I Vantaggi

Facilità di controllo

La manutenzione dell'infrastruttura, che sia essa hardware o software, è semplificata in modo tale da creare meno problemi ai team IT. Inoltre le applicazioni che necessitano di molto spazio fisico sono più semplici da usare in un ambiente cloud se confrontate con una medesima soluzione a livello locale. Anche dalla parte dell'utente tutto risulta più immediato: l'unica cosa di cui si ha bisogno è semplicemente un browser web con connettività Internet.

Riduzione dei costi

Il vantaggio maggiore per le piccole e medie imprese risiede proprio nel fatto che il cloud computing riduce drasticamente i costi IT per queste aziende. Non vi è più la necessità di comprare costosi sistemi per utilizzi occasionali di maggiori risorse computazionali. Inoltre il quantitativo di risorse umane è molto basso: basti infatti pensare che applicazioni come le email possono essere semplicemente configurate e moltissime altre sono gratuite come accade per pacchetto *Google Apps*.

Servizi sempre attivi

Le interruzioni dei servizi sono meno frequenti nelle piattaforme di cloud computing, in modo tale da assicurare all'utente la massima fruibilità.

Controllo sulle interruzioni improvvise

In caso di problemi gravi, un backup è ovviamente sempre utile. Mantenere il backup dei dati più importanti grazie ai servizi di cloud computing è una necessità per molte organizzazioni. Per questo i servizi di storage cloud non mantengono solo i dati al di fuori del sito, ma si assicurano di avere sistemi in loco capaci di riparare eventuali errori.

Green computing

A causa dell'eccessivo uso dei sistemi, emissioni nocive, spreco di elettricità ed enorme consumo di energia sono tra i più grandi svantaggi dei moderni sistemi di computazione. Tutto ciò può essere notevolmente ridotto attraverso i servizi di cloud computing.

1.4.2 Gli Svantaggi

I più grandi problemi che riguardano il cloud computing sono la sicurezza e la privacy.

Cedere i propri dati confidenziali ad un'altra organizzazione spesso crea agitazione nelle persone. Gli utenti aziendali esitano nell'adottare questi nuovi servizi se non hanno la certezza che le loro informazioni siano crittografate e sicure. Tuttavia le aziende che forniscono sistemi basati sul cloud puntano molto sulla reputazione guadagnata nel tempo. Sta quindi alle società offrire una buona qualità dei servizi in modo tale da non perdere la propria clientela. I clienti comprano servizi fintanto che vengono assicurate tutte le misure di sicurezza, altrimenti questi acquirenti abbandonano il servizio. Poiché i dati sono accessibili da qualunque postazione, è possibile che anche la privacy

dell'utente sia compromessa. Un modo per risolvere questo problema è l'utilizzo di appropriate tecniche di autenticazione. Un'altra soluzione possibile è di prevedere un meccanismo di autorizzazione, in modo tale che ogni utente possa accedere solamente ai dati e alle applicazioni utili al suo lavoro.

I tempi di replicazione e i costi giocano un altro importante ruolo. Infatti minore è il tempo di replicazione dei dati, maggiore è la robustezza e integrità dei dati.

Infine l'affidabilità è un problema: i server nel cloud possono soffrire degli stessi problemi dei server interni ad una azienda. Infatti un eventuale down delle strutture può accadere anche nel cloud computing.

Capitolo 2

La progettazione nel Cloud

Risorse di rete, software e servizi basati sul web sono condivisi tramite *multi-tenancy* e sono rilasciati *on-demand* ai clienti. È appunto questo il principio cardine della condivisione che permette di ottenere tutti i benefici del cloud computing: i provider di questi servizi possono così ammortizzare i costi attraverso più clienti.

Questo cambiamento di paradigma nell'infrastruttura computazionale è una logica conseguenza della facilità di accesso da remoto e dal *virtual computing* fornito da Internet. Il cloud può inoltre risolvere le tradizionali tensioni che si creano nei progetti di sviluppo software creando così quell'infrastruttura flessibile richiesta per avviare velocemente i progetti e quegli strumenti che permettono di realizzare un'interazione costruttiva tra gli investitori interni all'azienda e i team di sviluppo esterni.

Costruire applicazioni sulle piattaforme cloud è assai più rapido rispetto alle tradizionali applicazioni *on-premise*. Infatti queste offrono la possibilità di utilizzare moltissimi tools che possono aiutare a creare potenti applicazioni business, mobile ma anche siti Web. Poiché l'intera applicazione risiede sul cloud, i clienti non devono preoccuparsi dell'infrastruttura IT, degli upgrade, dell'uptime o dei backups.

Il Cloud Computing è stato visto come l'architettura di nuova generazione delle aziende IT. In contrasto con le soluzioni tradizionali, dove i servizi

IT sottostanno ad appropriati controlli fisici e logici, il Cloud sposta l'applicazione software e i database verso enormi data center, dove il controllo dei dati e dei servizi non sarà più così affidabile. Inoltre differisce dal classico modello client/server poiché le applicazioni vengono fornite da un server ed vengono eseguite e controllate dal browser dell'utente, senza dover installare necessariamente una versione client di questi applicativi sul PC. La centralizzazione dà ai provider cloud il controllo completo sulle versioni dei programmi *browser-based* rilasciati ai clienti, cosa che elimina la necessità di aggiornamenti o di controlli sulle licenze sui computer locali degli utenti.

2.1 Uno sviluppo lento e inefficiente

Con il passare del tempo il modo tradizionale di creare ed eseguire applicazioni business è diventato esageratamente complesso e scomodo. Esistono troppe parti da comprare, installare, configurare e di cui curare la manutenzione, siano esse hardware o software. Inoltre l'intera infrastruttura richiede manutenzione costante in modo tale che possa lavorare senza problemi.

Questo pesante sovraccarico impone quindi delle barriere negli standard di sviluppo delle applicazioni. Una complessità dell'ambiente computazionale porta al fatto che qualsiasi piccolo cambiamento può generare ripercussioni su tutta l'organizzazione. Questo riduce la reattività generale, riducendo la capacità di un'azienda di rispondere costantemente al cambiamento delle necessità del business. Invece lo sviluppo dei programmi per aziende procede molto lentamente, lasciando dietro di sè moltissimi lavori arretrati. Il risultato finale è che i manager non ottengono quelle applicazioni che avevano richiesto per il loro business iniziale e tutto questo si conclude spesso in un miscuglio di sistemi non integrati, costruiti su *spreadsheet* o database personali o qualsiasi altra piattaforma non supportata.

2.2 Un nuovo orizzonte

Il cloud computing è facile da comprendere: lo sviluppo delle applicazioni (ma anche la loro esecuzione) avviene totalmente sul Web e tutto ciò di cui si ha bisogno per accedervi è un browser. Solamente con l'utilizzo di una connessione Internet, gli sviluppatori possono accedere all'ambiente di sviluppo del programma con tutti i suoi tool e le sue risorse. In questo modo possono costruire applicativi completi senza il costo e la complessità di comprare e mantenere una propria infrastruttura di sviluppo. Gli sviluppatori possono essere aggiunti in qualunque parte del mondo ed avere rapido accesso a tutte le risorse utili allo sviluppo dell'applicazione Quest'ultimi possono velocemente compilare e rilasciare soluzioni, riducendo il tempo di costruzione del programma, massimizzando le risposte alle richieste degli utenti e dando quell'agilità dei processi IT adatta ad ottenere vantaggi sulle ultime opportunità di business.

2.3 Impatto del Cloud sulla progettazione

Nel repentino cambiamento dell'ambiente computazionale verso i servizi Web e le piattaforme cloud, lo sviluppo del software sta diventando sempre più impegnativo. Questo processo coinvolge piattaforme eterogenee, servizi distribuiti, molteplici aziende sparse nel mondo. Molto importante diventa a questo punto la fase di raccolta dei requisiti, che include l'interazione tra clienti, utenti e ingegneri del software. Il cambiamento di questi è infatti la causa maggiore dell'aumento della complessità crescente e della presenza di discrepanze nel budget. Apportare modifiche ad uno stato avanzato della progettazione del software farebbe crescere il costo del progetto esponenzialmente. Aggiungere altri programmatori a questo punto non risolverebbe i problemi relativi ai tempi di sviluppo anzi nascerebbe la necessità di coordinare le parti e ciò rallenterebbe tutto il processo. In conclusione è molto importante che la raccolta, la pianificazione e la modellazione dei requisiti di un software sia svolta coinvolgendo tutte le parti fin dall'inizio.

A questo punto è necessario includere i provider cloud, poiché essi provvederanno all'infrastruttura computazionale e al suo mantenumento. I provider conosceranno soltanto le dimensioni, i dettagli architetturali, la strategia di virtualizzazione e la percentuale di utilizzo delle risorse di tale impianto. Questi nuovi protagonisti potranno aiutare a rispondere alle domande relative al quantitativo di sviluppatori necessari, al riutilizzo dei componenti, alla stima dei costi e del piano di lavoro, alla gestione dei rischi, della configurazione e degli eventuali cambiamenti da effettuare e all'assicurare la dovuta qualità.

A causa del riutilizzo dei componenti nei servizi Web la dimensione di un software in termini di migliaia di linee di codice o di punti funzionali da sviluppare interamente si riduce sensibilmente ma la complessità di un progetto cresce enormemente grazie alla mancanza di documentazione con i dettagli implementativi e dei loro requisiti utili per l'integrazione. L'unica descrizione disponibile online è costituita dalle informazioni presenti sui metadati del servizio che vengono automaticamente generate dai computer.

Solamente le fasi di sviluppo del codice e di testing possono essere svolte indipendentemente tra gli sviluppatori: esse possono essere svolte sulla piattaforma cloud, il che è un enorme beneficio poiché ognuno può avere facile accesso al software in costruzione. Tutto questo riduce il costo e i tempi di testing e di conferma.

Ma gli sviluppatori software devono usare servizi Web e software open-source resi disponibili liberamente dal cloud piuttosto che procurarselo in altro modo. Sarebbe quindi utile che questi fossero esperti nel costruire un software sfruttando componenti già fruibili piuttosto che riscrivere tutto da capo e creare un'applicazione monolitica. Il *refactoring* di un programma esistente è dunque una conoscenza richiesta per utilizzare al meglio l'architettura dell'infrastruttura cloud in un'ottica di risparmio. Al giorno d'oggi, inoltre, i computer possiedono processori multicore e sono collegati in rete per cui gli ingegneri del software dovranno essere esperti anche nel calcolo distribuito e in parallelo per sfruttare al meglio queste nuove tecnologie.

Inoltre dovranno essere preparati sui protocolli Internet, XML, standard dei Web Service e architettura SOA (acronimo di *Service Oriented Application*) per sfruttare tutti i benefici del Web 2.0. I provider di servizi cloud insistono sul fatto che il software deve essere il più modulare possibile per poter consentire l'occasionale migrazione da un server ad un altro per il cosiddetto *load balancing*, che consiste nel distribuire il carico di uno specifico servizio, ad esempio un sito web, tra più server. Anche la fase di manutenzione deve includere i provider. Infatti tale compito è completamente demandato a loro piuttosto che agli sviluppatori veri e propri. Inoltre a causa della partecipazione di tali provider, i clienti devono firmare un contratto con loro in modo tale da evitare che il codice etico dell'ingegneria del software sia violato dagli stessi provider. In aggiunta protezione e sicurezza dei dati sono gli aspetti più importanti di loro competenza.

L'occasionale richiesta di un utilizzo maggiore delle risorse CPU e di rete da parte delle applicazioni potrebbe mettere in pericolo il modello *pay-by-use* poiché più programmi potrebbero richiedere un utilizzo di risorse ancora maggiore, tutte nello stesso momento, situazione non prevista inizialmente dai provider. Per risolvere tale problema, questi ultimi utilizzano tecniche di virtualizzazione delle risorse in modo tale da soddisfare molti clienti su richiesta in modo efficiente. Nel caso in cui ci sia bisogno di più risorse potrebbe essere necessario spostare l'applicazione da un server ad un altro oppure da un dispositivo di storage ad un altro. Questo potrebbe portare ad un conflitto di interessi con il cliente poiché quest'ultimo potrebbe richiedere risorse dedicate con alta disponibilità e affidabilità per le sue applicazioni. Per evitare tutto ciò i provider devono garantire una certa qualità del servizio in previsione di utenti con alta priorità.

Quanto può essere difficile l'interazione tra provider cloud e ingegneri del software? L'ammontare di interazioni tra ingegneri del software e provider cloud dipende dal tipo di cloud desiderato, sia esso pubblico, privato o ibrido. In quello privato c'è molto più controllo e amministrazione da parte del cliente stesso rispetto a quello pubblico. Inoltre un utente dovrebbe considerare di

utilizzare una piattaforma privata rispetto ad una pubblica per assicurarsi maggior disponibilità e affidabilità per le sue applicazioni più importanti. Ma il minor costo su piattaforma pubblica avrà sempre maggior peso rispetto ai benefici della minor complessità dello sviluppo del software su piattaforma privata e quindi risulta più allettante.

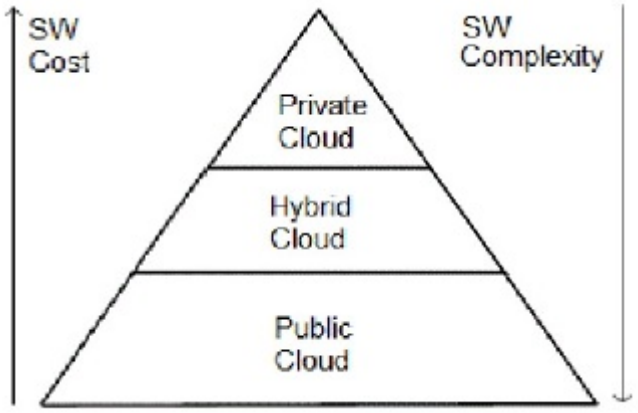

Figura 2.1: Costo vs Complessità nello sviluppo cloud

Infine è importante sottolineare i due più grandi aspetti positivi che riguardano il cloud: **adattabilità** e **riduzione dei costi**.

Adattabilità

Un'azienda può ottenere risorse computazionali implementate in pochissimo tempo per una parte del costo di una soluzione *on-premise*, per poi abbandonare altrettanto facilmente. I dipartimenti IT sono liberi di bilanciare la capacità a seconda dell'utilizzo in tempo reale, senza fare nessun altro intervento di rete o investimento a livello di hardware e storage. Gli utenti possono accedere alle informazioni da qualsiasi luogo piuttosto che rimanere alle loro postazioni.

Riduzione dei costi

Il cloud computing segue un modello nel quale i costi dei servizi sono basati sul consumo e fanno uso di un'infrastruttura altamente condivisa. Le aziende pagano solamente ciò che utilizzano e i provider possono distribuire le spese su più clienti.

Capitolo 3

Progettazione e Sviluppo di Software-as-a-Service

3.1 Introduzione a SaaS

Ci sono ovvie differenze tra SaaS ed una architettura software tradizionale. Per questo motivo è necessario un nuovo metodo di ingegnerizzazione del software per quanto riguarda il pattern SaaS. La maggior parte delle soluzioni SaaS è basata su un'architettura *multi-tenant*. Grazie a questo modello, un'unica versione dell'applicazione con un'unica configurazione, relativa ad hardware e rete, viene utilizzata da tutti i clienti, in inglese *tenants*. Per supportare la scalabilità tale software viene installato su più macchine, realizzando il cosiddetto *horizontal scaling*. Tuttavia non tutte le applicazioni seguono questo modello. Infatti molte altre sfruttano la virtualizzazione per poter supportare un alto numero di clienti.

Poiché non tutte le applicazioni SaaS condividono le stesse peculiarità, di seguito sono elencate le caratteristiche comuni.

- **Configurazione e personalizzazione**: come nei tradizionali software enterprise, un cliente può cambiare le opzioni di configurazione e alterare le sue funzionalità e il suo *look-and-feel*. Ogni singolo cliente può avere i propri parametri per tali opzioni. L'applicazione può quindi

essere personalizzata basandosi su un insieme di opzioni standard di configurazione.

- **Rilascio più rapido di nuove caratteristiche**: le applicazioni SaaS sono aggiornate più velocemente rispetto ai software tradizionali. Questo è dovuto a molti fattori, come per esempio il fatto che l'applicazione è centralizzata e quindi nuovi rilasci possono essere caricati senza dover costringere i clienti ad installare nuovo software.

- **Protocolli adatti all'integrazione**: poiché le applicazioni SaaS non possono accedere a sistemi interni all'azienda, esse offrono dei protocolli, basati principalmente su HTTP, REST, SOAP e JSON, o API che possono operare su una rete estesa in modo da facilitare tale integrazione.

- **Funzionalità collaborative**: ispirate dal successo dei social network, molte applicazioni SaaS offrono la possibilità agli utenti di collaborare e condividiere informazioni.

3.2 Piattaforme per SaaS

Attualmente esistono molte piattaforme che permettono lo sviluppo di applicazioni SaaS. Tra le più importanti si possono citare SalesForce.com, Windows Azure e Amazon Elastic Cloud Computing (EC2). In particolare ora si analizzerà Windows Azure, entrando nel merito delle sue caratteristiche e di come possa essere sviluppata un'applicazione sfruttando tale piattaforma.

3.3 Un caso di studio: Windows Azure

La piattaforma Windows Azure astrae le risorse hardware attraverso la virtualizzazione. Ogni applicazione rilasciata su questa piattaforma gira su una o più macchine virtuali (VM). Questi software si comportano quindi come se si trovassero su un computer dedicato, mentre in realtà condividono

le risorse fisiche come lo spazio su disco, la rete o i core delle CPU con altre VM sullo stesso host. Il beneficio principale di questo livello di astrazione al di sopra dell'hardware è la portabilità e la scalabilità.

La virtualizzazione di un servizio consente di spostarlo in un qualsiasi host all'interno del centro dati. Inoltre dà la possibilità di avere sia scalabilità verticale che orizzontale. Il primo termine indica la situazione nella quale, quando la domanda cresce, può essere aumentato il numero delle risorse fisiche, come core della CPU o memoria, su una specifica VM. Il secondo significa invece che possono essere aggiunte più istanze di VM che sono copie di servizi esistenti. Tutte queste istanze sono bilanciate a livello di rete in modo tale che altre richieste in arrivo siano distribuite tra di loro.

3.3.1 Componenti

La piattaforma Windows Azure include tre componenti principali:

- **Windows Azure** fornisce un ambiente di calcolo per le applicazioni basato su *Microsoft Windows Server* e uno spazio di storage per dati strutturati e non, come anche per messaggistica asincrona.
- **Windows Azure AppFabric** fornisce un insieme di servizi che possono essere utili per collegare utenti e applicazioni locali alle soluzioni cloud, per controllare le autenticazioni ed implementare il controllo dei dati e altri aspetti come per esempio il caching.
- **SQL Azure** è invece essenzialmente *Microsoft SQL Server* che fornisce un servizio sul cloud.

La piattaforma include inoltre un insieme di servizi di gestione che consente di controllare tutte queste risorse, sia attraverso un'interfaccia utente da Web sia programmaticamente. Nella maggior parte dei casi esistono API basate su REST che possono essere usate per determinare come dovrà lavorare il servizio sviluppato. Infine esiste un set di tool e kit di sviluppo software (*Software Development Kits*, SDK) che permettono di sviluppare, testare e rilasciare la propria applicazione.

3.3.2 Ambiente d'esecuzione

L'ambiente d'esecuzione di Windows Azure consiste in una piattaforma per le applicazioni e i servizi che viene hostata all'interno di una o più parti. Le parti implementabili sono le seguenti:

Azure Compute

Un'applicazione è costituita da una o più parti che girano all'interno dei data center Azure. Tipicamente esiste almeno una parte **Web** che viene utilizzata per l'accesso degli utenti all'applicazione stessa. Possono poi esistere altre parti, chiamate **Worker**, che sono usate per eseguire processi in background e per supportare le parti Web.

Parte VM

Questa parte permette di creare la propria istanza di un sistema operativo *Windows Server* all'interno dei data center Azure.

3.3.3 Gestione dei dati

Windows Azure, SQL Azure e i servizi associati danno la possibilità di salvare e gestire i dati in vario modo.

Azure Storage

Fornisce quattro servizi principali per uno storage di dati persistente e sicuro nel cloud. Tale servizio supporta un'interfaccia REST alla quale si può accedere sia da applicazioni hostate su Azure che da remoto.

- **Azure Table Service**: fornisce un meccanismo di storage strutturato su tabelle, basato quindi sul familiare formato righe/colonne, supportando anche query per la gestione dei dati. È principalmente mirato per le soluzioni dove si rende necessario un grande volume di dati da salvare, mantenendo facilità di accesso e aggiornamento.

- **Binary Large Object (BLOB) Service**: fornisce una serie di contenitori per effettuare lo storage di dati testuali o binari. Supporta sia contenitori **Block BLOB** per lo streaming di dati, sia **Page BLOB** per operazioni random di lettura/scrittura.

- **Queue Service**: fornisce un meccanismo per messaggistica sicura tra istanze di una parte, come per esempio una parte Web e una Worker.

- **Windows Azure Drives**: fornisce alle applicazioni la possibilità di montare un NTFS VHD [1] come un Page BLOB, di scaricare e caricare online questi VHD sfruttando il BLOB.

SQL Azure Database

Questo è un database cloud altamente scalabile costruito grazie alle tecnologie di *SQL Server* e supporta il modello relazionale basato su T-SQL[2]. Può essere usato con applicazioni hostate su Windows Azure, ma anche con altre applicazioni locali.

SQL Azure Data Sync

È un servizio di sincronizzazione dati basato sul cloud costruito grazie alle tecnologie *Microsoft Sync Framework*. Fornisce una sincronizzazione bidirezionale e capacità di gestione dei dati in modo tale da poter condividere questi dati tra vari database SQL Azure o tra database remoti e Azure.

[1] NTFS è l'acronimo di New Technology File System e indica il file system dei sistemi operativi basati sul kernel NT, sviluppato da Microsoft. VHD è l'acronimo di Virtual Hard Drive e indica un formato di un disco virtuale, ossia un file contenitore che include gli elementi di un normale hard disk fisico, come file, cartelle e partizioni.

[2] T-SQL è l'acronimo di Transact-SQL che è l'estensione proprietaria del linguaggio SQL sviluppata da Microsoft.

Caching

Questo servizio fornisce una modalità di caching molto avanzata che non richiede nessuna installazione o configurazione e che può andare ad aumentare o diminuire dinamicamente la dimensione di tale cache.

3.3.4 Servizi di rete

Windows Azure fornisce un certo numero di servizi di rete attraverso i quali poter massimizzare le performance, implementare le autenticazioni e migliorare la facilità d'uso delle applicazioni.

Content Delivery Network (CDN)

Il CDN dà la possibilità di depositare pubblicamente in cache dei dati per applicazioni in luoghi strategici che sono vicini, in termini di rete, agli utenti finali. Il CDN utilizza un certo numero di data center in determinate locazioni nel mondo, i quali salvano i dati nel BLOB che ha un accesso anonimo. Non vi è quindi la necessità che questi si trovino dove l'applicazione sta girando.

Virtual Network Connect

Questo servizio permette di configurare le parti di un'applicazione che gira su Windows Azure e sui computer locali in modo tale che possano sembrare sulla stessa rete. Sfrutta un agente software sui computer locali che stabilisce una connessione sicura IPsec[3] alla parte Windows Azure sul cloud, in modo tale da fornire la possibilità di amministrare, gestire, monitorare ed effettuare il debug delle parti direttamente.

[3]IPsec è l'abbreviazione di IP Security, uno standard per reti a pacchetto che si prefigge di ottenere connessioni sicure su reti IP.

Virtual Network Traffic Manager

Questo servizio permette di configurare redirecting e bilanciamento basandosi su tre diversi metodi, in modo tale da massimizzare le performance, reindirizzando le richieste degli utenti alle istanze nei data center più vicini.

Access Control

Questo è un servizio basato su standard per l'identificazione e il controllo dell'accesso. Agisce come un *Security Token Service* (STS) e facilita l'utilizzo di tecniche di autenticazione dove l'identità dell'utente deve essere confermata in domini esterni a quelli in cui risiede l'applicazione.

Service Bus

Fornisce una messaggistica sicura per applicazioni distribuite o ibride senza la necessità di un firewall complesso o infrastrutture di sicurezza. Può utilizzare un set di protocolli di comunicazione in modo tale da assicurare l'affidabilità dei messaggi.

3.3.5 Creare un servizio

Un servizio in Windows Azure è formato da un'applicazione che deve girare su tale servizio e da file di configurazione XML che definiscono come tale servizio debba girare. Il seguente diagramma mostra i componenti di un servizio su Windows Azure.

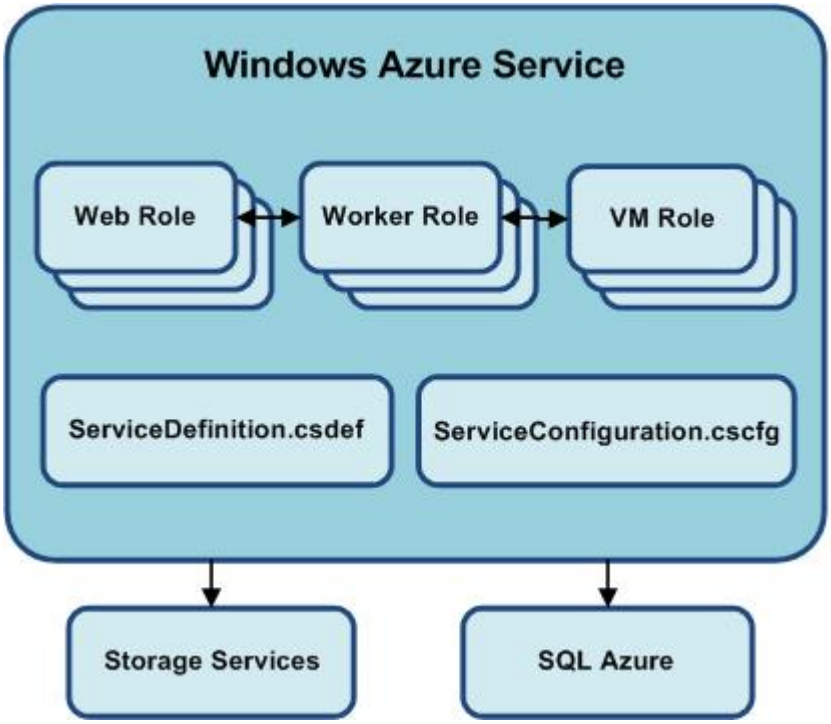

Figura 3.1: Servizio Windows Azure

Il modello del servizio è determinato dalle impostazioni presenti nel file *ServiceDefinition.csdef* ed è configurato attraverso il file *ServiceConfiguration.csdef*. Il file di definizione contiene il codice binario quando l'applicazione è pronta per essere rilasciata. L'altro file viene invece utilizzato da Windows Azure per determinare come l'applicazione debba girare.

Definendo le impostazioni del file *ServiceDefinition.csdef* vengono determinate le parti e le risorse di un'applicazione. Questa implementa una o più istanze dei tipi disponibili di parti. In Windows Azure le istanze di una parte

sono replicate fra più computer in modo tale da implementare tutte o parte delle funzionalità del servizio.

Definizione delle parti

Attualmente Windows Azure supporta le seguenti parti.

- **Parte Web**: è una parte che è personalizzabile per la programmazione di applicazioni Web sfruttando il supporto di IIS 7 e ASP.NET. Il beneficio derivante dall'utilizzo di questo tipo di parte è che la configurazione di IIS è già pronta. Per questo motivo tale parte è la migliore da utilizzare per fornire un *frontend* Web per il servizio. Non è adatta a processi di lunga durata.

- **Parte Worker**: è una parte che è utile per uno sviluppo generalizzato, e può eseguire processi in background per la parte Web. Quando vi è la necessità di far girare un processo in background con operazioni lunghe o occasionali bisognerebbe sfruttare questa parte.

- **Parte VM**: è tipo speciale di parte che consente di definire la configurazione e gli aggiornamenti del sistema operativo per la macchina virtuale. Mentre la parte Web e quella Worker girano su una macchina virtuale, questa parte è proprio la macchina virtuale e dà pieno controllo sulle operazioni. Quando si rendono necessarie lunghe e complicate installazioni nel sistema operativo oppure si riscontrano particolari questioni riguardanti la configurazione del sistema bisognerebbe utilizzare questa parte. Tale parte è inoltre adatta per la migrazione di applicazioni esistenti in modo da farle girare come servizi su Windows Azure.

Permessi di accesso

Dopo aver creato il servizio che contiene l'applicazione, si può accedere da remoto ad un'istanza della parte in modo tale da configurare le impostazioni

della macchina virtuale. Per fare questo bisogna assicurarsi di aver caricato il certificato appropriato per collegarsi al *Windows Azure Management Portal*, criptare la password utilizzata e aggiungere i moduli appropriati nel file di definizione.

Comunicazione tra le parti

Le istanze di una parte del servizio comunicano attraverso connessioni interne o esterne che dipendono dal tipo di comunicazione richiesta. Tali connessioni possono essere effettuate sfruttando HTTP, HTTPS e le API di Microsoft .NET per le socket TCP/IP. Una connessione esterna viene chiamata input endpoint, una interna internal endpoint. Gli endpoint sono associati a delle porte, in particolare gli endpoint esterni sono associati ad una porta predefinita mentre quelli interni sono assegnati dinamicamente da Windows Azure.

Operazioni iniziali

Potrebbe rendersi utile il lancio di operazioni prima che l'istanza parta effettivamente. Queste possono essere, per esempio, installazione di componenti o configurazione delle chiavi di registro e possono essere aggiunte alla parte modificando il file di definizione.

Creazione dell'applicazione

La classe *RoleEntryPoint* include metodi che sono chiamati da Windows Azure per far partire, girare o fermare una parte Web o Worker. Tali metodi possono essere sovrascritti per configurare determinate opzioni di inizializzazione.

Capitolo 4

SaaS Multi-Tenant

4.1 Introduzione

Tra le varie possibilità di sviluppo di un'applicazione SaaS, è interessante analizzare il caso *Multi-Tenant*. Tale approccio è descritto nell'articolo *A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform*, redatto da un gruppo di sviluppatori del *Korea Adavanced Institute of Science and Technology*.

4.2 Livelli dell'architettura

Architetturalmente parlando, le applicazioni SaaS sono bene o male simili alle altre applicazioni costruite secondo i principi di design *service-oriented*.

I *Process Services* presentano interfacce che i livelli di presentazione Web possono invocare e fanno partire un flusso di lavoro sincrono o lunghe transazioni che invocheranno i cosiddetti *Business Services* che interagiscono con i rispettivi data-store in modo tale da leggere e scrivere i dati. I *Security Services* sono invece responsabili del controllo degli accessi.

La più importante differenza è l'aggiunta dei *Metadata Services* che si occupano di controllare la configurazione dell'applicazione per ogni singolo cliente. I servizi possono interagire con i *Metadata Services* in modo tale

da ottenere informazioni che descrivono la configurazione specifica di ogni cliente.

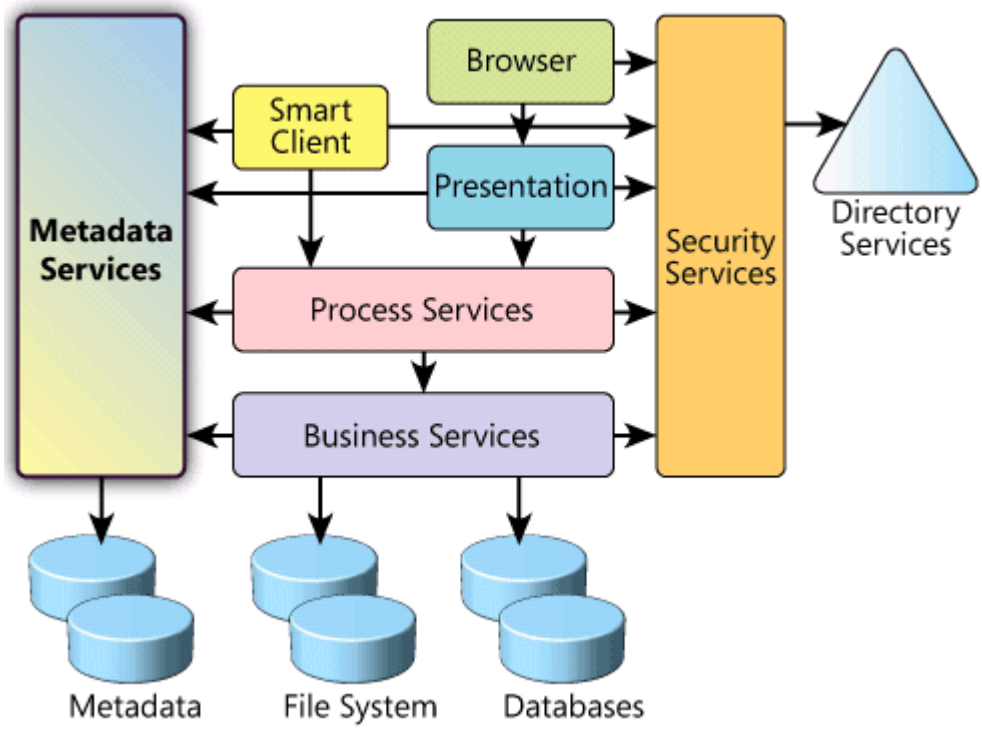

Figura 4.1: Livelli dell'architettura SaaS Multi-Tenant

4.3 Sviluppo di una applicazione SaaS

Un sistema ben modellato di un'applicazione SaaS possiede tre proprietà principali: deve essere costruito su misura per l'utente, elastico ed essere efficiente per la moltitudine di utenti che lo utilizzeranno. Questi requisiti, funzionali e non, necessitano, nella fase di realizzazione, di un nuovo modo di sviluppare il software.

Fondamentalmente le applicazioni SaaS possono essere suddivise in due grandi gruppi: le applicazioni Business e le applicazioni BPM.

Le prime presentano un'interfaccia utente *web-based* aperta a tutti i vari clienti che usufruiscono di essa. Lo scopo di tali applicazioni (ad esempio

CRM o ERP) è di processare transazioni business e collaborazione tra gli utenti con dati che risiedono nel DBMS, acronimo di *DataBase Management System*. Utenti diversi possono avere ruoli differenti come per esempio amministratore, manager, impiegato o qualsiasi altro incarico che possa avere le proprie autorizzazioni per accedere ai dati aziendali e alle logiche di business. A seconda dei ruoli e dei loro permessi di accesso, vengono fornite agli utenti determinate pagine web per richiedere servizi attraverso i browser web. Tali richieste vengono demandate all'*Application Server* che contiene le logiche di business che processano le varie transazioni. Gli aspetti principali di un'applicazione di questo genere sono quindi il modello dei dati, la logica di business, le pagine web e i ruoli.

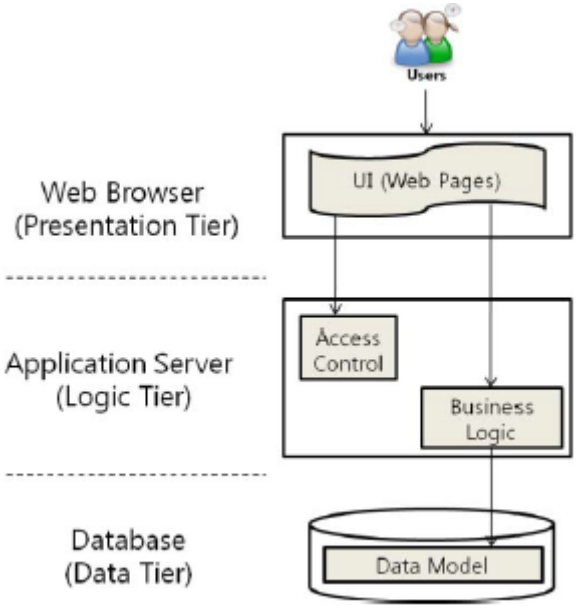

Figura 4.2: Applicazione Business

Oltre a questa tipologia, esistono le applicazioni *Business Process Management* (BPM). In contrasto con le prime che sono utilizzate per condividere informazioni e collaborazione fra gli utenti, le applicazioni BPM si incentrano sul flusso delle attività di un'azienda.

Lo scopo delle applicazioni BPM è di modellare ed eseguire processi di business che sono collezioni di attività correlate e strutturate di un'azienda. I processi business sono modellati da un analista o da un manager dell'organizzazione. Tali processi vengono salvati sotto forma di flusso di lavoro nel cosiddetto *Process Repository*. Un flusso di lavoro è una rappresentazione di una sequenza di operazioni che esegue il processo di business modellato usando un motore apposito. Ogni membro dell'azienda possiede un'interfaccia utente in modo da poter controllare e processare attività a lui assegnate. Gli aspetti principali di un'applicazione BPM sono quindi flusso di lavoro e attività.

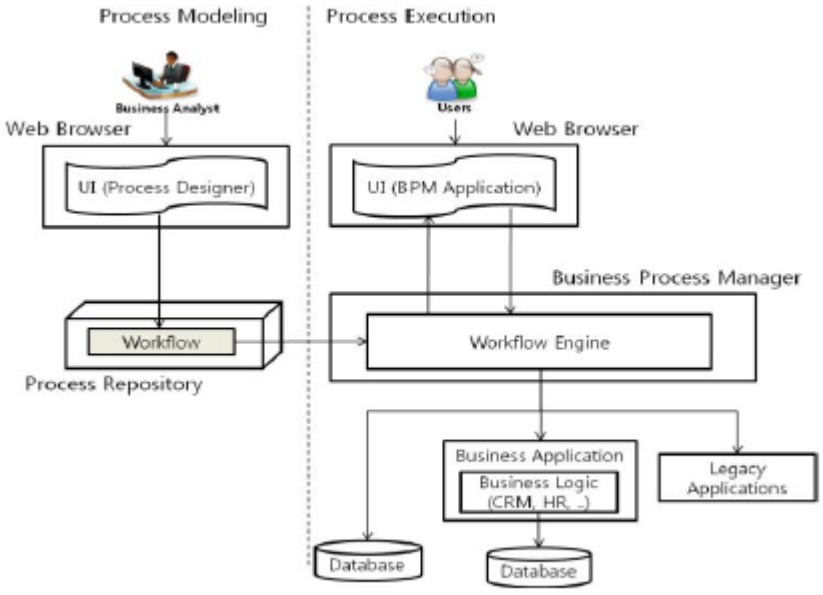

Figura 4.3: Applicazione BPM

4.4 Aspetti configurabili

Anche se la maggior parte delle aziende richiede un'applicazione SaaS standard rilasciata sulla piattaforma, ogni cliente avrà requisiti diversi per quanto riguarda componenti del servizio come la struttura organizzativa, ossia definizione dei ruoli e controllo degli accessi, interfaccia utente, modello dei dati, flusso di lavoro e logica di business. Le seguenti sottosezioni descrivono gli aspetti configurabili di un'applicazione SaaS.

a. Struttura Organizzativa

La configurabilità della struttura organizzativa è il primo aspetto essenziale perché è il vero unico aspetto che può cambiare spesso. Un manager dell'azienda può aggiungere o rimuovere ruoli basandosi su quelli creati inizialmente dallo sviluppatore dell'applicazione SaaS. Inoltre può creare un certo numero di livelli di autorizzazione per accedere al modello dei dati ed assegnarli ad ogni determinato ruolo all'interno dell'organizzazione. Dopo che tale manager ha eseguito queste operazioni, la piattaforma SaaS controlla le autorizzazioni ogni volta che si accede al modello dei dati o viene richiesto un servizio dall'utente. Per esempio, ai dati di sicurezza di un'azienda può accedere la direzione ma non gli impiegati.

b. Interfaccia Utente

La configurazione dell'interfaccia utente mira a cambiare il *look-and-feel* di un'applicazione. Il manager può cambiare colore ai componenti di una pagina web, ristrutturarne il layout, aggiungere/togliere *hyperlink* o introdurre nuovi componenti in modo tale da poter interagire con un nuovo modello dei dati. Inoltre è possibile differenziare i componenti del servizio in base al diverso ruolo nell'organizzazione. Nel frattempo è importante considerare l'impatto che può avere il cambiamento di un componente su altri aspetti. Per esempio, se si aggiunge un nuovo campo di dati ad un oggetto, verrà aggiunta una nuova colonna nella griglia dei componenti che visualizza tale

oggetto. Analogamente se un oggetto fosse cancellato dal database, la pagina che lo controlla deve diventare necessariamente inaccessibile.

c. Modello dei Dati

La configurazione del modello dei dati implica la possibilità di aggiungere/togliere oggetti dal database, di aggiungere/togliere campi da tali oggetti e di cambiare anche il loro nome, tipo, ecc. Il manager può accedere al DBMS della piattaforma e controllare il modello dei dati del software SaaS. Inoltre può configurare le autorizzazioni di accesso relative ad ogni ruolo per ciascun modello e per ciascun campo.

d. Flusso di Lavoro

Poiché diverse organizzazioni possono avere diverse strutture organizzative, processi di *decision making* e ruoli di business, i flussi di lavoro modellati dallo sviluppatore dell'applicazione dovrebbero essere configurati da ogni cliente. Utilizzando i tool di *Process Designer*, il manager può configurare varie componenti del processo di business in termini di flussi di lavoro, tipi di attività e regole di business.

e. Logica di Business

La logica di business è la principale funzionalità delle applicazioni business che tratta lo scambio di informazioni tra il database e l'interfaccia utente. Per questo motivo una configurazione totalmente aperta ad ogni possibile cliente può far perdere la vera sostanza dell'applicazione originale. Tuttavia è molto frequente, da parte dei clienti, la richiesta di aggiungere nuove logiche di business. In questo caso il manager può crearne una semplice sfruttando un template già pronto. Quest'ultimo fornisce molti metodi per manipolare il modello dei dati e restituire i risultati delle operazioni effettuate sui dati del database. A questo punto tale logica appena generata viene caricata dinamicamente sulla piattaforma e messa a disposizione attraverso uno specifico

URL. L'ultimo passo da compiere è quello di creare una nuova pagina con un'interfaccia grafica e collegarla a tale URL, in modo da renderla disponibile all'utente.

4.5 Multitenancy attraverso i Metadata

Poiché viene fornita un'applicazione SaaS a più clienti attraverso una singola istanza del servizio, l'architettura della piattaforma deve permettere la configurazione da parte dei clienti senza cambiare il codice sorgente del software per ogni singolo cliente e senza sospendere il servizio durante ogni configurazione. Per risolvere queste problematiche entra in gioco l'architettura *metadata-driven*.

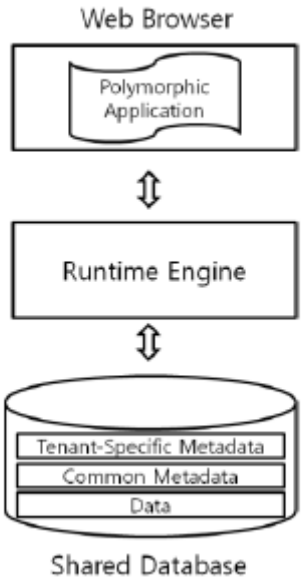

Figura 4.4: Architettura Metadata-driven

In tale architettura ogni aspetto dell'applicazione SaaS che è configurabile dai clienti viene posizionato in un *Metadata Database*. Quando il manager configura alcuni aspetti dell'applicazione, le informazioni vengono immagazzinate nel metadata specifico del cliente. Un motore runtime genera l'appli-

cazione polimorfica ad hoc per il singolo cliente utilizzando il codice di base del software e il metadata specifico. Attraverso questa applicazione all'utente sembra di utilizzare il proprio software business mentre invece l'istanza del servizio è condivisa fra tutti i clienti. Anche se i dati sono condivisi, essi vengono mantenuti al sicuro dall'applicazione polimorfica che permette un accesso individuale a questi sfruttando delle query ottimizzate.

4.6 Isolamento dei dati

Per un'applicazione destinata ad un unico cliente i dati sono racchiusi nel medesimo database senza dover incorrere nei problemi del loro isolamento. Tuttavia un'importante differenza è che tale aspetto è la prima cosa da considerare nell'ottica dello sviluppo cloud. In questo caso il database può essere dislocato su una vasta rete formata da più computer. Per assicurare che tali database siano correttamente aggiornati esistono due processi, la replicazione e la duplicazione. La prima coinvolge l'utilizzo di software specializzati per controllare cambiamenti nei database distribuiti. Una volta che vengono identificati, il processo di replicazione rende uguali tutti i database. Tuttavia tutto questo può essere molto complicato e richiedere molto tempo d'esecuzione nel caso in cui sia molto alto il numero di tali database distribuiti. La duplicazione non è invece così complicata. Fondamentalmente identifica un database come master e successivamente lo duplica. Questo processo viene normalmente eseguito ad intervalli di tempo precisi, in modo tale da assicurare che ogni locazione distribuita abbia gli stessi dati. Inoltre, sono autorizzati cambiamenti solo al database master in modo tale da assicurare che i dati locali non siano sovrascritti.

A questo punto vi sono tre tipologie di salvataggio dei dati.

- **Self-Contained Database**: questo schema prevede che ad ogni cliente sia associato un database. In questo caso l'isolamento dei dati viene assicurato nel miglior modo ed è sicuramente la modalità più sicura ma di conseguenza anche la più costosa. Il suo punto di forza è la semplificazione dell'estensioni al modello dei dati in modo tale da poter soddisfare i requisiti dei diversi clienti. In caso di guasti, recuperare i dati è relativamente semplice. Lo svantaggio sta invece nell'incrementare il numero di database da installare e quindi anche i costi di manutenzione crescono.

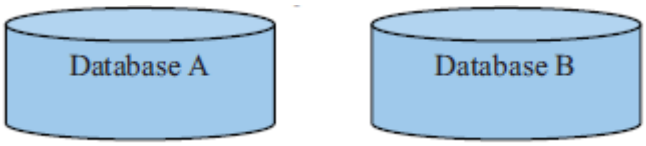

Figura 4.5: Self-Contained Database

- **Shared Database**: il secondo schema prevede la condivisione del database tra tutti gli utenti, i quali possiederanno al suo interno un loro specifico pattern di dati. Il punto di forza di tale soluzione è di fornire un isolamento logico dei dati a quei clienti che richiedono un alto livello di sicurezza anche se non c'è isolamente completo. Viene inoltre supportata la partizione a livelli in modo tale che i dati possano essere distribuiti su diversi server in modo da evitare una penalizzazione in termini di performance quando viene effettuata la loro riconvergenza. I difetti risiedono invece nel recupero dei dati in caso di guasti, poiché questa operazione, inevitabilmente, coinvolge i dati di altri clienti. Inoltre se fosse necessario tracciare delle statistiche, per esempio di utilizzo, tra i vari clienti sarebbe anche questa un'operazione assai complicata.

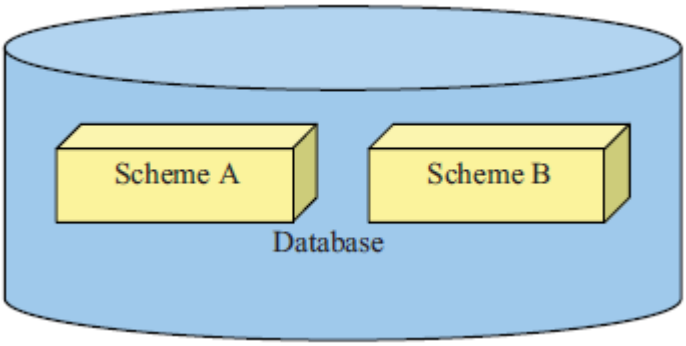

Figura 4.6: Shared Database

- **Shared Database and Shared Data Structure**: in questo terzo schema gli utenti condividono sia lo stesso database che lo stesso modello dei dati. Di conseguenza in una tabella del database, attraverso un ID, si differenziano i vari clienti e i loro dati. Questo è, ovviamente, il modello con la più alta condivisione e il minor isolamento. Infatti in questo caso costi di acquisizione e manutenzione sono molto bassi, poiché ogni database supporterà già la possibilità di avere più utenti. Tuttavia presenta anche il livello di isolamento e di sicurezza più basso in assoluto quindi è necessario, in fase di sviluppo dell'applicazione,

aumentare la sua affidabilità. Il recupero dei dati è assai difficoltoso, poiché la tabella va analizzata backup per backup in modo da poter tornare allo stato iniziale. Infatti a mano a mano che cresce il numero degli utenti, anche la tabella cresce di dimensioni e l'operazione di ricostruzione nel caso di guasti è sempre più problematica. Questa tipologia è la preferita se si vuole utilizzare un numero veramente minimo di server che verranno poi utilizzati da tutti i clienti, che dovranno però rinunciare all'isolamento dei dati in favore di una riduzione dei costi.

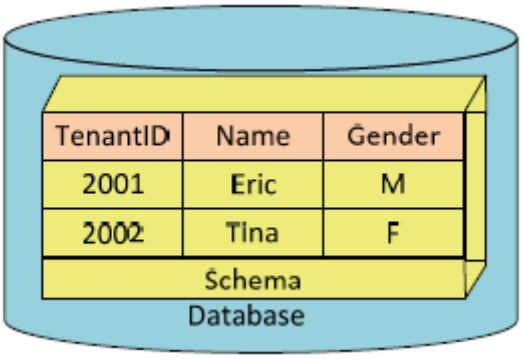

Figura 4.7: Shared Database and Shared Data Structure

Sempre in riferimento a quest'ultima modalità diventa quindi importante modellare correttamente gli aspetti relativi alla sicurezza. Per fare questo bisogna operare in due ambiti.

- **Livello di sistema**: vengono usati protocolli HTTPS e i dati vengono scambiati in SSL (*Security Socket Layer*) in modo tale da assicurare la sicurezza delle comunicazioni. Inoltre si può prevenire la distorsione dei processi di trasmissione sfruttando le firme digitali e backup automatici e frequenti dei dati importanti.

- **Livello di programma**: si può utilizzare un sistema di autorizzazioni sui dati, validazione *client-side* in modo da prevenire attacchi JS, SQL injection, ecc. Inoltre è possibile inserire, come ulteriore controllo, la verifica di inserimento delle password e dei captcha.

4.7 Ottimizzazione del Database

Quando più utenti utilizzano un'unica istanza del database, la concorrenza e l'ammontare dei dati del sistema cresce enormemente. In questo caso, se si continuasse a seguire la solita modalità di sviluppo del software, si avrebbe come risultato un decadimento delle performance del sistema. Per questo motivo diventa assai utile l'ottimizzazione del database.

a. Costruire Indici Appropriati

Tali indici devono possedere una forte selettività, in modo tale che siano nel minor numero possibile. Infatti un loro aumento potrebbe portare ad operazioni di revisione e cancellazione sul database in fase di replicazione dei dati, riducendo quindi le performance generali e aumentando lo spazio occupato.

b. Evitare troppe connessioni fra tabelle

Il tempo impiegato ad eseguire una query SQL non dipende solamente dagli indici. Infatti, operata la precedente ottimizzazione, si può vedere come il tempo relativo alla connessione fra le tabelle sia veramente elevato. Nella piattaforma SaaS il numero di utenti è altissimo. Quindi per le query su una singola tabella le performance non sono un grande problema se si usa la corretta metodologia di *indexing*, ma per più tabelle tali problemi vengono ovviamente alla luce.

c. Evitare query SQL troppo complicate

Tali query occupano, ovviamente, molto tempo. Andranno quindi divise in più query da eseguire singolarmente.

4.8 Implementazione dell'Interfaccia

In un'applicazione tradizionale, l'interfaccia del sistema soddisfa sempre i requisiti basilari per l'utente perché essa è costruita a misura delle richieste fatte dal cliente. Ma in un'applicazione SaaS usare lo stesso set di interfacce non potrà sicuramente accontentare tutte le necessità dei clienti. Così diventa importante anche considerare le caratteristiche configurabili dell'interfaccia dell'applicazione.

La pagina può essere suddivisa in menù di sistema ed elementi della pagina stessa.

- **Menù di sistema**: lo stesso menù per più clienti potrebbe avere nomi completamente differenti. Si rende quindi necessaria la totale personalizzazione di tale elemento.

- **Elementi di pagina**: similmente al menù, anche i contenuti delle varie pagine possono variare tra gli utenti. Diversi clienti possono avere viste differenti, in termini di numero di elementi nella pagina, posizione, ordine.

Tutti questi aspetti possono essere modellati sfruttando le proprietà dei metadata elencate nella sezione precedente.

Capitolo 5

Cloud-Aware e Cloud-Agnostic

5.1 Approcci per l'architettura

Analizzando gli approcci di sviluppo di un'applicazione cloud, si possono suddividere in due macrocategorie.

- **Cloud agnostic**: in questo approccio non ci si cura del fatto che l'applicazione sia realizzata sul cloud dove le risorse computazionali richieste dal software sono virtuali. Per questo la maggior parte decide di seguire le stesse linee guida e le stesse tecniche di sviluppo di una tradizionale applicazione non residente in ambiente cloud.

- *Cloud aware*: questo approccio prende in considerazione il fatto che le risorse computazionali relative all'applicazione sono virtuali e possono essere scalate (nota) e manipolate in modo differente rispetto a quello tradizionale.

Quando si dice che il software è a conoscenza della natura virtuale della piattaforma sottostante, non significa che l'applicazione abbia la necessità di invocare alcune API speciali fornite dalla virtualizzazione di basso livello in modo da ottenere servizi dalla piattaforma cloud. Piuttosto ciò significa che prima di tutto lo sviluppatore deve conoscere l'architettura della piattaforma sottostante che è altamente distribuita e virtuale; successivamente egli

stesso deve cercare di sfruttare una piattaforma così distribuita, virtuale e flessibile per soddisfare i requisiti non funzionali desiderati dall'applicazione in questione.

Ci sono però alcuni aspetti da tenere in considerazione in entrambi gli approcci. Il primo di questi riguarda i costi delle risorse. Risorse come CPU, storage e banda di rete sono calcolate sulla base dell'utilizzo ed ognuna può avere un costo differente. Altro aspetto sono gli scambi tra CPU, storage e rete che giocano un importante ruolo nell'ottimizzazione del costo di esecuzione di un'applicazione sulle piattaforme cloud. Infine è importante dire che prevedere la capacità di vendita sul cloud non è del tutto simile ad un ambiente non-cloud.

5.2 Requisiti non funzionali

a. Deployment

Property	Cloud-aware	Cloud-Agnostic
IDE integration of deployment tools	worse	same
Identical environment for production and non-production deployments	worse	same
Availability of hot-deployment option	worse	same
Deployment rollback capability	worse	same
Availability of mature automation APIs	worse	same

Tabella 5.1: Risultati delle proprietà per il Deployment

Durante le fasi di sviluppo del software gli ingegneri devono effettuare iterativamente il deploy dell'applicazione nelle loro macchine in locale o su qualche ambiente di sviluppo in comune. Tutto questo deve avvenire molto velocemente e senza lunghi periodi di attesa. Le prime due proprietà, integrazione IDE dei tools e somiglianza degli ambienti di sviluppo, ricevono un punteggio minore nell'approccio *cloud-aware* piuttosto che in quello *cloud-agnostic*. Questo perché la maggior parte delle piattaforme cloud possiedono un supporto IDE assai basilare. Allo stesso modo, la somiglianza degli ambienti di sviluppo è anch'esso insoddisfacente nel caso in cui si utilizzi il modello *cloud-aware*. Un'altra importante proprietà è la disponibilità di API. Anche in questo caso l'approccio *cloud-agnostic* risulta essere migliore, più che altro per il fatto che è la metodologia usata da più tempo, essendo il *cloud-aware* più recente.

b. Manutenibilità e modificabilità

Property	Cloud-aware	Cloud-Agnostic
Availability of modular programming support	same	same
Availability of quality IDEs	worse	same
Maturity of the APIs and their implementations	worse	same
Complexity of programming model	worse	same
Module coupling	same	same
Scaling model (horizontal vs. vertical)	worse	same

Tabella 5.2: Risultati delle proprietà per la Manutenibilità e la modificabilità

Molti fattori che influenzano la manutenibilità di un'applicazione sono correlati al grado di maturità della progettazione del software e dei processi di sviluppo. Ugualmente importante è la qualità e la maturità delle strutture a monte, come per esempio sistemi, framework e API sulle quali l'applicazione si basa. Ottenere un alto livello di manutenibilità nell'approccio *cloud-aware* è assai difficile. Infatti in questo caso è necessario un nuovo modo di risolvere i problemi, in particolare su come il modello dei dati e la concorrenza vengano gestite in un ambiente virtuale altamente distribuito. Inoltre il modello di programmazione è diverso dal ben conosciuto approccio *cloud-agnostic*. Ancora una volta il *cloud-aware*, essendo una nuova tipologia, presenta qualche svantaggio, possedendo ancora un supporto molto limitato.

c. Portabilità ed interoperabilità

Property	Cloud-aware	Cloud-Agnostic
Support for popular programming platforms. More platforms are better	worse	same
Availability of standardized APIs for basic services (persistance, messaging, etc.)	worse	same
Availability of multiple implementations of standard APIs	worse	same
Adoption and reach of standard APIs	worse	same
Maturity of the APIs and their implementations	worse	same
Release cycle length of APIs and their implementations	worse	same

Tabella 5.3: Risultati delle proprietà per la Portabilità

La maggior parte dei provider PaaS hanno delle API personali per servizi che offrono. Basarsi su API specifiche di una piattaforma, come può avvenire nell'approccio *cloud-aware*, rende portabilità ed interoperabilità piuttosto difficili. Inoltre poiché tale approccio fa uso di tecniche e framework che funzionano correttamente solo su ambienti virtuali e altamente distribuiti come il cloud, è necessaria una ridefinizione dell'applicazione se si desidera spostarsi verso un ambiente non cloud. L'approccio *cloud-agnostic*, invece, non soffre di così grandi problemi di portabilità. Infatti esso dipende da API già esistenti e sfrutta la disponibilità di standard comunemente adottati per molti servizi come per esempio la persistenza, la sicurezza e la ricerca dei dati. Spostare un software costruito secondo una visione *cloud-agnostic* tra piattaforme cloud e non cloud non richiede quindi nessuna rielaborazione.

d. Sicurezza

Property	Cloud-aware	Cloud-Agnostic
Multi-tenancy isolation	worse	worse
Maturity and stability of virtualization platform	worse	worse
Encryption of data-at-rest	same	same
Ability to choose legal jurisdiction of hosted data	worse	worse

Tabella 5.4: Risultati delle proprietà per la Sicurezza

A grandi linee la sicurezza nel cloud ricade in due grandi aree: quella relativa alle garanzie di fiducia che un cliente richiede al provider cloud e quella relativa alla sicurezza stessa del software, sia a livello di infrastruttura che di applicazione. Nel primo caso l'obiettivo è di assicurare la confidenzialità dei dati del cliente, nel secondo la sicurezza dell'intero sistema. Per quanto riguarda la fiducia del cliente sulla piattaforma, possiamo dire che questo è un parametro facilmente comparabile tra i tradizionali datacenter e il cloud. Nel primo caso i dati sono mantenuti nelle strutture aziendali del cliente e quindi egli ne è sempre in possesso: la fiducia è implicita. La situazione nel cloud è radicalmente diversa. I provider cloud devono assicurare che i dati del cliente siano sempre protetti da un accesso non autorizzato, come anche dalla loro distruzione. C'è sicuramente più lavoro da fare rispetto ai meccanismi di criptaggio ed autenticazione. Leggi e norme del paese giocano a questo punto un importante ruolo. In molte nazioni esse danno diritto alle agenzie governative di avere accesso ai dati senza un esplicito consenso del proprietario, ma Poiché i provider cloud solitamente ospitano dati e servizi in diverse aree geografiche, il problema della loro sicurezza diventa molto complicato. La sicurezza della piattaforma software presenta altri problemi. Il cloud ha essenzialmente introdotto due entità chiave nell'ecosistema dei datacenter: la

virtualizzazione delle risorse e il concetto di *multi-tenancy* a livello hardware (ossia la condivisione di più macchine virtuali sullo stesso hardware). In un ambiente come il cloud, l'isolamento tra i vari utenti dipende dalle capacità e dalla robustezza del livello di virtualizzazione. Potenziali vulnerabilità nel software di virtualizzazione aumentano l'area attaccabile sulla piattaforma cloud. Nel cloud pubblico, per esempio, bisogna preoccuparsi di mantenere sia l'*hypervisor* sia il sistema operativo dell'utente che gira sulla macchina virtuale in un ambiente sicuro. In un tradizionale datacenter l'unico problema è mantenere il sistema operativo ben aggiornato per evitare problemi di sicurezza.

e. Scalabilità

Property	**Cloud-aware**	**Cloud-Agnostic**
Horizontal scaling	better	same
Vertical scaling	better	better
Ability to exploit NoSQL based data stores	better	same
Data replication	better	better
Inter-tier I/O strategy (collocated vs. distributed tiers)	better	better

Tabella 5.5: Risultati delle proprietà per la Scalabilità

La scalabilità di un' applicazione è prima di tutto determinata dalla sua progettazione. La massiccia scalabilità permessa dal cloud è attribuita a determinate scelte di progettazione sull'architettura del software. Solamente una specifica classe di applicazioni può essere estremamente scalata sul cloud. Tale proprietà deriva in gran parte da:

- utilizzo di transazione BASE piuttosto che ACID[1]
- utilizzo di modelli non condivisi per stati e dati
- sfruttamento del parallelismo presente nella logica dell'applicazione.

Per essere totalmente scalabile la logica dell'applicazione necessita di essere adatta allo sfruttamento di almeno una delle proprietà elencate precedentemente. In quanto tale non c'è nulla nel cloud stesso che renda un'applicazione più scalabile di come sarebbe su una piattaforma tradizionale. Ovviamente la flessbilità fornita dal cloud attraverso la virtualizzazione rende più facile l'aggiunta o la rimozione dinamica delle risorse in risposta alle necessità di scalabilità dell'applicazione.

[1]ACID è l'acronimo di Atomicity, Consistency, Isolation, Durability ed indicano le proprietà logiche che devono avere le transazioni

Capitolo 6

I costi di progettazione

Lo spostarsi da una tradizionale infrastruttura IT per adottare le nuove tecnologie messe a disposizione dal cloud computing fa sorgere due importanti questioni. La prima riguarda il costo richiesto per un tale cambiamento, la seconda la qualità dei servizi forniti.

La stima dei costi nel cloud computing è più facile da determinare rispetto alle soluzioni a sviluppo standard. Infatti, nel primo caso, tutto dipende prima di tutto dall'utilizzo dei tre tipi di modelli, SaaS, PaaS e IaaS, e il consumo effettivo è calcolato sulla base delle risorse impiegate o sul numero di utenti all'ora. Stimare il costo di un software standard è invece una procedura assai più complessa, a causa della grande varietà di spese del personale, del prodotto, dei processi e hardware.

Per quanto riguarda la categoria SaaS, il costo da sostenere è tipicamente limitato alla quota di sottoscrizione. Questo modello fornisce un investimento predicibile che segue uno schema di pagamento del tipo *pay-per-use*. Costi relativi a PaaS e IaaS sono invece solitamente interconnessi. I primi includono utilizzo dei sistemi operativi, dei sistemi di controllo dei database, dei server di web hosting, del software di *batch processing*, degli ambienti di sviluppo delle applicazioni e vengono calcolati sulla base delle richieste ricevute on-demand all'ora. I secondi includono invece utilizzo della CPU, della memoria, dell'hard disk ed in questo caso può essere stipulato un accordo annuale sulle

risorse richieste e sul livello di utilizzo delle componenti hardware.

6.1 Le infrastrutture IT

Quando si effettua la stima dei costi relativi a sviluppo e manutenzione del software è importante considerare anche quelli relativi all'infrastruttura IT. Ciò che li influenza sono gli attributi operativi e le considerazioni iniziali sul business.

I primi si riferiscono ai costi hardware, software e alle tariffe delle licenze. Per quanto riguarda l'hardware bisogna includere l'acquisto di nuove risorse, che a sua volta dipende dal costo di server, PC, periferiche, ecc., e la manutenzione e/o sostituzione di quelle esistenti. Le spese relative a quest'ultima vengono solitamente calcolate attraverso misure che tengono conto del *Mean Time Between Failures* (MTBF), ossia il valore atteso del tempo che intercorre tra un guasto ed il successivo. Le tariffe delle licenze software, dei sistemi e dei database si riferiscono a ciò che viene installato nei computer affinchè essi possano essere pronti ad accogliere la nuova applicazione. Infine il costo delle licenze è definito dal numero di postazioni nelle quali verrà installato il nuovo programma. Il numero di utenti solitamente influisce sui costi delle licenze software richieste e sui costi di reclutamento e formazione.

Tantissimi fattori di performance sono associati con i requisiti non funzionali di un'applicazione che , oltre al fatto che saranno necessariamente incorporati nel software, faranno nascere anche delle considerazioni iniziali sul business, come per esempio la velocità media di transazione, lo storage richiesto, i problemi di sicurezza e fattori di affidabilità.

6.2 Lo sviluppo del software

I costi di sviluppo del software sono divisi in quattro gruppi: quelli relativi al prodotto, alla piattaforma, al processo e al personale.

I costi del prodotto si riferiscono al progetto del software, includendo tipo dell'applicazione sviluppata, tipologia e numero di utenti a cui essa è dedicata e grandezza. I requisiti non funzionali possono alterare invece i valori dei costi relativi alla piattaforma, in modo tale da porre vincoli o portare addirittura a conflitti d'interesse; esempi di questa categoria sono l'affidabilità del programma e la sicurezza. Per quanto riguarda i costi relativi al processo bisogna prendere in considerazione tutti quei supplementi che possono essere usati nello sviluppo e nella consegna di un software di qualità rimanendo nel monte spesa e nelle tempistiche. Per l'ultima categoria bisogno invece considerare come il costo di un software può dipendere dal personale che ha lavorato su di esso. L'esperienza del team, le capacità degli analisti, la familiarità con il linguaggio di sviluppo sono tutti fattori che vanno quindi ad influire su tale costo.

6.3 Un modello decisionale

Al giorno d'oggi i manager IT si trovano di fronte al problema di selezionare come e dove sviluppare le proprie applicazioni. I loro requisiti determineranno quindi la scelta tra cloud computing, sviluppo in locale, o addirittura una combinazione tra queste.

Per prendere una decisione è possibile introdurre una procedura a tre step.

Valutare costi del software e dello sviluppo dell'infrastruttura

Questa procedura coinvolge la valutazione dei costi delle due soluzioni alternative e si basa semplicemente su una stima, dati alla mano, di quella che potrebbe essere la spesa necessaria per finalizzare il progetto.

Definire le caratteristiche qualitative

Le caratteristiche qualitative sono associate agli obiettivi del business e la maggior parte delle volte sono definite come requisiti non funzionali. Tra i vantaggi del cloud computing troviamo sicuramente le alte performance, infatti in un ambiente cloud i sistemi girano più velocemente a causa dei pochi programmi e processi caricati in memoria. Altri importanti attributi sono la compatibilità, l'interoperabilità e la disponibilità, che permettono all'utente di avere accesso al sistema in qualsiasi momento e in qualsiasi luogo. Per quanto riguarda invece le soluzioni locali, i vantaggi maggiori sono sicuramente l'accessibilità dei dati, la sicurezza e il completo controllo. Quest'ultimo è l'aspetto più importante perché con questi software si ha accesso totale sui dati critici ed è inoltre un beneficio per quelle applicazioni che devono supportare un gran volume di transazioni. Altri vantaggi sono la facilità di integrazione con le risorse software/hardware esistenti e la personalizzazione.

Stimare la domanda degli utenti

La stima della domanda degli utenti è, infine, un aspetto assai importante da non sottovalutare. I modelli a database centralizzato presentano costi ragionevoli per 5-10 utenti, ma tale costo cresce esponenzialmente con il numero dei clienti. I modelli distribuiti risolvono questo problema poiché spostano questi costi sui PC. Nell'effettuare questa stima bisognerebbe sempre tenere a mente quattro tipi possibili di richiesta: quella stagionale, quella temporanea, quella *batch processing* e quella temporale. Prendendo come esempio un e-commerce, la prima indica la richiesta che arriva nei periodi prestabiliti nei quali si registra un aumento della domanda, la seconda indica genericamente periodi di liquidazione delle scorte, la terza indica periodi nei quali può essere svolta intensa attività di pubblicità del sito e la quarta si verifica quando un prodotto riscuote un successo inaspettato.

Conclusioni

Questa tesi ha analizzato la progettazione e lo sviluppo delle applicazioni nella nuova ottica del cloud computing.

Inizialmente è stato analizzato lo sviluppo dei software attuali, troppo spesso pesante e inefficiente a causa dell'esagerato numero di parti da installare e configurare. Per questo motivo viene introdotta la piattaforma cloud: gli sviluppatori potranno ora accedere all'ambiente di sviluppo da qualsiasi postazione in qualsiasi luogo, eliminando inoltre la necessità di un'infrastruttura centralizzata.

Nella nuova modalità di sviluppo assume particolare importanza la fase di raccolta dei requisiti, in modo tale che essi vengano definiti alla perfezione per evitare intoppi nelle fasi successive del processo di progettazione del software. Molto importante è anche l'intervento dei provider cloud in questa fase, poiché essi potranno conoscere approfonditamente i dettagli architetturali della piattaforma sottostante.

Si è poi passati alla descrizione della piattaforma SaaS, individuando le due tipologie principali di software: le applicazioni Business e le applicazioni BPM. Si è potuta notare la grande configurabilità di queste soluzioni, in relazione a struttura organizzativa, interfacce utente, modelli dei dati, flussi di lavoro e logiche di business. Inoltre è importante rilevare come, attraverso l'utilizzo dei metadati, sia possibile realizzare software fruibili da più aziende in modo tale che soddisfino la cosiddetta proprietà di *multi-tenancy*. In questo modo è possibile sfruttare la stessa piattaforma per servire più clienti, attuando un ulteriore risparmio sui costi.

Se da una parte sono enormi i vantaggi, permangono ancora molte problematiche relative alla sicurezza e all'isolamento dei dati. In questo senso la ricerca dovrà sviluppare sempre nuove tecnlogie in modo tale da consentire la loro risoluzione, anche se esistono già contromisure efficaci, come il protocollo SSL o le firme digitali.

Volendo poi scegliere tra uno sviluppo cloud-aware ed uno cloud-agnostic, è per il momento preferibile la seconda soluzione, più che altro per la maggior esperienza con questo approccio. Tuttavia la prima offre sicuramente migliori prestazioni in relazione alla scalabilità, data appunto la sua natura votata al cloud.

Bibliografia

[1] Wikipedia - *http://en.wikipedia.org*

[2] MSDN Library - *http://msdn.microsoft.com*

[3] IEEE Explore - *http://ieeexplore.ieee.org*

[4] Mondo Digitale - *http://www.mondodigitale.net*

[5] Yashpalsinh Jadeja, Kirit Modi, *Cloud Computing - Concepts, Architecture and Challenges*, India, IEEE, 2012

[6] Shyam Patidar, Dhaeeraj Rane, Pritesh Jain, *Challenges of Software Development on Cloud Platform*, India, IEEE, 2011

[7] Sungjoo Kang, Sungwon Kang, Sungijn Hur, *A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform*, Korea, IEEE, 2011

[8] Hong He, *Applications Deployment on the SaaS Platform*, China, IEEE, 2010

[9] Balwinder Sodhi, *Application Architecture Considerations for Cloud Platforms*, India, IEEE, 2011

[10] Stamatia Bibi, Dimitrios Katsaros, Panayiotis Bozanis, *Application Development: Fly to the Clouds or Stay-in-House?*, Greece, IEEE, 2010

Ringraziamenti

Ringrazio la mia Je perché grazie alla sua preziosa vicinanza ho potuto raggiungere questo obiettivo.

Ringrazio la mia famiglia perché mi ha dato la possibilità di poter conseguire questi studi, in ogni modo.

Ringrazio tutti i miei amici, universitari e non, che mi hanno accompagnato e divertito in questi anni.

Ringrazio il mio relatore, Alessandro Ricci, per avermi aiutato nella stesura di questa tesi.