*Matr. n.* 0000356837

***Alma Mater Studiorum - Università di Bologna***

**FACOLTÀ SCIENZE E TECNOLOGIE INFORMATICHE**

CORSO DI LAUREA IN SCIENZE MATEMATICHE, FISICHE E NATURALI

# Community per la condivisione di contenuti multimediali geolocalizzati su piattaforma mobile

TESI DI LAUREA IN MOBILE WEB DESIGN

Relatore

Dott. Mirko Ravaioli

Presentata da

Marco Leali

Sessione II

*Anno Accademico 2011-2012*

# INDICE

# Introduzione

## 1.1 – Storia

Smartphone o telefono intelligente, è un dispositivo portatile con tutte le funzionalità di un telefonino in aggiunta alle caratteristiche di un palmare. Negli ultimi anni l'evoluzione e la commercializzazione di questi dispositivi è stata talmente rapida e importante da cambiare radicalmente le abitudini degli utenti oltre all'approccio nello sviluppo delle applicazioni.

Tralasciando la telefonia nel suo senso più stretto, la prima apparizione degli smartphone possiamo farla risalire al 1992, con l'immissione nel mercato di *Simon* da parte di *IBM*. La storia parte da questo dispositivo poiché si tratta del primo con funzionalità aggiuntive rispetto ai normali telefonini: calendario, notepad, lista degli indirizzi, esportazione di contatti e, infine, una caratteristica comune e ormai obbligatoria per uno smartphone odierno: il touchscreen.

Degni di nota sono i *Nokia* appartenenti alla famiglia *9xxx*, lanciati qualche anno più tardi sul mercato (1996). Il modello *9210* è il primo ad usare un sistema operativo open-source.

Uno dei primi a poter essere chiamato effettivamente smartphone è il *3800* di *Ericsson*, primo anche nell'utilizzo del sistema operativo *Symbian*, sviluppato alla fine degli anni novanta dalla società Psion .

Negli anni successivi si moltiplicano i dispositivi immessi in commercio. Ricordiamo il *Palm Treo*, il *P800* e il primo *Blackberry*, il cui punto di forza consisteva nel rendere facilmente accessibile la ricezione e l'invio di mail tramite le reti senza fili.

La rivoluzione è avvenuta però con l'avvento del 2007 con l'immissione sul mercato del primo *iphone* da parte di Apple e poco dopo con l'uscita sul mercato dei primi terminali Android.

## 1.2 – La Diffusione

In un periodo di crisi globale come questo possiamo affermare che nulla come il settore mobile, stia subendo una crescita così importante come negli ultimi anni. Infatti se parliamo di recessione a livello mondiale per quanto riguarda micro-elettronica, informatica di consumo o informatica a livello di business o enterprise, non possiamo dire la stessa cosa del fenomeno degli smartphone, dei tablet e di tutti i device da loro derivati.

Dispositivi come smartphone o tablet sono diventati di uso comune con una velocità sbalorditiva e questa iperbole non sembra destinata a rallentare. Anzi, secondo dati forniti da *The Week*, nel 2016 la popolazione mondiale si attesterà a 7,3 miliardi, col raggiungimento dell'uso di 10 miliardi di dispositivi mobili, circa 1.4 per abitante.

## 1.3 – Mobile vs Computer

Se un tempo la telefonia si presentava come un mondo distaccato da quello del computer, nel corso del tempo i confini sono diventati sempre più labili. Gli smartphone accrescono sempre di più la loro potenza, includendo ad ogni release nuove funzionalità e "convertono" un numero crescente di utenti. Nel 2011 c'è stata la prima svolta epocale dall'entrata sul mercato dei dispositivi mobile; infatti la vendita di smartphone e tablet ha raggiunto la quota globale di 480 milioni di unità distribuite e vendute, mentre la vendita di PC, notebook, netbook e sistemi enterprise si è attestata a "soli" 350 milioni di unità.

**Worldwide smart phone and client PC shipments**
**Shipments and growth rates by category, Q4 2011 and full year 2011**

| Category | Q4 2011 shipments (millions) | Growth Q4'11/Q4'10 | Full year 2011 shipments (millions) | Growth 2011/2010 |
|---|---|---|---|---|
| Smart phones | 158.5 | 56.6% | 487.7 | 62.7% |
| Total client PCs | 120.2 | 16.3% | 414.6 | 14.8% |
| - Pads | 26.5 | 186.2% | 63.2 | 274.2% |
| - Netbooks | 6.7 | -32.4% | 29.4 | -25.3% |
| - Notebooks | 57.9 | 7.3% | 209.6 | 7.5% |
| - Desktops | 29.1 | -3.6% | 112.4 | 2.3% |

Source: Canalys estimates © Canalys 2012

*Confronto di vendita dei tipi di dispositivi 2011*

## 1.3.1 – La Connettività

Oltre all'evoluzione hardware, ciò che ha portato all'incredibile crescita del numero di utilizzi è dato dal miglioramento della connettività. Stiamo assistendo infatti alla continua crescita di reti senza fili come wireless oppure le più recenti tecnologie di connessione dati come il 2G, il 3G e il 4G o LTE (*Long Term Evolution*).

Grazie a questo, il numero di utenti che accede giornalmente alla rete non dalla propria postazione fissa o notebook è in costante aumento. Poter aprire le nostre pagine preferite, recuperare un documento importante o leggere le ultime mail ricevute ovunque ci troviamo, non può che suscitare la curiosità e l'attenzione di tutti gli utenti.

Tutto ciò è dimostrato dal più conosciuto e utilizzato social network del mondo: Facebook.

Facebook al raggiunto, all'inizio del terzo trimestre del 2012, il record di un miliardo di utenti registrati e di questi, più di 600 milioni sono utenti attivi. Il 40% proviene da piattaforme mobili e la percentuale è destinata a salire. Infatti, rapportando i dati di acceso ai vari social network del 2011 con quelli dell'anno precedente, la crescita è enorme.

Anche nell'ambito business ed enterprise il mobile ha preso piede con successo; In questo campo infatti sono necessari accessi ai dati veloci e sicuri, ma soprattutto ogni volta che se ne ha il bisogno, da qualsiasi luogo.

### 1.3.2 – Le Applicazioni

L'esplosione delle applicazioni per il mobile permette di compiere gran parte del proprio lavoro normalmente svolto su un computer desktop o portatile, tramite il proprio dispositivo.

Centinaia di migliaia di programmi sono già presenti per le diverse piattaforme e il loro numero continua a crescere. Il solo Apple Store ha raggiunto, nei primi mesi del 2012, i 25 miliardi di download. Sebbene non sempre i programmi disponibili siano all'altezza della loro controparte desktop, bisogna considerarne il rapido incremento in numeri per notare che ci saranno sempre più applicazioni non presenti invece nel mercato desktop con il proseguire del tempo. Viceversa ci sono una miriade di applicazioni che possono essere utilizzate solo su piattaforme mobile, grazie alla presenza di connessioni o sensori presenti solo su tablet o smartphone, piuttosto che notebook o desktop tradizionali; l'esempio più semplice e lampante è la presenza di GPS (e/o GLONASS per gli stati asiatici), accellerometro e giroscopio in tutti i dispositivi di ultima generazione.

### 1.3.3 – Specifiche tecniche

Uno dei punti in cui il settore mobile fa ancora fatica a combattere contro gli apparecchi tradizionali è la potenza in termini di velocità, di esecuzione e frequenza dei componenti. Il mobile deve affrontare il gap tecnologico prodotto dalla grande differenza di anni nella nascita delle due tecnologie. Incredibile è quindi constatare come le specifiche hardware degli

smartphone stiano, pian piano, raggiungendo la potenza delle nostre postazioni fisse.

E' difficile poter stilare delle specifiche hardware precise per quanto riguarda questo settore. Le componenti sono in continuo mutamento e miglioramento ed il tasso di crescita è altissimo.

Possiamo però guardare a quali sono stati i progressi nel corso di così poco tempo.

Il *Nokia 9000* presentava una memoria interna di 8MB, di cui solamente 2 destinati ad essere utilizzati dall'utente. La cpu era una *Intel 386*. Questo nel 1996.

Facciamo un salto in avanti fino al 2005, con l'introduzione da parte di Nokia della serie N. Prendendo in considerazione il modello *N70*, possiamo vedere un aumento delle specifiche inferiore a quello che ci si potrebbe aspettare: 64MB disponibili con la scheda di memoria esterna e 22MB interni, coadiuvati da un processore a 220 MHz

A partire dal 2007 la curva di evoluzione si impenna. Il primo *iPhone* rilasciato presenta un processore ARM con frequenza di 620 MHz e una memoria base di 16GB. Un salto colossale rispetto all'N70 di due soli anni prima.

Nel 2010 *LG* annuncia Optimus 2X, il primo smartphone con una CPU *dual-core*. Si tratta del chip *Tegra 2* prodotto da *Nvidia*.

L'anno dopo troviamo il *Samsung Galaxy SII*, che presenta 1GB di ram e una memoria interna di 32GB, oltre ad un processore dual-core da 1.2 GHz. Al momento della scrittura uno dei terminali più promettenti è il Nokia Lumia 920 che debutterà fra pochi giorni e le specifiche tecniche annoverano un processore ARM dual-core Snapdragon S4 alla frequenza di 1.5 Ghz, 2GB di RAM LPDDR2 (Low Power Double Data Rate) e 32 GB di memoria interna.

La differenza consiste nella velocità con la quale si è raggiunto lo stato attuale. L'aggiornamento dell'hardware degli smartphone ha subito un'accelerazione notevole, tanto da rendere possibile l'esecuzione di molte delle applicazioni prima riservate ai PC.

### 1.3.4 – La Sicurezza

Non meno importante è la parte relativa alla sicurezza. Le più grandi minacce per gli utenti PC sono virus e *Trojan*, capaci di compromettere l'integrità dei nostri computer e rubare dati sensibili. I sistemi operativi mobile condividono gran parte del codice dei sistemi operativi tradizionali e questo mette gli hacker di tutto il mondo nella posizione di non doversi re-inventare codici malevoli da sfruttare su dispositivi mobili. Oltre a questo si aggiunge il fatto che i device odierni possiedono molte connessioni che fino a pochi anni orsono si trovavano a fatica su qualsiasi tipo di dispositivo, sto parlando di wireless, 2G, 3G, H3G, 4G o LTE, NFC (Near Field Communicator), Bluetooh e IrDA. Nell'ultimo periodo sono stati scoperti da note società di sicurezza, exploit creati appositamente che permettevano agli hacker di entrare in possesso delle informazioni sensibili presenti nei terminali degli utenti. Ulteriori vettori che permettono il diffondersi di codice malevolo, sono le autorizzazioni e i privilegi che gli utenti mettono a disposizione delle varie applicazioni che vengono scaricate dagli store (Google Play, Micorosft Store o App Hub e infine App Store); in questo modo il malware presente nell'applicazione scaricata può indisturbatamente entrare in possesso dei dati sensibili dell'utente ed esserne a completa disposizione. E' per questo motivo che negli ultimi due anni c'è stato un boom di applicazioni sviluppate ad hoc, che sono entrate a far parte dei cosiddetti *antivirus-mobile.*

### 1.3.5 – Sommario

Il mobile sta avendo un ruolo chiave nell'elettronica e nell'informatica ed è un ambito ancora in enorme crescita.

Nessuno, al momento, sa quali possano essere gli sviluppi futuri di un settore che evolve più rapidamente di quanto l'industria possa attualmente sostenere.

## 1.4 – Sistemi Operativi

Il sistema operativo è il software che gestisce ogni singolo aspetto dei moderni device, cioè quello "strato" che si occupa di mettere in comunicazione l'apparato hardware del dispositivo (processore, ram, supporto di archiviazione, sensori) e l'interfaccia utente.

Gli odierni sistemi operativi hanno abbondantemente superato le funzionalità che solo pochi anni fa venivano offerte dalle controparti desktop. Probabilmente questo è dovuto al fatto che i moderni dispositivi mobile dispongono anche di sensori che nel passato erano inimmaginabili per i computer fissi, ma probabilmente anche poco utili.

Oggigiorno le più grandi software house del mondo (e non solo) su buttano a capofitto nella realizzazione di sistemi operativi proprietari nella speranza di riuscire a trovare un piccolo spazio nella nuvola di sistemi operativi presenti.

Di seguito sarà fatto un breve elenco dei sistemi operativi attualmente in commercio o in fase di realizzazione, per poi spiegarli più dettagliatamente in seguito:

- Android, Google
- iOS, Apple
- Windows Phone, Microsoft
- Windows Mobile, Microsoft
- Blackberry, Research In Motion (RIM)
- Symbian, attualmente di proprietà di Nokia
- WebOS, sistema operativo open-source la cui realizzazione è stata portata avanti da HP e dalla comunità mondiale di sviluppatori
- BADA, Samsung
- Tizen, Limo Foundation
- MeeGo, Nokia e Intel
- Firefox OS, Mozilla Foundation
- OS Huawei, Huawei

Come potete ben osservare i sistemi operativi in ambito mobile sono i più svariati. Di questi però, solo Apple e Google detengono più dell'80% del mercato; questo infatti va ad influire ancora di più sulla popolarità dei vari software, con il risultato che gli altri sistemi operativi perdono costantemente quote di mercato, avvicinandosi sempre di più verso il declino. Da questo discorso pare chiaro che entrare nel 2012 in un mercato già saturo di sistemi operativi è pressoché impossibile, rendendo vani gli sforzi da parte dei produttori di software. L'unico sistema operativo che è in costante crescita, anche se molto bassa, è Windows Phone di Microsoft. L'SO di Microsoft infatti detiene circa il 5% del mercato nonostante sia uscito nel 2010 inoltrato, in un momento in cui Android e iOS stavano letteralmente conquistando il mercato e gli utenti. In alcuni paesi inoltre ha quasi raggiunto la quota dell'11% raggiungendo il rivale Symbian ed andando ad assediare il terzo posto nel market share, a distanza di meno di un punto percentuale

Nel grafico proposto infatti si può notare il market share di Italia e unione europea, confrontato tra il 2011 (a sinistra) e il 2012 (a destra).

| | | | |
|---|---|---|---|
| Italy | 100.0% | 100.0% | 0.0 |
| iOS | 18.4 | 14.7 | -3.7 |
| Android | 37.2 | 58.7 | 21.5 |
| RIM | 4.8 | 3.5 | -1.3 |
| Symbian | 32.6 | 11.0 | -21.6 |
| Windows | 3.8 | 10.4 | 6.6 |
| Bada | 2.4 | 1.8 | -0.6 |
| Other | 0.8 | 0.0 | -0.8 |
| EU5 | 100.0% | 100.0% | 0.0 |
| iOS | 18.3 | 14.0 | -4.3 |
| Android | 48.6 | 68.5 | 19.8 |
| RIM | 12.2 | 6.4 | -5.8 |
| Symbian | 13.3 | 3.7 | -9.6 |
| Windows | 3.8 | 5.0 | 1.2 |
| Bada | 3.4 | 1.9 | -1.4 |
| Other | 0.5 | 0.6 | 0.1 |

*Market share in Italia e in Europa – Confronto 2011/2012*

## 1.5 – Differenze

Per capire meglio lo stato attuale della panoramica mobile, verranno confrontati i vari sistemi operativi.

Android: il software di Google è attualmente il sistema operativo più diffuso sul mercato mobile, questo infatti, detiene poco più del 50% del mercato.

Il progetto è stato portato avanti dall'associazione Open Handset Alliance, un gruppo formato da 84 gruppi del mondo dell'informatica e dell'elettronica;

Ha fatto la sua apparizione sul mercato la prima volta a fine del 2008 e il suo fine è molto ambizioso, si tratta infatti di un sistema operativo open-source basato sul kernel di Linux. Ciò implica che uno sviluppatore ha la completa libertà di prendere il codice sorgente di Android e modificarlo a proprio piacimento; succede sempre più spesso infatti che i produttori di smartphone non aggiornino in maniera costante i propri dispositivi, perciò la comunità di sviluppatori indipendenti si adoperano per modificare i kernel personalizzati dalle grandi case per portare gli aggiornamenti più recenti sui propri dispositivi.

iOS: è il sistema operativo sviluppato da Apple nato nel 2007, utilizzato sui dispositivi iPhone, iPad e iPod Touch, che si basa su UNIX.

Windows Phone: è stato sviluppato da Microsoft, basa le proprie API su quelle di Win32 e il proprio kernel su Windows CE, appositamente sviluppato per device mobili. Il sistema operativo di Microsoft si può considerare competitivo dal 2010 quando è uscita la versione 7.5 del software, chiamata anche Mango; questa porta moltissime migliorie grazie ad un codice riscritto in tutte le sue parti tranne che nel suo nucleo fondamentale che ancora eredita dalla versione del 2003: il kernel.

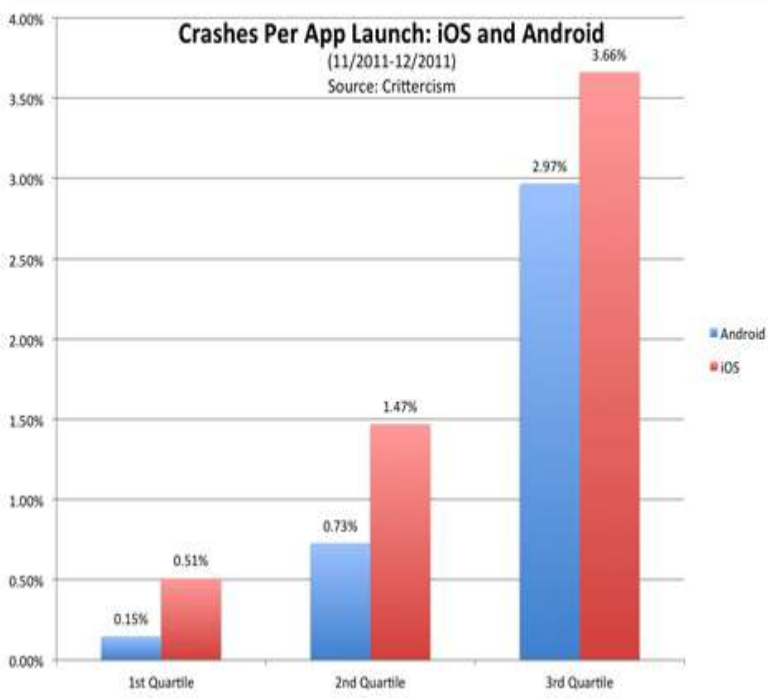
Symbian: è il sistema operativo di casa Nokia, che lo rilevò nel 2008 dalla Symbian Foundation. Fino a quel momento Symbian ebbe una diffusione capillare, dovuta soprattutto all'egemonia di Nokia nel campo dei cellulari. Nonostante sia tuttora un ottimo sistema operativo mobile, Symbian, ha dovuto lasciare il passo ad altri SO quali Android e iOS e infatti le sue quote di mercato si stanno affievolendo sempre di più, infatti contando tutti i dispositivi mobili, Symbian è passato da un market share del 70% del 2007 a circa il 10% attuale. Probabilmente tutto ciò non è dovuto al sistema operativo in se, ma alla scarsa innovazione da parte di Nokia nel mondo mobile. Infatti fino al 2008 non c'è mai stato un cambio radicale in ambito di design, in quanto cellulari e gli smartphone semplicemente vendevano, ma l'entrata sul mercato dell'iPhone di Apple ha cambiato le carte in tavola e Nokia è dovuta correre ai ripari anche se, complice una gestione errata della società e un piano produttivo poco chiaro, non ha portato i frutti che tutti si aspettavano.

I numeri parlano chiaro: con circa 70 mila nuove attivazioni giornaliere Android regna padrone su tutti il mercato mobile, dagli smartphone di fascia bassa, fino ai tablet ultra-costosi. Questo è dovuto ad una serie di fattori, tra cui appunto la diffusione di Android anche in terminali che partono, sul mercato occidentale, dai 50 - 60 euro, oltre alle offerte dei vari operatori telefonici che con una spesa minima mensile permettono di usufruire di un terminale di ultima generazione e il più delle volte adotta, come sistema operativo appunto, Android.

Inoltre circa 650 mila applicazioni, delle quali quasi il 70% è gratis, rendono il sistema Android più appetibile per gli utenti. Non sono da sottovalutare inoltre i tempi di apprendimento della piattaforma e quelli di sviluppo della propria applicazione che risultano essere minori rispetto alle controparti. Nonostante questo molti sviluppatori hanno deciso di spostarsi o di avvicinarsi sempre di più al mondo Apple piuttosto che alle realtà di Google o Microsoft; ciò è dovuto al fatto che, in

percentuale, il guadagno in un'applicazione iOS è nettamente maggiore rispetto agli altri.

Windows Phone  si sta distinguendo a livello mondiale per la sua straordinaria stabilità. Lo stesso discorso non si può fare per Android, a causa della sua frammentazione di release mentre, per iOS, la situazione pare sia ancora peggio.




*Percentuale di crash delle applicazioni per Android e iOS*

Android non presenta un supporto nativo al Cloud, se non tramite applicazioni di terze parti come *DropBox*. Anche il supporto fornito a Windows Phone grazie all'App integrata *SkyDrive*, non regge il confronto con l'implementazione scelta da Apple.

Se, nel caso di Windows Phone e Android mettono a disposizione un *file loker*, cioè semplicemente un luogo in cui archiviare i propri file per poi potervi accedere in un secondo momento, *iCloud* di Apple invece da la possibilità di sincronizzare tutti i dati, comprese quelle di terze parti, che vengono sincronizzati tra il dispositivo e i computer connessi. Il tutto in modo semplice e senza nessuna fatica da parte dell'utente.

# Progettazione

## 2.1 – Descrizione del progetto

Il progetto realizzato e presentato in questa tesi prende il nome di: TakePlace.

Ogni giorno nascono e muoiono social network che cercano di sfondare nel web.

L'idea sviluppata è una concezione diversa dal solito social network. L'applicazione si basa su due funzioni principali, ovvero: la possibilità di eseguire l'upload delle proprie foto in una community oppure eseguire l'upload delle proprie foto in maniera che sia solo l'utente che in seguito le possa rivedere.

Per via di questo ultimo punto, l'applicazione si può definire, più che social, un personal network.

Alla base dell'applicazione c'è la possibilità di condivisione dei propri dati multimediali con tutti gli utenti che dispongono dell'applicazione. Gli utenti infatti possono eseguire l'upload delle proprie foto o dei propri video in maniera geolocalizzata, cioè viene messo a disposizione un servizio di mappe, quello di Microsoft Bing, in maniera che foto o video siano legati indissolubilmente ad un luogo.

Man mano che l'utente accederà al dettaglio della mappa, verranno aggiunti dei segnaposto (chiamati Pushpin) che andranno ad indicare che un utente in quel punto ha scattato una foto oppure ha girato un video e ne ha eseguito l'upload. I Pushpin vengono aggiunti dinamicamente ogni volta che viene aumentato di un ordine lo zoom sulla mappa, viceversa eseguendo lo zoom indietro i Pushpin saranno sempre più rarefatti, questo per ottimizzare la memoria.

La seconda funzionalità principale dell'applicazione è data dalla possibilità di poter eseguire l'upload di dati multimediali, foto o video, ma in questo caso è solo colui che ha eseguito l'upload che potrà rivedere in futuro i propri dati. Se vogliamo vederla in un certo modo, con questa funzionalità l'utente potrà avere un proprio archivio personale di foto e video e sapere esattamente anche dove sono stati registrati.

A causa di alcune limitazioni imposte dal sistema operativo (o dalle scelte di Microsoft) non è possibile interagire con alcune parti di esso ed integrare le funzionalità sviluppate da programmatori di terze parti con le applicazioni di base di Windows Phone. Nel caso ti TakePlace ci si riferisce alla possibilità di integrare l'applicazione con quella della fotocamera integrata in Windows Phone. Sarebbe stato l'ideale infatti, se l'utente avesse avuto la possibilità di scegliere se eseguire l'upload di una foto o un video registrati con il software di base messo a disposizione dal Microsoft. Per questo motivo ho dovuto invece integrare nella mia applicazione, un modulo che permettesse di scattare foto o girare video. Questa offre le funzionalità di base che offre anche la fotocamera di Windows Phone. E' un servizio che ho deciso di integrare nell'applicazione, in quanto permette all'utente di eseguire le operazioni da un'unica applicazione; l'altra strada sarebbe stata quella di far scegliere all'utente i propri media da caricare solamente dalla galleria.

L'upload è possibile anche con i file multimediali catturati precedentemente. L'utente può scegliere di caricare una foto o un video dalla propria galleria o dalla propria scheda di memoria. La prima nota negativa di questa funzionalità che viene in mente è che l'utente possa eseguire l'upload di foto in luoghi che non c'entrano assolutamente con quello che è ritratto nella foto o nel video.

L'idea di questa applicazione è venuta in quanto lo Store di Windows Phone è ancora relativamente acerbo rispetto ai concorrenti Android e iOS, ma con l'avvento del nuovo sistema operativo di Microsoft, Windows 8 e quindi

Windows Phone 8, le cose dovrebbero prendere un'altra piega. Inoltre facendo un giro sullo Store di Windows Phone 7.5, non si trovano applicazioni simili a questa ma soprattutto non si trovano alcuni grandi nomi quali Instagram o FourSquare, che permettono, nei sistemi operativi concorrenti, di eseguire il chek-in nei luoghi in cui ci si trova oppure di inserire un tag di geolocalizzazione all'interno delle foto scattate; per questo motivo TakePlace si prende l'incarico molto ambizioso di cercare di coprire quella mancanza che, a quanto pare, si sente.

## 2.2 – Diagramma delle classi

Per lo sviluppo dell'applicazione sono state utilizzate alcune peculiarità del formalismo UML. Questo permette, prima di buttarsi sulla scrittura del codice vero e proprio, di capire come dovranno essere gestite le classi e l'interazione fra loro. Anche per un progetto relativamente piccolo come questo UML è stato molto utile per comprendere, in una prima fase di analisi, quali sarebbero stati i problemi che avrei dovuto affrontare.

Grazie ad un'analisi iniziale sono stati risolti preventivamente alcuni problemi che, se si fossero presentati in uno stadio più avanzato del progetto, avrebbero richiesto una ristrutturazione completa del codice.

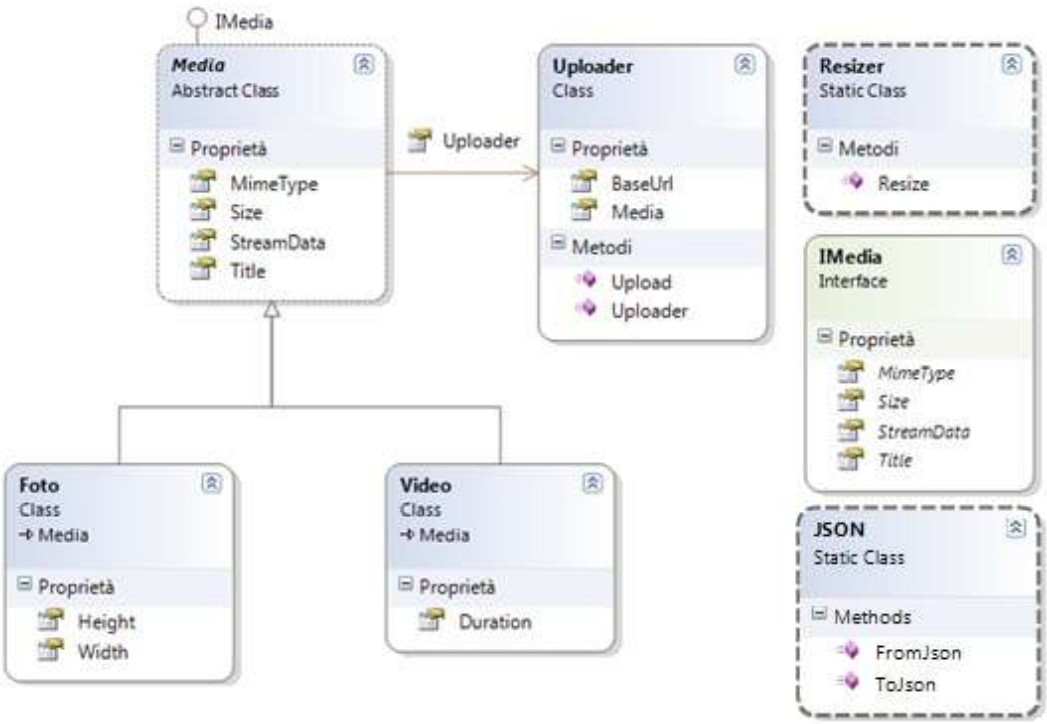



*Diagramma delle classi del progetto*

Come si può osservare il diagramma delle classi è relativamente semplice. Si è deciso di utilizzare un, sottovalutato ma molto utile, Design Pattern: il Dependency Injection.

La classe pubblica Uploader infatti espone un interfaccia di tipo Imedia, che viene inizializzata quando viene richiamato il costruttore della classe Uploader, a cui viene passato un parametro di tipo IMedia, più precisamente una classe che implementa l'interfaccia IMedia. In questo modo ho dovuto creare un solo metodo che utilizza il tipo IMedia che si adatta al tipo di classe che viene passato come parametro in quel momento. Ho dovuto differenziare gli oggetti foto e video perché l'upload viene gestito in modo leggermente diverso.

Prima di eseguire l'upload degli oggetti multimediali sul server si deve eseguire un'operazione di ridimensionamento, esposta dalla classe statica Resizer. Questo perché per lo sviluppo dell'applicazione si è utilizzato un server privato, quindi non a pagamento, perciò si deve limitare in ogni modo

l’utilizzo di risorse, che in questo caso si traducono in risparmio di banda, risparmio di spazio occupato sull’hard disk del server e di conseguenza un risparmio sull’utilizzo del processore.

Nel caso si debba eseguire l’upload di una foto infatti si esegue un ridimensionamento ad un massimo di 2048 x 2048 pixel e se non è già in formato jpeg, viene convertito nel suddetto formato siccome è quello che permette, mantenendo un buon livello di qualità, una dimensione e un utilizzo di spazio su disco contenuto. Nel caso si trattasse di un video invece, si esegue prima un controllo, per verificare il formato, il codec utilizzato, il livello di compressione, la risoluzione di cattura e infine la dimensione su disco. Per ognuno di questi parametri si esegue, nel caso non soddisfi i miei requisiti, una conversione del video in formato H.264 e dell’audio in formato AAC che, come nel caso delle foto, mantengono una buona qualità riducendo però drasticamente lo spazio occupato per avere maggiore velocità e quindi migliori prestazioni quando si dovrà inviare al server tutte le informazioni.

## 2.3 – Diagramma dei casi d’uso

Un secondo diagramma del formalismo UML che è stato utilizzato è quello dei casi d’uso. Questo tipo di diagramma ha permesso di definire in maniera preventiva le operazione che l’utente avrebbe potuto svolgere.

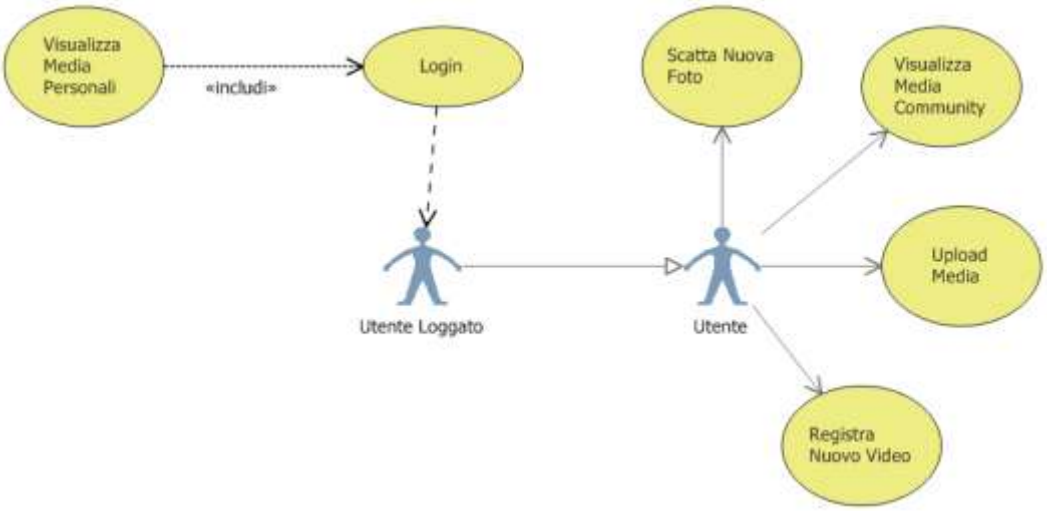



*Diagramma dei casi d'uso del progetto*

Come si può osservare dall'immagine, le attività che un utente può svolgere, oltre alla modifica delle impostazioni, sono principalmente cinque.

- Scatta nuova foto: l'utente può decidere di utilizzare il servizio integrato in TakePlace per scattare una foto che verrà salvata automaticamente nella galleria personale.
- Visualizza Media Community: l'utente può visualizzare la mappa di Bing, in cui appaiono i Pushpin in corrispondenza del luogo in cui è stato caricato da parte di un utente, una risorsa multimediale. All'aumentare dello zoom della mappa crescerà il numero di Pushpin, ad indicare che ci sono più media visualizzabili.
- Upload Media: con questa funzionalità l'utente può eseguire l'upload di un'immagine o di un video presenti sul proprio dispositivo direttamente sul server, per vedere il tutto in un secondo momento.
- Registra Nuovo Video: come per le foto, l'utente può utilizzare il servizio messo a disposizione da TakePlace per poter girare un filmato che verrà salvato automaticamente nella galleria personale.
- Visualizza Media Personali: questa funzionalità permette all'utente di visualizzare i media personali caricati precedentemente ed è un servizio riservato ai soli utenti registrati. I possibili modi per tenere traccia di chi esegue il caricamento di una risorsa erano diversi;

infatti si sarebbe potuto salvare il numero di telefono dell'utente oppure l'identificativo IMEI. Nel caso in cui l'utente avesse cambiato SIM oppure avesse comprato un telefono nuovo, queste informazioni sarebbero andate perdute. La scelta del login, è quella più scomoda per l'utente, ma è anche la più sicura e globale, infatti se ipoteticamente venisse sviluppato TakePlace anche per Android o iOS, con l'accesso tramite credenziali, l'utente ritroverebbe i contenuti caricati da un altro

Tutte le attività che possono essere svolte da un utente non registrato, posso essere eseguite anche da un utente che ha eseguito il login.

## 2.4 – Logica progettuale

La logica progettuale dell'applicazione può essere suddivisa in due parti ben distinte: il lato client e quello server.

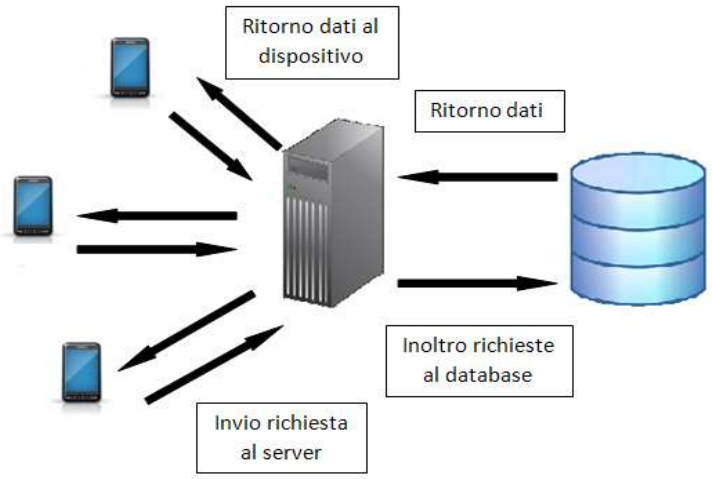
Il server si occupa di tenere memorizzati gli utenti, le foto, i video caricati e le relative posizioni.

Il client, in questo caso smartphone o tablet, si occupa di visualizzare all'utente tutte le informazioni che vengono ricevute dal server.

A causa di alcuni limitazioni dovute al sistema operativo Microsoft Windows Phone 7.5 non è possibile affidarsi alla persistenza dei dati tramite database anche sul device; sarebbe stato possibile immagazzinare alcune informazioni volta per volta, invece così si rende necessario, ad ogni modifica eseguire nuovamente una richiesta di dati al server.

## 2.4.1 – Lato server

Attraverso un sistema sviluppato appositamente, disponibile come servizio web, l'applicazione sviluppata ottiene le informazioni aggiornate in merito agli utenti che nelle vicinanze hanno effettuato l'upload delle proprie immagini o video. Infatti quando un dispositivo invia una richiesta al server remoto, quest'ultimo apre una nuova connessione al database, compie una specifica query e ritorna i dati necessari allo smartphone.




*Come avviene lo scambio di dati tra i dispositivi client e il server*

Principalmente i tipi di richieste che il client può fare al server sono tre, cioè

richiedere l'oggetto che contiene tutti i dati per visualizzare correttamente la foto o il video. L'oggetto che viene restituito dal server contiene 3 tipi di informazioni:

ID: l'identificativo della risorsa da visualizzare, questa proprietà sarò utile nel caso l'utente volesse visualizzare a schermo intero l'immagine o il video

URL: l'indirizzo che indica dove è stata salvata la risorsa e dal quale è possibile andarla a recuperare per poterla visualizzare sullo schermo dello smartphone. Questa proprietà indica l'indirizzo in cui è stata salvata la

risorsa ridimensionata a livello del server, cioè che si tratti di una foto o un video, il server restituirà l'indirizzo di una miniatura della risorsa per non doverla ridimensionare nel momento della visualizzazione perché sarebbe solo uno spreco di risorse.

Coordinate: la posizione che è stata specificata per salvare la risorsa, è formata dalla longitudine e dalla latitudine.

Gli altri due tipi di richiesta che il client può fare al server, sono quelle che si hanno nel momento del login o della registrazione, cioè nel primo caso il client invia al server le informazioni dell'utente per verificare che siano state immesse correttamente, nel secondo caso il client invia ancora tutte le informazioni immesse dall'utente, ma questa volta per verificare che siano state specificate correttamente e infine che non esista un utente che si sia già registrato con credenziali uguali.

**Il Database Remoto**

Il Database remoto si occupa di mantenere la persistenza dei dati tra tutti gli utenti, cosa che il client, cioè il device non può fare.

| MEDIA | |
|---|---|
| IdMedia | Int |
| Title | varchar |
| Url | varchar |
| UrlThumbnail | varchar |
| Latitude | float |
| Longitude | float |

| | |
|---|---|
| Type | boolean |
| Username | varchar |
| Visible | boolean |

Verrà data una spiegazione di come è stata progettata la tabella.

- IdMedia: è l'identificativo della risorsa che è stata salvata sul disco rigido ed è un campo auto-incrementate
- Title: è il titolo che è stato dato all'immagine dall'utente nel momento del salvataggio
- Url: questa è il campo che specifica in quale locazione è stato salvato il file e da dove è possibile andarlo a recuperare per visualizzarlo
- UrlThumbnail: questo invece è il campo che indica dove si trova la risorsa che è stata ridimensionata
- Latitude: indica la posizione latitudinale della risorsa
- Logitude: indica la posizione longitudinale della risorsa
- Type: campo che indica se la risorsa è una foto o un video, essendo un campo booleano, 1 se la risorsa è una foto, 0 viceversa
- Username: è il campo sul quale è stata applicata la Foreign Key, cioè la chiave esterna che va a legare una specifica risorsa ad un proprietario, nel qual caso ci fosse. Ho deciso di utilizzare lo username e non l'Id come chiave esterna perché così non ci possono essere Username duplicati. Questo campo ovviamente può assumere anche i valori NULL, nel caso in cui colui che ha eseguito l'upload della risorsa non abbia eseguito il login o non si sia ancora registrato
- Visible: anche questo può assumere il valore NULL, nel caso in cui la foto appartiene ad un utente non registrato. Questo campo sta ad indicare che colui che ha eseguito l'upload vuole rendere visualizzabile o meno la risorsa da tutta la community di TakePlace.

La seconda tabella che va a formare il database remoto di TakePlace è la seguente:

| ACCOUNT | |
|---|---|
| UserId | int |
| UserName | varchar |
| Password | varchar |
| CryptoKey | varchar |

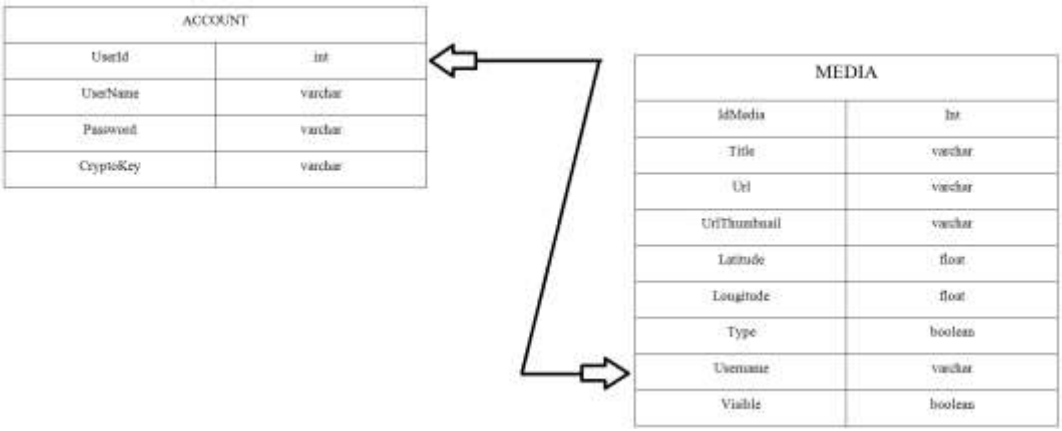
Si è deciso di non inserire l'indirizzo e-mail degli utenti nella tabella Account, in quanto si sarebbero utilizzate solo per le comunicazioni di servizio e per quello si dispone già l'indirizzo Windows Live che hanno usato gli utenti per entrare nel Microsoft Store e scaricare l'applicazione.

Si passerà ora a spiegare i campi della tabella, anche se sono auto-esplicativi.

- UserId: questo è un campo generato dinamicamente ed è auto-incrementante
- UserName: questo è il campo chiave, non può assumere valori NULL e non può assumere valori duplicati. E' anche il campo sul quale viene fatta la Foreign Key con la tabella precedente. Questo campo viene utilizzato in combinazione con la password, per autenticarsi all'interno dell'applicazione
- Password: è la password che gli utenti inseriscono per autenticarsi
- CryptoKey: è una chiave creata dinamicamente che serve per crittografare la password del campo precedente, in modo che non sia visibile pubblicamente, se non prima di averla decriptata.

Le richieste effettuate dall'applicazione richiamano apposito codice che risiede sul server. Questo si occuperà di recuperare i dati dal database e restituirli nel modo migliore al dispositivo, oppure di modificarli.

Di seguito un'immagine che riepiloga brevemente il database presente nella sua interezza:




*Struttura del database remoto*

## 2.4.2 – Lato client

La persistenza dei dati su Windows Phone

Una scelta progettuale di casa Microsoft per questa versione del suo sistema operativo per mobile, è stata quella di non poter utilizzare una scheda di memoria esterna come dispositivo di archiviazione, ma soprattutto l'impossibilità totale per poter avere una persistenza dei dati anche a livello di client.

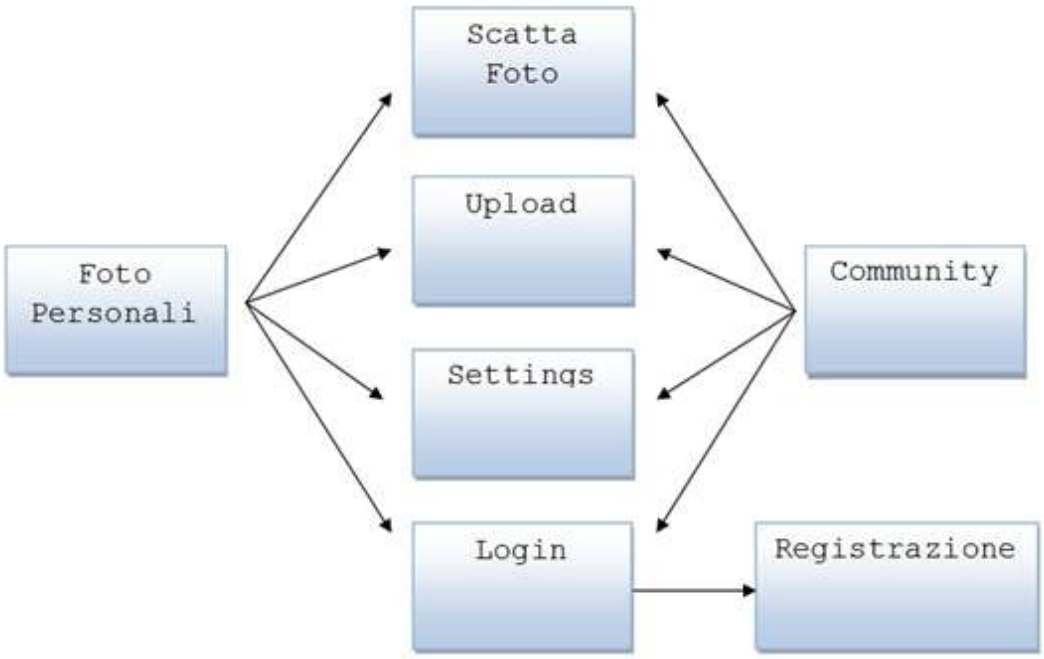
Sullo smartphone ci si può limitare a salvare le credenziali dell'utente in un formato di testo nascosto, in modo che alla prossima apertura della'applicazione l'eventuale utente che si era registrato, non dovrà immettere nuovamente le sue credenziali, quali username e password.

**Le finestre**

Si passerà ora ad elencare brevemente come si è deciso di sviluppare l'interfaccia grafica dell'applicazione TakePlace.

Windows Phone permette di utilizzare 2 tipi di pagine che si portano differenze rispetto ai sistemi operativi concorrenti: l'interfaccia Panorama e quella Pivot.

La prima permette di avere più finestre affiancate che condividono lo stesso sfondo, perciò da come l'impressione che la pagina sia realmente la stessa. Il secondo tipo di visualizzazione, quello Pivot, è invece quello che ho scelto io per la mia applicazione, TakePlace. Questo tipo di finestra è molto simile alla modalità Panorama, però non condivide con le altre attività affiancate lo background, perciò in realtà, quando si esegue lo scroll laterale la pagina viene ricreata completamente da zero (a meno che non vi si sia già acceduto e allora la pagina è ancora in memoria).




*Collegamento delle viste dell'applicazione*

Come si può osservare dall'immagine, le finestre di attività principali sono due.

A queste due pagine ci si arriva dopo aver lanciato l'applicazione. Dopodiché tramite la barra inferiore, che si riduce automaticamente quando non viene utilizzata dopo un certo periodo di tempo, è possibile raggiungere tutte le altre attività tramite i pulsanti presenti sulla barra.

-Community: è la pagina principale che viene visualizzata al lancio dell'applicazione nel caso l'utente non sia ancora registrato. Visualizza sulla mappa di Bing, un volta che vengono ricevuti i dati dal server, i Pushpin relativi ad un upload da parte di un altro utente.

-Foto personali: questa è la pagina che viene nascosta, nel caso in cui l'utente non sia ne collegato, ne registrato. In caso contrario invece questa diventa l'home page. Infatti al lancio dell'applicazione eseguo un controllo sulla disponibilità delle credenziali dell'utente. Qualora venissero trovati le credenziali salvate in un file criptato allora l'utente è già registrato, quindi viene impostata questa come la pagina principale dell'applicazione; per i dettagli implementativi rimando al prossimo e ultimo capitolo.

-Scatta foto: come le prossime che verranno spiegate, questa pagina è possibile raggiungerla tramite i bottoni presenti nella toolbar inferiore dello schermo. Viene data la possibilità all'utente di scattare una foto o di registrare un video, proprio come fa l'applicazione di base del sistema operativo Windows Phone.

-Upload: da questa pagina è possibile scegliere una, più immagini oppure un video da poter caricare sul server scegliendo il punto esatto della mappa in cui si vuole geolocalizzare la propria risorsa.

-Settings: questa è la pagina delle impostazioni, ci sono poche impostazioni da cambiare, ma queste danno all'utente la possibilità di risparmiare l'utilizzo della rete, perciò se attivato, per ogni richiesta fatta al server viene indicato che i dati da ricevere devono essere minori possibili. E' stato scelto di dare la possibilità all'utente di poter condividere tutte le risorse multimediali che ha caricato o che sta caricando, in modo che non debba

preoccuparsi da quale pagina sta eseguendo l'upload. Per gli utenti più sbadati. Inoltre c'è un campo di spunta che, se attivato, permette di visualizzare solo le foto provenienti dalla Community.

-Login: da questa pagina è possibile effettuare il login come utente registrato.

-Registrazione: questa è l'unica pagina che non è possibile raggiungere direttamente al lancio dell'applicazione e dalla toolbar sul lato inferiore dello schermo. Questa scelta è dovuta al fatto che, una volta che ci si è registrati, a questa pagina non verrà fatto più accesso, se invece un utente non vuole registrarsi non avrà la scomodità grafica di vedere ogni volta il pulsante per accedere alla pagina della registrazione.

Ovviamente, da ogni pagina in cui si sta eseguendo un'attività, è possibile tornare all'home page tramite l'apposito pulsante "back" integrato in ogni dispositivo Windows Phone, dalla versione 7.5 in poi.

Come avrete potuto notare, si è cercato di fare in modo che ogni attività dell'applicazione sia facilmente raggiungibile da qualsiasi punto di quest'ultima; questo aspetto è stato pensato per mettere l'utente a suo agio e dare, fin dal primo utilizzo dell'applicazione, un senso di immediatezza che tante altre applicazioni (di Windows Phone come di Android o iOS) non riescono a dare.

# Implementazione

## 3.1 – Server Side

### 3.1.1 – Database remoto

Per lo sviluppo del database remoto, cioè che risiede sul server, si è utilizzata la tecnologia MySQL.

Per prima cosa si tratta di un database open-source, per cui di libero utilizzo, infatti è distribuito sotto licenza GPL (General Public License).

Il secondo motivo è l'ampia disponibilità di server su cui è possibile utilizzarlo. Difatti se per un database come SQL Server di Microsoft è necessario disporre di un server dedicato oppure appoggiarsi a servizi comunque onerosi economicamente, per MySQL vengono offerti gratuitamente molti servizi di base. Per questo progetto si è utilizzato proprio uno di questi servizi.

Il terzo motivo è la facilità di utilizzo che presenta. E' infatti possibile interfacciarsi ad esso utilizzando comodi strumenti, come phpMyAdmin, che si tratta di un servizio Web che permette di eseguire i comandi SQL direttamente da una pagina web dopo essersi collegati con il proprio account.

In ultima analisi MySQL permette di essere interfacciato in maniera ottimale tramite il linguaggio di programmazione server-side PHP, che ho scelto per le interrogazioni dei dati effettuate sul database.

Ricordo che la parte di progettazione è stata approfondita nel secondo capitolo, in cui è presente anche come è stato creato il database.

### 3.1.2 – Richieste

Di seguito verranno presentate le richieste che vengono fatte dal dispositivo mobile al database che risiede sul server, che a sua volta restituisce le informazioni al client.

Per evitare che le interrogazioni al database diventino troppo onerose a livello di risorse richieste e che ci sia un eccessivo consumo di banda da parte del client, in quasi tutte le richieste effettuate viene specificato il livello di zoom attuale del dispositivo e la posizione corrente. Grazie a questo stratagemma il server remoto, effettuerà i dovuti calcoli ed estrarrà dal database solo quelle informazioni, che all'utente del dispositivo interessano in quell'istante. Facendo un esempio pratico per chiarire meglio le idee, diciamo che il client su cui è visualizzata la mappa di Bing si trova alle coordinate x = 142.5, y = 2.77 con una distanza di visuale di 10 chilometri, il server dovrà inviare le informazioni solo del punto (142.5,2.77) e dei 10 chilometri circostanti.

- Recupero di tutte le foto e tutti i video della zona circostante:

  ```
  SELECT Title, UrlThumbnail, Latitude, Longitude, Type
  FROM MEDIA
  WHERE Visible = TRUE
  AND Latitude > $lat – zoom
  AND Latitude < $lat + zoom
  AND Longitude > $long – zoom
  AND Longitude < $long + zoom
  ```

  In questo modo vengono restituite solo le informazioni relative alla zona che circonda l'utente sulla mappa. Ad ogni spostamento che supera una soglia minima viene eseguita nuovamente la richiesta al server. Quest'ultimo restituisce i dati in formato JSON (JavaScript Object Notation) che verranno interpretati dal client.

- Recupero di tutte le foto e tutti i video di un utente specifico:

```
SELECT Title, UrlThumbnail, Latitude, Longitude, Type
FROM MEDIA
WHERE Username = '$user'
```

In questo modo vengono restituite dal server tutte le risorse che sono attribuite ad un utente specifico. Su questa query non eseguo controlli sull'esistenza dello Username, in quanto i controlli sono stati fatti all'accesso, quindi ho la certezza che l'utente esiste e lo Username sia corretto.

- Recupero di tutte le foto (e non i video):

```
SELECT IdMedia, Title, UrlThumbnail, Latitude, Longitude
FROM MEDIA
WHERE Type = 1
AND Visible = TRUE
AND Latitude > $lat – zoom
AND Latitude < $lat + zoom
AND Longitude > $long – zoom
AND Longitude < $long + zoom
```

- Recupero di tutti i video:

```
SELECT IdMedia, Title, UrlThumbnail, Latitude, Longitude
FROM MEDIA
WHERE Type = 0
AND Visible = TRUE
AND Latitude > $lat – zoom
AND Latitude < $lat + zoom
AND Longitude > $long – zoom
AND Longitude < $long + zoom
```

  In queste ultime due query (molto simili) non è stato inserito il tipo di risorsa che il client richiede in quanto si sa già in partenza quale ci si aspetta, inoltre è stata spostata al primo posto della clausula WHERE in quanto rende più efficiente la query eseguendo una prima scrematura solo su un campo boolean.

Come è stato accennato nel secondo capitolo, si lascia all'utente la possibilità di richiedere solo un numero limitato di risorse (è possibile impostare questo parametro nella pagina Settings) nel caso si disponga di banda ad alta latenza o una connessione dati molto lenta. Per questo motivo ad ogni chiamata che faccio al server prima di eseguire la query, faccio un controllo sulla presenza o meno di un parametro definito '$top' e nel caso sia presente inserisco una clausula nella SELECT iniziale. Senza ripetere tutte le query farò un esempio di query modificata:

- Recupero di una parte delle foto e dei video della zona circostante:

```
SELECT IdMedia, Title, UrlThumbnail, Latitude, Longitude, Type
FROM MEDIA
WHERE Visible = TRUE
AND Latitude > $lat – zoom
AND Latitude < $lat + zoom
AND Longitude > $long – zoom
AND Longitude < $long + zoom
LIMIT 10
```

Questa query prende solo i primi 10 media dalla tabella Media.

- Upload di una foto:

```
INSERT INTO MEDIA(Title, Url, UrlThumbnail, Latitude,
Longitude, Type, UserName, Visible)
VALUES ($title, $url, $urlThumbnail, $latitude,$longitude, 1,
$userName, $visible)
```

- Upload di un video:

```
INSERT INTO MEDIA(Title, Url, UrlThumbnail, Latitude,
Longitude, Type, UserName, Visible)
VALUES ($title, $url, $urlThumbnail, $latitude,$longitude, 0,
$userName, $visible)
```

Le due query eseguono l'inserimento dei valori all'interno del database, dove i valori del campo UrlThumbnail vengono ricavati a partire dal valore del campo Url, in quanto le immagini vengono salvate in una cartella e poi viene creata la copia corrispondente nella cartella "Images/Thumbnail/" in modo tale che non viene richiesta esplicitamente un'immagine a dimensione reale.

- Visualizzazione di una singola foto o di un video:
  SELECT Url
  FROM MEDIA
  WHERE IdMedia = $id

  Finalmente ecco spiegato il motivo per cui in tutte le richieste inserisco anche il campo relativo all'ID della risorsa. Quando ho bisogno di ricevere l'url dell'immagine o del video a dimensione reale, non devo fare altro che specificare l'ID nella query sulla tabella MEDIA. In questo caso non ho bisogno di specificare di quale risorsa si tratti, se foto o video, in quanto l'ID è univoco.

- Login di un utente:
  CALL cryptPassword($password,@pwd)
  SELECT *
  FROM ACCOUNT
  WHERE UserName = '$user'
  AND Password = @pwd

Come si può notare la query per il login è leggermente diversa rispetto alle altre, questo è dovuto al fatto che le password sono state criptate sul database per un fattore di sicurezza, quindi per verificare che la password immessa dall'utente e quella memorizzata sul database siano le medesime, devo prima criptare la password inviatami dal client. Questo lo faccio tramite una Stored Function che richiamo prima del comando SQL di SELECT, questa funzione permette di criptare la password e conseguentemente verificare che siano uguali.

Nel caso in cui dal database non ricevessi nessun dato, restituisco all'utente un errore di autenticazione.

- Registrazione di un nuovo utente:

  SELECT UserName
  FROM ACCOUNT
  WHERE UserName = '$user'

```
CALL cryptPassword($password,@pwd)
INSERT INTO ACCOUNT(UserName, Password, CryptoKey)
VALUES($userName, @pwd)
```

Nella query precedente prima si verifica che non esista ancora nessun utente con lo username specificato, quindi si inseriscono all'interno della tabella ACCOUNT i valori di userName e password; quest'ultima criptata tramite la stored function che ho accennato poco sopra.

La stored function si presenta in questo modo:

```
CREATE PROCEDURE cryptPassword(OUT param1 VARCHAR)
RETURN VARCHAR
BEGIN
{ …}
END
```

# 3.2 – Client Side

## 3.2.1 – Perché Windows Phone?

Per quanto riguarda la fase di sviluppo del codice sul dispositivo la scelta, come già accennato è ricaduta su Windows Phone 7.5 (Mango).

Oltre alle ragioni di carattere personale e di preferenze vi sono altre motivazione che hanno portato a fare questa scelta e verranno presentate brevemente qui sotto.

Innanzitutto guardando lo stadio di sviluppo dei futuri prodotti di casa Microsoft, c'è quello di Windows 8 che inevitabilmente, visto il periodo della "moda" per il mobile, condividerà il kernel e il framework .NET con la

controparte Windows Phone 8. Da questo discorso nasce, almeno personalmente, la speranza che Windows con le sue nuove mosse di mercato possa rientrare in un mercato che è stato letteralmente conquistato da concorrenti molto agguerriti come iOS di Apple e Android di Google.

Inoltre c'è da ricordare che il Windows Store è ancora acerbo rispetto alle controparti, contando ad ora poco più di 120 mila applicazioni scaricabili, contro le 700 mila e le 650 mila di iOS e Android rispettivamente. Se per molti questo può essere un difetto, penso che per gli sviluppatori di software possa essere un grande vantaggio, in quanto è ancora possibile sviluppare app che non hanno trovato ancora posto sullo Store. Questo discorso per iOS e Android è molto più difficile.

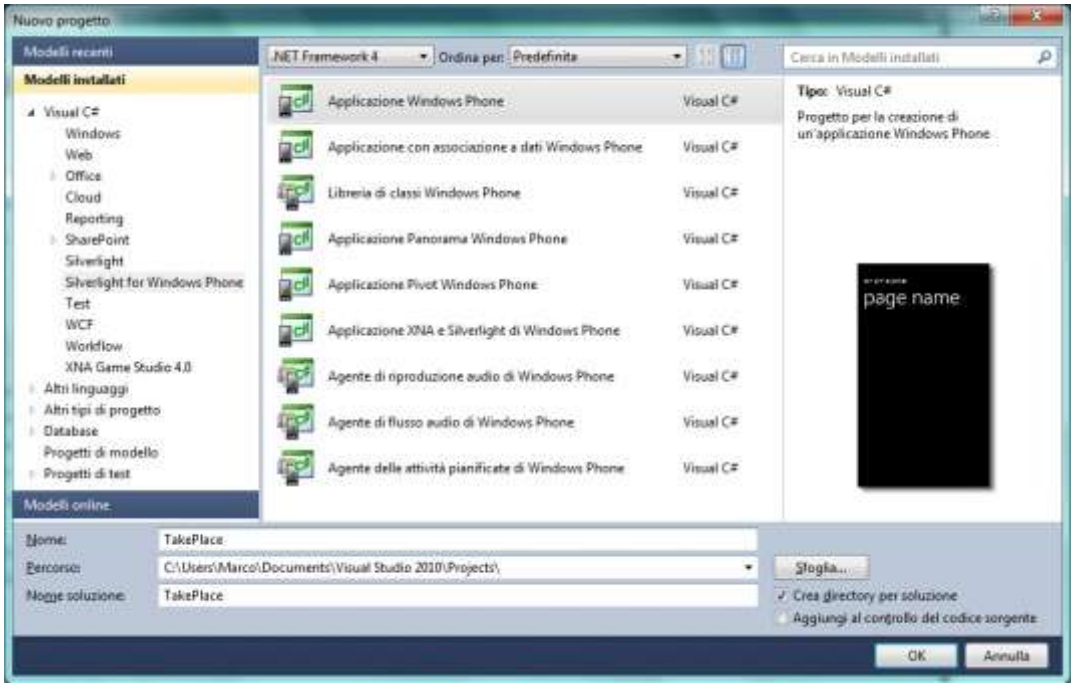
Un altro punto molto importante riguarda il costo. Ricordo che per diventare sviluppatore Apple ci si deve registrare come sviluppatori ufficiali sull'App Store pagando una quota annua di di 99 dollari per la pubblicazione, trattenendo il 30% dal costo di vendita di ogni applicazione. Per Android il costo risulta essere minore, 25 dollari una tantum, ma è anche vero che sul Play Store non ci sono controlli di nussuna sorta. Il discorso cambia per Microsoft, in quanto il costo annuo di iscrizione sarebbe di 99 dollari oltre ad una trattenuta del 20% sulla vendita di ogni applicazione, ma in casa Microsoft hanno pensato anche ai giovani sviluppatori, che magari hanno la voglia di sviluppare qualche applicazione per fare esperienza ma non si possono permettere certe cifre. Per questo se ci si registra come studente sviluppatore al Microsoft Store è possibile immettere sul mercato le proprie applicazione a patto che siano completamente gratuite e senza forme di pubblicità all'interno delle App.

Un'altra nota positiva è il fatto che per sviluppare applicazioni per Windows Phone si deve utilizzare Microsoft Visual Studio, che mette a disposizione una serie di funzionalità che facilitano molto la vita di noi programmatori, quali ambienti di test semi-automatizzati, emulatori, tool per generare codice direttamente dal diagramma delle classi.

Un ultimo punto importante è sicuramente la possibilità di testare la propria applicazione anche su un dispositivo fisico, possibilità che viene data anche da Android tramite Eclipse, ma non dall'ambiente di sviluppo Xcode di Apple.

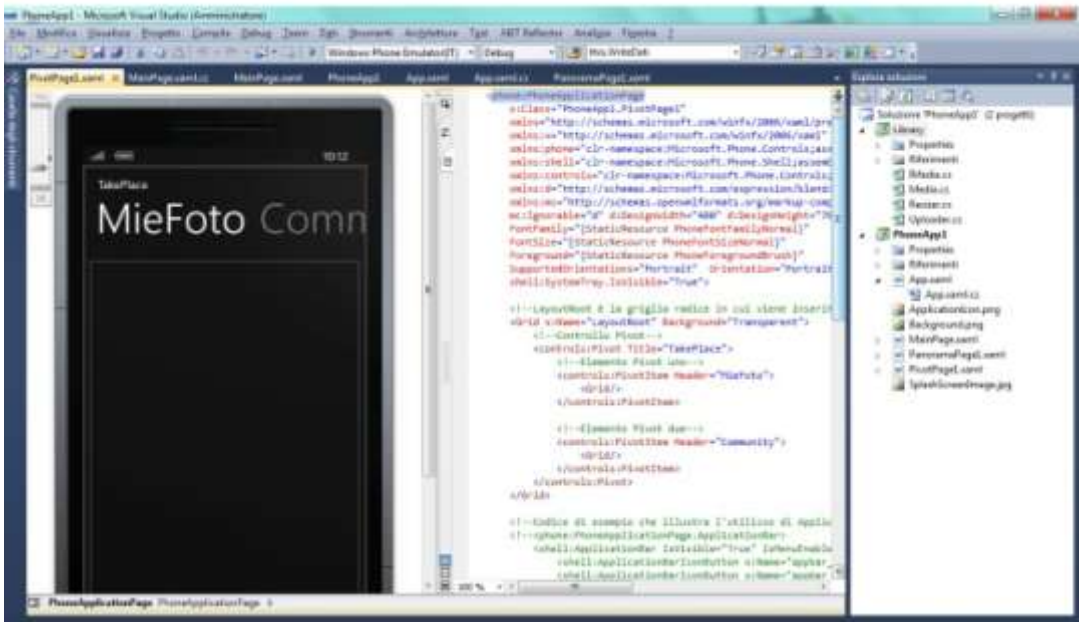
## 3.2.2 – Sviluppare in Windows Phone

Per sviluppare un progetto in Window Phone tramite Visual Studio è necessario creare prima un progetto di tipo Windows Phone Project, oltre a questo Visual Studio mette a disposizione svariati progetti da cui è possibile iniziare, mettendo a disposizione un template di codice da cui è comodo partire




*Creazione di un nuovo progetto in Visual Studio 2010*

Una volta scelto il tipo di progetto che si vuole sviluppare la schermata che appare sarà come questa:




*Come si presenta la schermata del progetto*

La definizione dello stile delle finestre del progetto è demandata allo XAML (eXtensible Application Markup Language) mentre la parte di code behind è demandata al C#.

D'ora in poi ci si soffermerà sulla descrizione del codice che si è sviluppato per fare funzionare l'applicazione.

All'interno della classe App, che deriva a sua volta dalla classe Application, è possibile trovare i metodi e le proprietà che servono al corretto funzionamento dell'applcazione. Tra questi troviamo il Frame radice dell'applicazione, il costruttore della classe App e i cinque eventi principali dell'applicazione:

- Application_Launching: questo è il delegate che viene richiamato quando l'applicazione viene lanciata per la prima volta. Questo codice non viene eseguito quando l'applicazione viene riattivata dalla memoria

- Application_Activated: questo è il delegate che viene richiamato quando l'applicazione viene riportata in primo piano, ma era già stata avviata per la prima volta precedentemente. Questo codice non viene eseguito al lancio dell'applicazione la prima volta

- Application_Deactivated: questo è il delegate che viene richiamato quando l'applicazione viene inviata in background, cioè disattivata. Questo codice non viene eseguito alla chiusura dell'applicazione

- Application_Closing: questo è il delegate che viene richiamato quando l'applicazione viene chiusa definitivamente. Questo codice non viene eseguito quando l'applicazione viene disattivata o portata in secondo piano

- RootFrame_NavigationFailed: questo è il delegate che viene richiamato quando un operazione di navigazione tra le pagine ritorna esito negativo.

## Localizzazione dei Contenuti

La piattaforma Windows Phone, offre una potente funzionalità, cioè quella di poter localizzare le stringhe in maniera molto veloce. Infatti basta semplicemente creare un file di risorse con il nome dato da:
-Nome del file da localizzare,
-Lingua e cultura nel quale si vuole tradurre il proprio file,
-Estensione del file .resx
Esempio: MainPage.xaml.EN-us.resx oppure MainPage.xaml.IT-it.resx.
Nei due casi precedenti vengono creati due file, uno per la localizzazione della lingua inglese - americana e l'altro per l'italiano.
La scelta su quale file di risorse caricare viene fatta in base al sistema operativo sul quale si esegue l'applicazione oppure sulle impostazioni della lingua di sistema, nel caso quest'ultima fosse diversa dalla lingua originaria del sistema operativo.

Una volta che sono stati creati i file di risorse, è sufficiente andare ad inserire nelle proprietà dei controlli che si vogliono localizzare, un tag: *<meta:resourcekey=”ID”>* dove ID è l’identificativo all’interno del file di risorse.

## 3.2.3 – Geolocalizzazione

Inizialmente si era pensato di utilizzare le mappe di Google perché, avendo più tempo di sviluppo sulle spalle,  dispongono di una maggiore accuratezza e di un maggior numero di luoghi presenti. Microsoft però non ha ancora autorizzato ufficialmente le API di Google Maps, per questo motivo un’applicazione presentata sul Windows Store per essere commercializzata, non verrebbe accettata.

Detto questo, si è passati ad utilizzare le mappe di Microsoft Bing che hanno un’ottima definizione, ma un numero di luoghi presenti inferiore.

Per utilizzarle è necessario registrare la propria applicazione con le mappe di Bing, che inizialmente offrono un uso gratuito per tre applicazioni. Dopo essersi registrati si riceve un’ID da passare come parametro al costruttore del metodo ApplicationIdCredentialsProvider


Dopo aver istanziato nella classe App il servizio di mappe di Bing è possibile utilizzarlo all’interno di tutto di tutta l’applicazione tramite il seguente metodo

```
private readonly CredentialsProvider _credentialProvider = new ApplicationIdCredentialsProvider(App.Id);
public CredentialsProvider CredentialsProvider
{
    get { return _credentialProvider; }
}
```

In questo modo possiamo creare un DataBinding dall'interfaccia XAML grazie all'apposita proprietà dell'oggetto Map:

```
CredentialsProvider="{Binding CredentialsProvider}"
```

In questo modo l'oggetto mappa verrà creato automaticamente utilizzando l'apposito ID fornito da Microsoft.

Di seguito si proporrà il codice sviluppato per gestire i mutamenti, effettuati da parte dell'utente, sullo zoom della mappa, che in questo caso servirà per eseguire una nuova richiesta al server, in quanto devono essere aggiornati i dati.

```
private double _zoom;
public double Zoom
{
    get { return _zoom ;}
    set
    {
        double zoomRichiesto = Math.Min(MINZOOM, Math.Min(MAXZOOM, value));
        if (zoomRichiesto != _zoom)
        {
            _zoom = zoomRichiesto;
            NotifyPropertyChanged("Zoom");
        }
    }
}
public event PropertyChangedEventHandler PropertyChanged;
private void NotifyPropertyChanged(string proprietà)
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(proprietà));
}
```

Grazie a questo costrutto ogni volta che l'utente esegue uno zoom in avanti o all'indietro, tramite il delegate sopra proposto vengo avvisato del cambiamento e si può agire di conseguenza, cioè effettuare una nuova chiamata al server.

Infine, per visualizzare le informazioni sulla mappa, cioè le risorse che vengono inviate dal server sottoforma di foto e video, si deve creare un oggetto Pushpin che prende come parametro in ingresso un immagine. Quest'immagine sarà, nel caso sia il riferimento di una foto, la thumbnail della foto stessa, creata a partire dall'Url della risorsa presente sul server; nel caso invece si tratti del riferimento ad un video, l'immagine passata come parametro sarà il primo fotogramma catturato dal video, con una presentazione diversa rispetto a quella delle foto, con i bordi superiore e inferiori neri.

```
public class Pushpin
{
    private Microsoft.Phone.Controls.Maps.Pushpin AddPushpin(ImageBrush image)
    {
        if (!Availability.Network())
            return null;
        Microsoft.Phone.Controls.Maps.Pushpin pin = new Microsoft.Phone.Controls.Maps.Pushpin();
        pin.Location = new System.Device.Location.GeoCoordinate();
        pin.Background = new SolidColorBrush(Colors.Orange);
        pin.Content = new Ellipse
        {
            Fill = image,
            UseLayoutRounding = true,
            Height = 100,
            Width = 100,
            Opacity = 0.5
        };
        pin.Tap += new EventHandler<GestureEventArgs>(OnPinTap);
        return pin;
    }
    public EventHandler<GestureEventArgs> OnPinTap { get; set; }
}
```

In questo metodo, da come si può appurare dall'immagine sopra, si crea un oggetto Pushpin con delle proprietà condivise, tranne che l'immagine di sfondo. Queste proprietà includono:

Location: la posizione attuale dell'utente, la chiamata al metodo GeoCoordinate() è un wrapper del .NET Framework che richiama il servizio GPS per ricevere le informazioni sulla posizione.

Background: è il colore di sfondo del Pushpin, impostato ad arancione per avere un forte contrasto con il colore di sfondo della mappa

Tap: è il gestore dell'evento OnTap, cioè il delegato che viene richiamato nel momento in cui l'utente esegue un tocco o un doppio tocco sull'icona Pushpin.

Prima di impostare tutte le proprietà del Pushpin, si esegue un controllo sulla presenza di connessione di rete e, nel caso fosse assente, ritorna alla classe chiamante NULL, la quale avviserà l'utente che non è possibile completare l'operazione a causa di un errore.

### 3.2.4 – Acquisizione Media

Per l'acquisizione e l'upload delle risorse multimediali sono stati utilizzati principalmente 3 metodi: GetData, PostData e FtpData.

GetData è il metodo che si occupa di inviare una richiesta GET al server, il codice che è stato utilizzato è il seguente:

```
public static string HttpGet(string URI)
{
    System.Net.WebRequest req = System.Net.WebRequest.Create(URI);
    System.Net.WebResponse resp = req.GetResponse();
    System.IO.StreamReader sr = new System.IO.StreamReader(resp.GetResponseStream());
    string response = sr.ReadToEnd();
    sr.DiscardBufferedData();
    sr.Dispose();
    return responseString;
}
```

Il codice del metodo PostData è il seguente:

```
public static string HttpPost(string URI, string Parameters)
{
    System.Net.WebRequest req = System.Net.WebRequest.Create(URI);
    req.Method = "POST";
    byte[] bytes = System.Text.Encoding.ASCII.GetBytes(Parameters);
    req.ContentLength = bytes.Length;
    System.IO.Stream os = req.GetRequestStream();
    os.Close();
    System.Net.WebResponse resp = req.GetResponse();
    if (resp == null)
        return null;
    System.IO.StreamReader sr = new System.IO.StreamReader(resp.GetResponseStream());
    string response = sr.ReadToEnd();
    sr.DiscardBufferedData();
    sr.Dispose();
    return response;
}
```

Mentre il codice del metodo FtpData è il seguente:

```
public static void FtpData(string filename)
{
    FtpWebRequest reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri(url));
    reqFTP.Credentials = new NetworkCredential(ftpUserID, ftpPassword);
    reqFTP.KeepAlive = false;
    reqFTP.Method = WebRequestMethods.Ftp.UploadFile;
    reqFTP.UseBinary = true;
    reqFTP.ContentLength = fileInf.Length;
    int buffLength = 2048;
    byte[] buff = new byte[buffLength];
    int contentLen;
    FileStream fs = fileInf.OpenRead();
    try
    {
        Stream strm = reqFTP.GetRequestStream();
        contentLen = fs.Read(buff, 0, buffLength);
        while (contentLen != 0)
        {
            strm.Write(buff, 0, contentLen);
            contentLen = fs.Read(buff, 0, buffLength);
        }
        strm.Close();
        fs.Close();
}
```

Il funzionamento del codice per tutti e tre i metodi è molto simile, infatti tutti e tre impostano prima, la connessione al server remoto tramite l'url che gli si passa come parametro, in secondo luogo istanziano un Stream per ricevere/inviare i dati, infine rilasciano tutte le risorse che erano state allocate da classi che fanno uso di codice non gestito (Unmanaged) e che quindi non prevedono di essere liberate automaticamente dal Garbage Collector del .NET framework. L'unica differenza notevole che può essere imputata a uno dei tre metodi è quello per l'upload di risorse tramite l'FTP, infatti questo metodo viene invocato in un Thread separato, in modo che l'utente continui normalmente la sua esperienza. Il codice utilizzato è molto semplice:

```
static void StartAsyncUpload()
{
    System.Threading.Thread t = new System.Threading.Thread( FtpData );
    t.Start();
}
```

Questo permette di eseguire il codice in un secondo Thread, lasciando intatto il normale flusso dell'applicazione.

Una volta che ricevute tutte le risorse di una singola richiesta da parte del server, si utilizza un particolare tipo di lista messa a disposizione dal Framework a partire dalla versione 3.5: la classe Lazy<T>, dove T è il Template che deve essere sostituito dall'oggetto con cui vogliamo creare l'istanza Lazy. Questo particolare tipo di Lista, permette di istanziare il valore di ogni singolo oggetto solo nell'istante in cui questo viene richiesto, permettendo così il cosiddetto Lazy Initialization. Avendo implementato le classi, che ereditano da un'interfaccia pubblica, è stato possibile utilizzare un tipo generico quale Lazy<IMedia>, che permette tramite il paradigma del Late Binding, di modificare la propria implementazione dei metodi in base al tipo di oggetto che viene istanziato, nel caso questo fosse una foto o un video.

### Formattare i dati Json

Come si sarà potuto notare nel secondo capitolo, all'interno del diagramma delle classi, c'è una classe statica chiamata JSON. Questa è un utility che viene richiamata ogni qualvolta il client debba inviare informazioni al server e viceversa. Questo perché si devono formattare eventuali informazioni che vengono ricevute o inviate, che non sono contenute all'interno della chiamata GET o POST. Per questo motivo il codice di cui si serve l'applicazione per dialogare con il server è il seguente:

```
public static string ToJson<T>(this T obj)
{
    System.Runtime.Serialization.Json.DataContractJsonSerializer serializer =
        new System.Runtime.Serialization.Json.DataContractJsonSerializer(typeof(T));
    System.IO.MemoryStream ms = new System.IO.MemoryStream();
    serializer.WriteObject(ms, obj);
    string serialized = System.Text.Encoding.Default.GetString(ms.ToArray());
    ms.Dispose();
    return serialized;
}
```

Questo è il metodo che prende in ingresso un qualsiasi tipo di oggetto , lo serializza e lo formatta seguendo le regole standard Json.

```
public static T FromJson<T>(this string strJson)
{
    System.IO.MemoryStream ms =
        new System.IO.MemoryStream(System.Text.Encoding.ASCII.GetBytes(strJson));
    object serializer =
        new System.Runtime.Serializ                    ontractJsonSerializer(typeof(T)).ReadObject(ms);
    ms.Dispose();
    return (T)serializer;
}
```

namespace System

Il precedente codice invece mostra come, passata in ingresso una stringa, essa viene decodificata e in base al tipo di dato che viene indicato al metodo (template T) viene eseguito il cast diretto e viene restituito al metodo chiamante l'oggetto deserializzato.

# Conclusioni e sviluppi futuri

Il lavoro svolto si è mostrato molto interessante e ha portato ad affrontare molte differenti tematiche relative all'ambiente di sviluppo scelto.

Seppur semplice, rispetto ad altre applicazioni di tutto il rispetto nel mondo Windows Phone e non, lo sviluppo dell'applicazione si è rivelata un'opportunità per vedere dal vivo lo sviluppo di un prodotto, che potrebbe diventare commerciale, dall'inizio alla sua fine, vagliando diversi linguaggi di programmazione, passando dalla progettazione logica e concettuale, fino al design del database.

Mentre veniva sviluppato il progetto sono venute in mente idee su una sua possibile evoluzione futura.

Probabilmente infatti, al giorno d'oggi entrare su uno Store con un'applicazione che non ha collegamenti con i più importanti social network è un pò rischioso perché mina la propagazione dell'applicazione. Per questo ci si riferisce ad un eventuale integrazione con Facebook e Twitter, per quanto riguarda la condivisione di foto e video. Inoltre vedendo il grande successo di software come FourSquare, non è remota l'idea di inserire il Check-In di un luogo (cioè il geo Tag del luogo in cui ci si trova).

Un importante aggiornamento che potrebbe essere fatto all'applicazione potrebbe essere quello dell'aggiornamento del server, infatti in questo momento le immagini e i video vengono ridimensionati a causa della limitata disponibilità sul disco del server. Questo tipo di aggiornamento potrebbe migliorare l'esperienza dell'utente, sapendo che i propri dati multimediali non perderanno della loro qualità una volta che verranno caricati sul Cloud.

Infine un ulteriore passo per spingere il grande pubblico a conoscere l'applicazione può essere la conversione di quest'ultimo, per essere sfruttato anche sull'ultimo sistema operativo di casa Microsoft che sta per uscire: Windows 8. Purtroppo al momento della scrittura non è ancora disponibile l'SDK per poter sviluppare applicazioni per Windows 8 e Windows Phone 8.

# Bibliografia


-Worldwide Smartphone and Client PC shipped

http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011

- Worldwide Smartphone 2012–2016 Forecast and Analysis
http://www.idc.com/getdoc.jsp?containerId=233553

- The Future of Smartphone Growth

http://theweek.com/article/index/224535/the-future-of-smartphone-growth-by-the-numbers

- Alexander Wolfe, "Is The Smartphone Your Next Computer?"

http://www.informationweek.com/news/personal-tech/smart-phones/210605369?pgno=1

- Scott Webster, "Crittercism: "iOS apps crash more than Android apps"

http://www.androidguys.com/2012/02/03/crittercism-ios-apps-crash more-than-android-apps/

- Create Stored Functions in MySQL

http://dev.mysql.com/doc/refman/5.0/en/create-procedure.html

- Windows Phone make progress in Europe

http://www.kantarworldpanel.com/global/News/Windows-makes-progress-in-Europe