ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

FACOLTA' IN SCIENZE MATEMATICHE, FISICHE E NATURALI

Applicazione Smartphone Geo-Social Game

TESI DI LAUREA IN MOBILE WEB DESIGN

RELATORE:

Dott. Mirko Ravaioli

PRESENTATA DA:

Nicola Lucchi Casadei

Seconda sessione

SOMMARIO

1 INTRODUZIONE

Con il passare del tempo, l'evoluzione tecnologica che ha accompagnato lo sviluppo dei normali PC, ha coinvolto i dispositivi così detti "mobili". Evoluzione che li ha trasformati da semplici organizer "da tasca" a veri e propri terminali ricchi di funzionalità, discreta potenza di calcolo ma soprattutto di connettività. Quest'ultima caratteristica li ha resi estremamente versatili soprattutto per applicazione di tipo aziendale e di produttività personale.

Esistono sostanzialmente quattro tipologie di dispositivi mobili, ognuna dotata di caratteristiche particolari da tenere in considerazione quando progettiamo le nostre applicazioni:

- **Pocket PC**: sono i così detti "palmari". Tra le principali caratteristiche di questi dispositivi troviamo il display di tipo touch screen, la compattezza, una discreta possibilità di espansione e una discreta potenza di calcolo. I Pocket PC si dividono a loro volta in due famiglie:

  - Standard (non dotati di funzionalità telefoniche)
  - Phone Edition: hanno tutte le caratteristiche dei Pocket PC standard con l'aggiunta delle funzionalità telefoniche.

- **Smartphone**: Uno smartphone è un dispositivo portatile che mette in comune le funzionalità di un telefono cellulare con quelle di un computer desktop, come la navigazione su Internet, la riproduzione di file audio e video, l'invio di mail e molto altro ancora.

- **Tablet PC**: sono dispositivi molto simili ai normali Notebook ma dotati di touch screen e di funzionalità di riconoscimento della scrittura.
- **UMPC (Ultra Mobile PC)**: sono una via di mezzo tra il Pocket PC e il Notebook. Più grandi e potenti di un Pocket PC ma più piccoli e meno potenti di un Notebook, sono più orientati verso il mercato consumer che business.

1.1 DIFFUSIONE DEGLI SMARTPHONE SUL MERCATO

La diffusione di questi dispositivi parte nel 1994 con l'immissione sul mercato di Simon da parte di IBM: il primo dispositivo touch screen che permetteva di inviare mail, fax, usare una calcolatrice ed altre funzioni identiche a quelle di un palmare.

Da questa data fino ad oggi la domanda degli smartphone nel mercato globale è cresciuta enormemente, infatti i dati resi noti dalla Strategy Analytics affermano che nel terzo trimestre del 2012 si è superata la soglia del miliardo di dispositivi venduti (una perona su sette al mondo possiede uno smartphone).

Un elemento di svolta nella mondo smartphone, che ha segnato il vero e proprio boom sul mercato di questi dispositivi, è rappresentato dall'avvento di iPhone di Apple, nel giugno del 2007. Sebbene già allora esistessero molti dispositivi capaci di collegarsi alla rete in mobilità, è stato solo con il cellulare di Apple, e con la conseguente uscita dei primi dispositivi Android, che lo sfruttamento della connettività mobile, forse

per la possibilità di disporre un modello d'uso innovativo, ha iniziato a crescere in maniera esponenziale rivoluzionando il mondo consumer e quello aziendale.

Inoltre anche l'avvento e la diffusione dei social network ha influito sulla crescita del mercato smartphone in quanto gli utenti hanno sentito sempre più l'esigenza (se non, addirittura, l'ossessione) di una costante presenza sul web, portando così le persone ad aumentare sia il loro tempo trascorso online, che il numero e la tipologia di dispositivi con i quali si collegano con la conseguenza di assottigliare la differenza tra i dispositivi personali e quelli professionali.

Questo crescente bisogno per gli utenti di una costante presenza sul web è stato soddisfatto dalle sempre più avanzate tecnologie di connessione dati wireless (2G,3G ed il 4G o Long Term Evolution) che hanno reso possibile il collegamento ad internet da pressochè qualsiasi area del pianeta. Il poter svolgere la maggior parte delle attività che si svolgevano con un pc desktop (dalla navigazione alle applicazioni, dall'email al business management, dai social network al gaming e dalla lettura allo shopping) è uno dei principali motivi per i quali gli utenti stanno sempre più abbandonando le postazione fisse per rivolgersi al mondo mobile. I dati parlano chiaro: Le vendite di smartphone, per la prima volta, nel 2012 hanno superato le vendite di client PC, categoria in cui la società di ricerca Canalys inserisce anche i tablet. 487.7 milioni di smartphone acquistati in un anno contro i 415 milioni di client PC. In un solo anno il mercato smartphone è crescito del 63% (487.7 milioni contro i 299.7 del 2010), mentre il mercato dei client PC registra una crescita più modesta (15%).

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

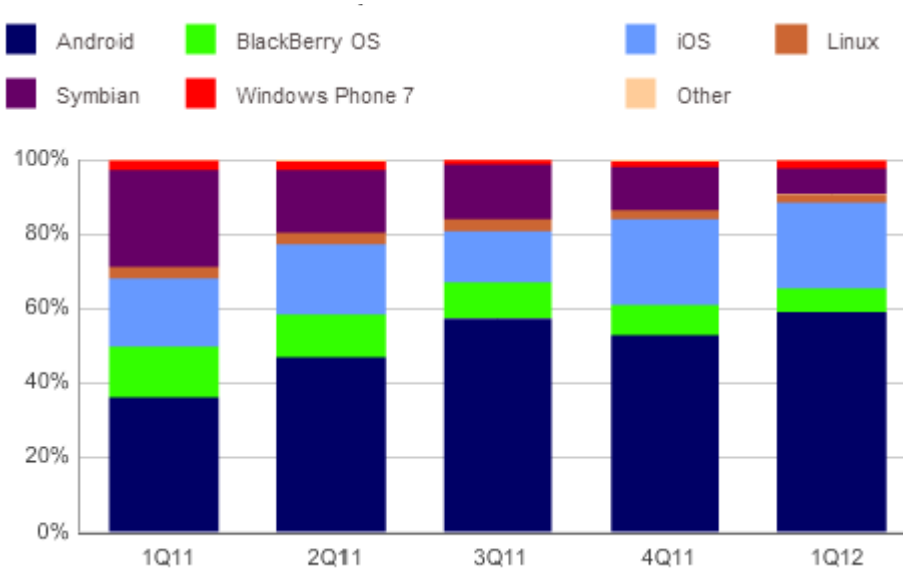
1.2 SISTEMI OPERATIVI PER DISPOSITIVI MOBILI

Un sistema operativo è costituito da un insieme di componenti software che hanno il compito di tradurre il funzionamento logico del calcolatore in una serie di comandi direttamente interpretabili dalle parti fisiche (processore, memoria, periferiche). Rappresenta l'anima di ogni strumento di dotato di capacità di elaborazione. Ad alto livello, invece, si occupa di gestire la comunicazione tra il dispositivo e l'utilizzatore attraverso l'interfaccia-utente.

A differenza dei sistemi operativi per pc, i sistemi operativi per dispositivi mobili devono affrontare problematiche derivanti alla limitatezza delle risorse (CPU, memoria), modalità di input differenti (touchscreen), e diversi protocolli per il trasferimento dati (bluetooth, ecc..) e la connessione a internet (2G, 3G, ecc..).

In seguito è riportato un elenco dei sistemi operativi per dispositivi mobili attualmente in commercio, e la loro divisione del mercato:

Sistema Operativo	Compagnia	Sistema Operativo	Compagnia
iOS	Apple	Symbian	Symbian Foundation
Android	Google Inc.	Meego	Intel e Nokia
webOS	HP/Palm	Bada	Samsung
Windows Mobile	Microsoft	Maemo	Nokia
Windows Phone	Microsoft	Openmoko	Openmoko Inc.
BlackBerry	RIM	Tizen, Meego	Limo Foundation
Symbian	Symbian Foundation	LiMo	Linux Foundation



Come si può notare dal grafico, Android ed iOs detengono più dell'80% del mercato mentre blackberry OS e symbian stanno perdendo sempre più terreno, solo windows phone ha registrato una costante anche se piccola crescita. Pur essendo uscito nel 2010, in uno scenario di mercato

ben definito, Windows Phone oggi detiene circa il 5% delle quote di mercato. Secondo le stime della società di analisi di mercato IDC le quote di mercato del sistema operativo di casa Microsoft dovrebbero eguagliare quelle di iOS (circa il 19% per entrambe).

In seguito vengono elencate i vari dettagli-differenze tra i principali sistemi operativi mobile.

1.2.1 ANDROID

Android è un sistema operativo basato sul kernel di Linux utilizzato principalmente per dispositivi mobile touchscreen come smartphone e tablet. È stato sviluppato da Google in collaborazione con l'OHT (Open Headset Alliance, un consorzio di 84 compagnie per lo sviluppo di "Standard aperti" per dispositivi mobili). Android nasce nel 2008 come sistema operativo open source ed a fine 2010 diventa la piattaforma smartphone dominante superando symbian. Essendo un sistema operativo Open Source, ogni casa costruttrice di dispositivi mobili potrà personalizzare ed aggiungere funzioni al sistema operativo.

Android gestisce la memoria RAM per mantenere al minimo i consumi di energia: quando una applicazione non è utilizzata verrà automaticamente sospesa in memoria dal sistema e messa in attesa in background finchè non viene richiesto di nuovo il suo utilizzo, in questo modo non è necessario chiudere e riaprire le applicazioni ogni volta e le applicazioni in background non consumeranno energia inutilmente; quando la memoria viene usata intensamente Android esegue un kill dei processi che sono stati inattivi da più tempo (oldest first).

Partendo dalla prima versione di Android (Android 1.0) rilasciata nel Settembre del 2008 oggi si è arrivati alla versione 4.1 (Jelly Bean), ogni versione tipicamente aggiusta bug della versione precedente ed aggiunge nuove funzionalità.

1.2.2 iOS

iOS (precedentemente iPhone OS) è un sistema operativo sviluppato da Apple per iPhone, iPod touch e iPad. Come Mac OS X è una derivazione di UNIX. A differenza di android e windows phone, Apple non ha dato la licenza di installare iOS su dispositivi con hardware non-Apple. Il sistema operativo è stato presentato il 9 gennaio 2007, e la versione 1.0, ancora priva di nome, è entrata in commercio con il primo iPhone il 29 giugno dello stesso anno.

Nel luglio del 2008 viene pubblicato in concomitanza della vendita di iPhone 3G l'aggiornamento a iPhone OS 2.0 che aggiunge, tra le altre funzioni, il molto atteso App Store e la possibilità di installare applicazioni di terze parti tramite l'app.

Il quarto rilascio del sistema operativo, pubblicato con iPhone 4 nel giugno 2010, ha aggiunto numerose funzioni quali il multitasking. L'ora rinominato "iOS", ha unificato i vari dispositivi (iPhone, iPod touch e iPad) con una versione comune, la 4.2.1.

Nel Giugno del 2011 è stata presentata al WWDC (Worldwide Developer Conference) la quinta versione di iOS con numerose nuove funzioni, tra cui l'integrazione con il servizio iCloud di Apple.

Oggi si è arrivati alla versione sei di iOS, che ha aggiunto numerose funzioni, tra le quali un’applicazione mappe completamente rinnovata e nuove lingue e funzioni per l’assistente vocale Siri

1.2.3 WINDOWS PHONE 7 E WINDOWS PHONE 8

Windows Phone 7 non è un aggiornamento dei precedenti sistemi operativi mobile Microsoft come Windows Mobile e Windows Phone 6.5. Microsoft ha deciso di basare il kernel di Windows Phone 7 su quello di Windows CE (Windows Emedded Compact, un sistema operativo Microsoft per dispositivi portatili), l'interfaccia utente è quella di zune (linea di prodotti multimediali Microsoft); le sue Api si basano su quelle di Win32 alle quali ne sono state aggiunte di nuove per interfacciare il sistema operativo con l'hardware mobile, i sensori, ecc..

La prima versione di Windows Phone 7 è stata rilasciata nell'ottobre del 2010, nei mesi seguenti Microsoft ha rilasciato un aggiornamento che aggiungeva il supporto per l'istruzione copia/incolla ed alcuni miglioramenti alle performances, ma l'aggiornamento più importante è stato quello rilasciato nel settembre del 2011: la versione 7.5 nel quale sono state apportati molti miglioramenti ed aggiunte nuove funzionalità (ad esempio: multy task per applicazioni di terze parti, supporto per i task in background ecc..) .

Windows Phone 8 è l'ultimo sistema operativo mobile di casa Microsoft, è stato rilasciato recentemente (29 ottobre 2012), il kernel è totalmente cambiato, il quale viene ereditato da quello del tablet desktop, in questo modo gli sviluppatori possono beneficiare di Api in comune per sviluppare applicazioni sia per smartphone, tablet e PC.

Windows Phone 8 toglie dei limiti vecchia versione 7, come la possibilità digestire finalmente display in HDReady, processori dual core e forse quad core oltre che la compatibilità per le schede microSD tramite appositi slot.

In quanto non è possibile installare Windows Phone 8 sui dispositivi che possiedono Windows Phone 7 come sistema operativo originario, la Mircrosoft rilascerà a breve un aggiornamento alla versione 7.8 per aggiungere alcune delle funzionalità di Windows Phone 8 alla versione 7.

1.3 APPLICAZIONI PER SMARTPHONE E APPLICATION STORES

La rapida crescita del mercato smartphone ha generato un'enorme domanda di applicazioni per dispositivi mobili. Queste applicazioni possono essere pre-installate sul dispositivo oppure possono essere scaricate dai diversi application market places. Un application market place, o application store, è una piattaforma digitale che ha l'obbiettivo di fornire software ai dispositivi mobili; solitamente ogni sistema operativo possiede il proprio market place (Google Play per Android, Windows Phone Store per Windows Phone 7, ecc..), ma si possono scaricare applicazioni anche da market places di terze parti come ad esempio Samsung Application Store o Amazon Appstore. Per lo sviluppo di queste applicazioni ogni proprietario di sistema operativo fornisce il proprio SDK (Software Developement Kit) che contiene gli strumenti necessari allo sviluppatore per scrivere testare e pubblicare applicazioni. Sotto vengono elencati gli application market places per i sistemi operativi analizzati precedentemente.

Google Play è l'application store sviluppato da Google per i dispositivi android, un'applicazione chiamata Play Store viene preinstallata sulla maggior parte dei dispositivi android e permette agli utenti di scaricare applicazioni pubblicate da sviluppatori di terze parti

oltre che a leggerne le recensioni e installarne gli aggiornamenti. Gli sviluppatori pagano un contributo di $25 per la registrazione allo store che gli permette di pubblicare le proprie applicazioni. Al momento sono presenti circa 700.000 applicazioni su Google Play.

L'App Store è la piattaforma digitale sviluppata dalla Apple Inc. per la distribuzione di applicazioni per iOS. Questo servizio permette agli utenti di cercare applicazioni e scaricarle dall'iTunes Store. A differenza di Google Play, il processo di pubblicazione di una applicazione sull'App Store è più lento ed ha più restrizioni: le applicazioni possono essere rifiutate se infrangono le regole imposte da Apple come, ad esempio, un eccessivo utilizzo della connessione telefonica o l'entrare in diretta competizione con i prodotti Apple. Gli sviluppatori devono pagare un contributo di $99 l'anno per poter pubblicare applicazioni sull'App Store.

Windows Phone Store è la piattaforma Microsoft che permette di scaricare applicazioni per dispositivi Windows Phone. Come per Apple gli sviluppatori devono pagare un contributo annuale di $99 per la pubblicazione di applicazioni, non c'è un limite al numero di applicazioni a pagamento che uno sviluppatore può pubblicare, mentre c'è un limite di 100 applicazioni gratuite pubblicate, superato questo limite è necessario pagare $19.99 per applicazione.

1.4 GEOLOCALIZZAZIONE

La maggior parte degli smartphone di ultima generazione è dotata di GPS, è possibile sfruttare questo strumento, oltre che alla connessione dati o quella Wi-Fi per geolocalizzare il telefono. La geolocalizzazione è l'identificazione della posizione geografica (latitudine e longitudine) di

una persona, di un oggetto in un dato momento. Con la diffusione dei sistemi GPS, degli smartphone e di internet la geolocalizzazione diventa importante in quanto comunicando la propria posizione è possibile ottenere ad esempio informazioni dalla rete su tutto quello che ci circonda. In rete tutti i maggiori social network utilizzano la posizione comunicata dall'utente per offire pubblicità mirata o servizi riferiti al luogo dove ci si trova. Come detto precedentemente i dispositivi mobili sfruttano GPS, Wi-Fi oppure la rete GSM per la geolocalizzazione del dispositivo.

- Geolocalizzazione tramite GPS (Global Positioning System)

La localizzazione tramite GPS viene effettuta sfruttando la ricezione di segnali provenienti da vari satelliti che si trovano in orbita: Il principio del GPS è abbastanza semplice: attraverso i segnali ricevuti dai satelliti in orbita (almeno 4) il dispositivo riesce a calcolare la propria posizione conoscendo il tempo di volo del segnali, la mappa dei satelliti e la loro sincronizzazione temporale. Il suo grande vantaggio è la precisione che può arrivare al metro. Lo svantaggio di questa tecnologia è che non può funzionare dentro agli edifici.

- Geolocalizzazione tramite rete GSM

Il processo di localizzazione si basa sulla possibilità del GSM di conoscere la distanza approssimativa dispositivo collegato alla rete GSM dalla stazione radio base con la quale é connesso. Ripetendo in sequenza l'operazione di stima da più basi di trasmissione circostanti, ed effettuando alcune operazioni di triangolazione matematica é possibile stimare con buona precisione la posizione sul territorio del telefonino.

- Geolocalizzazione tramite rete Wi-Fi

Questo sistema di geolocalizzazione sfrutta la presenza di access point WiFi sparsi nel territorio: conoscendo l'esatta ubicazione degli access point ricevuti dal proprio terminale, è possibile determinare la posizione con un'accuratezza di poche decine di metri. Questo sistema è chiamato anche WPS (*Wireless Positioning System*).

- Geolocalizzazione tramite rete internet

Questo sistema sfrutta il solo indirizzo IP per rilevare la posizione. Ogni provider ha un range di indirizzi IP a sua volta suddiviso in sottogruppi di indirizzi IP che servono determinate regioni e province. Pertanto conoscendo un indirizzo IP possiamo visualizzare l'area approssimativa del dispositivo connesso ad internet**.**

2 FASE PROGETTUALE

2.1 DESCRIZIONE DEL PROGETTO

Il progetto sviluppato e presentato in questa tesi prende il nome di Geo Smart Social Game (Non Definitivo).

L'idea del progetto è quella di sviluppare un gioco per smartphone dove l'utente deve prendersi cura di un proprio personaggio virtuale, accudendolo e facendogli svolgere tutte le azioni della sua vita quotidiana: come nutrirsi, dormire, giocare e curarsi per farlo crescere e vivere il più a lungo possibile.

Sfruttando la geolocalizzazione è possibile ottenere dei bonus per ogni azione svolta: se l'utente decide di far eseguire al personaggio una certa azione in prossimità di un punto di interesse valido per quest'utlima, (ad esempio, un ristorante per il nutrimento o un ospedale per la cura del personaggio) si otterranno dei bonus aggiuntivi come un recupero maggiore di salute o incremento di felicità addizionale.

L'applicazione quindi, utilizzando i servizi di geolocalizzazione ottiene la posizione attuale dell'utente ed in seguito esegue una ricerca dei punti di interesse che possono essere sfruttati per ottenere bonus in un range di poche decine di metri dalla posizione dell'utente; è comunque possibile far eseguire azioni al personaggio senza sfruttare i servizi di geolocalizzazione perdendo però tali bonus.

Bisognerà quindi essere sufficientemente vicini ad un punto di interesse per ottenere il relativo bonus. Attraverso una mappa l'applicazione da modo all'utente di visualizzare una lista dei punti di interesse in un'area di qualche kilometro dalla sua posizione attuale (visualizzati come pushpin sulla mappa), permettendogli di avvicinarsi fisicamente a questi ultimi per ottenere i vari bonus.

Dopo aver creato il suo personaggio, l'utente può visualizzare nel menù delle statistiche tutte le informazioni relative al personaggio creato come: i livelli di salute, sazietà, felicità, forma fisica e sonno e di conseguenza decidere quali azione fargli svolgere per farlo crescere nel modo corretto.

L'applicazione gestisce il decremento di queste statistiche nel tempo: il decremento è più rapido nei periodi di tempo in cui l'applicazione è attiva, cioè quando l'utente la sta usando, mentre è molto più lento nei periodi di inattività, in questo modo l'utente non viene troppo penalizzato dai periodi in cui è impossibilitato ad usare il dispositivo come, ad esempio, durante il sonno. Se durante il tempo in cui l'utente non sta usando l'applicazione qualche statistica raggiunge un livello basso, l'applicazione lo notificherà all'utente facendo utilizzo di una notifica push.

Se trascurato, il personaggio può abbandonare l'utente o addirittura morire, i livelli di felicità e salute determinano questi due fattori: se il livello di felicità o quello di salute raggiungono lo zero il personaggio rispettivamente se ne andrà oppure morirà, in entrambi i casi l'utente sarà costretto ad iniziare una nuova partita.

2.2 MECCANICHE DI GIOCO

Riposo: Periodicamente il proprio personaggio avrà bisogno di riposarsi, l'utente dovrà quindi metterlo a dormire per un certo lasso di tempo, durante il quale non potrà svolgere nessun'altra azione. L'azione del far riposare il proprio personaggio influisce sul livello di sonno, decrementandolo. Col passare del tempo il livello del sonno del proprio personaggio aumenterà. Un livello di sonno troppo alto porta ad una graduale diminuzione della forma fisica, e ad un conseguente aumento della probabilità che il proprio personaggio possa contrarre una malattia. Il bonus a questa azione si ottiene nei pressi di Hotel e consiste in un maggior decremento del livello di sonno. Inoltre, è necessario far riposare il personaggio per fargli ripristinare parte del suo livello di energia per fargli compiere altre azioni.

Nutrimento: Se il livello di appetito del personaggio è alto, è necessario nutrirlo. Ogni personaggio possiede dei cibi preferiti ed altri che non sono di suo gradimento; ogni volta che l'utente decide di nutrire il suo personaggio potrà scegliere quali cibi fargli mangiare tra quelli generati casualmente dall'applicazione (vengono generati da 2 a 3 cibi se non si fa uso della localizzazione, mentre se l'utente si trova nei pressi di un ristorante ne possono essere generati da 4 a 5). Ogni cibo, indipendentemente dal fatto che sia di gradimento o meno dal personaggio, aumenta il livello di sazietà, e decrementa, anche se di poco, quello della forma fisica, quindi il far mangiare troppo frequentemente il proprio personaggio può decrementare di molto la sua forma fisica ed aumentare il rischio che contragga una malattia. Nutrire il proprio personaggio con cibi di suo gradimento aumenta il suo livello

di felicità, al contrario nutrirlo con cibi non di suo gradimento abbassa il livello di felicità.

Gioco: L'utente può far giocare il proprio personaggio per aumentare il suo livello di felicità e la sua forma fisica. Questa azione viene svolta con attraverso un mini-game nel quale l'utente dovrà spostare il proprio personaggio nello schermo facendo uso dell'accelerometro per farlo muovere e fargli raccogliere quante più sfere possibile prima che tocchino terra. Il trovarsi nei pressi di un parco offre un bonus a questa azione: i punti felicità ottenuti alla fine del mini-gioco vengono raddoppiati, ed il mini-gioco ha una durata maggiore. Il livello di felicità incrementerà di un valore pari al numero di sfere raccolte diviso per 5.

Malattia: Se l'icona  è presente nella barra dell'interfaccia di gioco, l'utente può capire che il suo personaggio è malato. Finchè il personaggio rimane malato, il suo livello di salute decrementerà nel tempo.

Cura: E' possibile curare il personaggio una volta al giorno, se il personaggio è malato l'azione di cura rimuoverà la malattia, se il personaggio non è malato l'azione di cura aumenterà il suo livello di salute. Il bonus che si può ottenere da questa azione sfruttando la geolocalizzazione è un ulteriore aumento del livello di salute (se il personaggio è malato l'azione di cura oltre al rimuovere la malattia aumenta anche il livello di salute). Per poter usufruire del bonus, l'utente dovrà trovarsi nei pressi di un ospedale, di un dottore o un dentista.

Forma fisica: Il livello di forma fisica influisce negativamente o positivamente sulla probabilità di contrarre malattie. La probabilità che il proprio personaggio contragga una malattia aumenta o diminuisce in

base al fatto che la forma fisica del proprio personaggio diminuisca o aumenti rispettivamente.

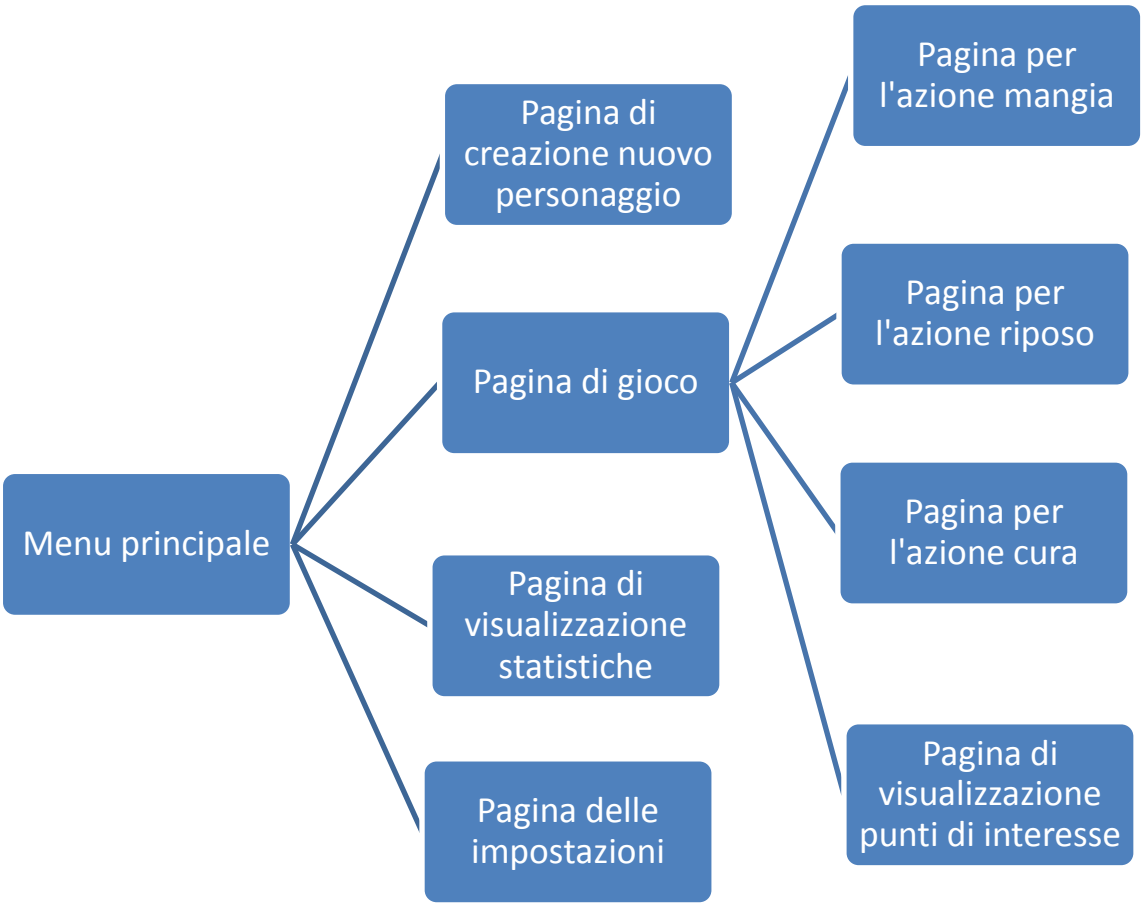
Energia: l'esecuzione di ogni azione tranne quella del riposo, consuma parte del livello di energia del personaggio, è necessario far riposare il personaggio per ripristinare la sua energia. Se il personaggio è a riposo, la sua energia aumenta gradualmente nel tempo. Con l'aumentare del livello del personaggio, il livello massimo di energia che può essere raggiunto aumenterà.

Livello: il livello indica lo stato della crescita del personaggio, più è alto più l'utente si è preso cura del suo personaggio. Il livello aumenta in base all'esperienza, per salire al livello successivo, il personaggio necessita di un certo quantitativo di esperienza. Più il livello del personaggio è alto, più l'esperienza necessaria al raggiungimento del livello successivo sarà alta. Ogni volta che il livello del personaggio aumenta, vengono aumentati i livelli massimi di tutte le altre statistiche.

Esperienza: L'esperienza influisce sull'aumento del livello del personaggio. Il personaggio può guadagnare esperienza attraverso l'azione lavoro. Il lavoro del tamagotchi consiste nello scattare foto, ogni volta che l'utente attraverso lo svolgimento dell'azione lavoro, scatta una foto, il tamagotchi guadagna esperienza. Se la foto viene scattata nei pressi di un monumento o di una chiesa l'esperienza guadagnata dal tamagotchi è maggiore.

2.3 PAGINE DELL'APPLICAZIONE

La struttura delle pagine dell'applicazione è la seguente:



All'avvio dell'applicazione l'utente viene indirizzato alla pagina del menù principale, questa è la pagina principale, attraverso ad essa l'utente può raggiungere tutte le altre pagine.

Pagina di creazione di un nuovo personaggio: in questa pagina l'utente può selezionare da una lista il personaggio che vorrà utilizzare durante la

partita e dargli un nome. Il personaggio verrà quindi creato e le sue statistiche inizializzate a dei valori default.

Pagina di visualizzazione delle statistiche: questa è la pagina per la visualizzazione delle statistiche del proprio personaggio, nel caso in cui l'utente non abbia creato ancora il proprio personaggio, questa pagina verrà nascosta.

Pagina delle impostazioni: Attraverso questa pagina l'utente può decidere se dare all'applicazione i permessi per utilizzare i servizi di geolocalizzazione e sceglierne la precisione (può decidere tra precisione standard oppure alta).

Pagina di gioco: In questa pagina rappresenta l'interfaccia di gioco principale, l'utente può far muovere il proprio personaggio sul background e può fargli compiere le azioni che ritiene necessarie, può visualizzare anche lo stato della connettività e disattivare i servizi di geolocalizzazione. Dopo aver deciso l'azione da far compiere al proprio personaggio, l'utente viene reindirizzato alla pagina corrispondete. La pagina di gioco, inoltre, si occupa anche della gestione del mini-game relativo all'azione gioco.

Pagina di visualizzazione dei punti di interesse: In questa pagina l'utente può visualizzare sia su di una mappa, che elencati in una lista, tutti i punti di interesse utili all'ottenimento dei bonus del gioco, in un'area di due kilometri dalla sua posizione attuale. Quando la pagina viene caricata, la mappa viene centrata sulla posizione attuale dell'utente e vengono creati i pushpin in corrispondenti alle coordinate dei punti di interesse trovati. Eseguendo un tap su un punto di interesse presente nella lista, la mappa si centrerà sul pushpin corrispondente.

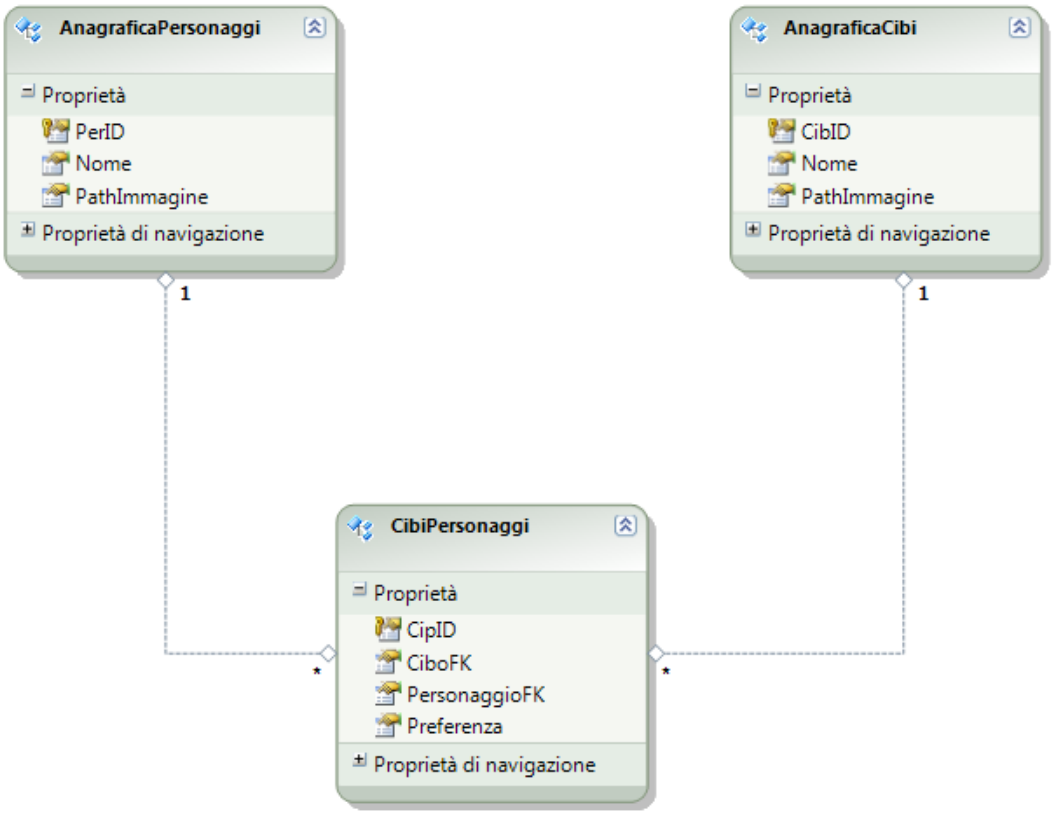
Pagina relativa all'azione mangia: attraverso questa pagina l'utente può nutrire il suo personaggio per diminuire il suo livello di appetito. Quando la pagina viene caricata, vengono visualizzati i cibi generati in modo casale. Prima di eseguire questa generazione, l'applicazione esegue un controllo sulla posizione dell'utente: se quest'ultimo è nei pressi di un ristorante vengono generati dai quattro ai cinque cibi, altrimenti dai due ai tre. Una volta selezionati i cibi, per farli consumare al proprio personaggio l'utente dovrà scuotere il dispositivo fino al riempimento di una progressbar.

Pagina relativa all'azione cura: attraverso questa pagina l'utente può curare il suo personaggio e ripristinare il suo livello di salute. L'utente ha a disposizione tre tentativi per curare o aumentare il livello di salute del suo personaggio. Durante ogni tentativo l'utente dovrà premere nell'ordine giusto una combinazione di tasti ed eseguire gesture, in una certa unità di tempo. Il numero di punti salute ripristinati dipendono dalla parte di combinazione svolta correttamente dall'utente (da un minimo di uno ad un massimo di 5 punti salute ripristinati).

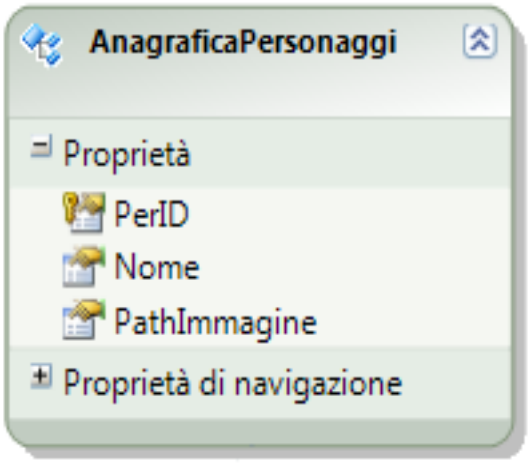
Pagina relativa all'azione riposo: in questa pagina l'utente può far riposare il suo personaggio. Il personaggio riposerà per qualche minuto e durante questo tempo sarà impossibile fargli svolgere qualsiasi altra azione.

2.4 SPECIFICHE DI PROGETTAZIONE

Nella pagina di creazione del personaggio l'utente può scegliere quale personaggio voler allevare. Ogni personaggio si distingue da un altro per aspetto (immagine del personaggio) e per preferenze sui cibi. L'anagrafica dei personaggi disponibili, quella dei cibi, e le relazioni che le legano dovranno quindi essere persistenti in memoria. Per far ciò l'applicazione fa utilizzo di un database locale con la seguente struttura:



In seguito viene riportata una breve descrizione delle tabelle che lo compongono:



Descrizione dei campi:

- PerID: chiave primaria della tabella, campo intero auto-incrementante
- Nome: campo di tipo stringa che descrive il nome originale del personaggio
- PathImmagine: campo di tipo stringa che descrive il percorso relativo dell'immagine, il nome dell'immagine è composto in tal modo: valore del campo Nome + estensione dell'immagine. Es: NomeImmagine.png.

Descrizione dei campi:

La tabella è strutturata pressoché nello stesso modo della tabella precedente.

Descrizione dei campi:

- CipID: chiave primaria della tabella, campo intero auto-incrementante
- CiboFK; chiave esterna che fa riferimento al campo CibID della tabella AnagraficaCibi
- PersonaggioFK: chiave estena che fa riferimento al campo PerID della tabella AnagraficaPersonaggi
- Preferenza: campo intero che può assumere tre valori e che descrive la preferenza o meno del personaggio indicato dal campo PersonaggioFK nei confronti del cibo indicato dal campo CiboFK.
  Valore 0: il personaggio è indifferente a quel cibo (nessun cambiamento nel livello di felicità se mangiato da quel personaggio).
  Valore 1: il personaggio preferisce quel cibo (aumento del livello di felicità se mangiato da quel personaggio).
  Valore -1: il personaggio sfavorisce quel cibo (diminuzione del livello di felicità se mangiato da quel personaggio).

Gli altri dati che necessitano di essere persistenti in memoria sono le impostazioni relative ai permessi per l'utilizzo della geolocalizzazione e la relativa precisione e tutte le statistiche del personaggio.

3 IMPLEMENTAZIONE DELL'APPLICAZIONE

3.1 PERCHE' SVILUPPARE PER WINDOWS PHONE

La scelta del sistema operativo sul quale sviluppare questo progetto è ricaduta su Windows Phone Mango (7.5).

Le motivazioni che mi hanno spinto alla scelta di sviluppare il progetto per questo sistema operativo sono varie, partendo da quelle di carattere personale, di sicuro, il voler imparare a progettare un'applicazione per Windows Phone, cosa che non ho avuto modo di fare durante le lezioni, ha influito sulla scelta, inoltre grazie all'ottimo emulatore messo a disposizione dall'SDK di Windows Phone, ho potuto eseguire i debug del progetto testandolo in ogni sua parte (l'emulatore simula tra le altre cose, anche la lettura di dati da accelerometro e dai servizi di location providing) anche senza poterlo provare direttamente su di un dispositivo.

Passando ad altri motivi, il fatto che l'application market place di Windows Phone possiede un mercato di applicazioni più ristretto rispetto ai corrispondenti iOS ed Android è sicuramente un punto a favore per gli sviluppatori che vogliono pubblicare i loro progetti, in quanto l'avere un'idea per lo

sviluppo di un’applicazione che ancora non è presente nello store, è senza dubbio più semplice.

Inoltre se le previsioni future per la crescita della quota di mercato di Windows Phone sono corrette, da qui a quattro anni i dispositivi che faranno utilizzo di questo sistema operativo saranno notevolmente più diffusi.

3.2 LINGUAGGI UTILIZZATI ED AMBIENTE DI SVILUPPO

Per sviluppare un’applicazione per windows phone 7.5 è necessario scaricare l’SDK 7.1 fornita da Microsoft, i principali contenuti di quest’ultima sono: Visual Studio Express per Windows Phone (una volta scaricata ed installata l’SDK è comunque possibile utilizzare un’altra versione di visual studio per lo sviluppo dell’applicazione), un emulatore attraverso il quale è possibile eseguire un debug del progetto anche senza possedere un dispositivo vero e proprio, Microsoft Expression Blend ed altro ancora…

Visual studio mette a disposizione svariate tipologie di progetto da cui lo sviluppatore può iniziare, per lo sviluppo di questa applicazione ho deciso di partire dal template di progetto XNA

e Silverlight per Windows Phone, questo template integra le funzionalità dell'XNA Game Framework con quelle di Silverlight, la pagina di gioco ed il mini-game sono state sviluppate facendo un uso combinato delle funzionalità offerte dal Game Framework XNA (per gestire il ciclo di gioco in maniera più intuitiva) e di Silverlight (per la creazione di pulsanti nell'interfaccia di gioco), le restanti pagine sono create facendo un utilizzo esclusivo di Silverlight.

Nelle applicazioni Windows Phone, lo sviluppo dell'interfaccia utente viene eseguito utilizzando XAML, il linguaggio dichiarativo di Microsoft, mentre tutta la parte di code-behind è scritta in linguaggio C#.

3.3 PROGETTAZIONE DELLE PARTI FONDAMENTALI

3.3.1 GEOLOCALIZZAZIONE

Per eseguire la ricerca dei punti di interesse utili all'ottenimento dei bonus, in primo luogo è stato necessario ricavare le coordinate geografiche relative alla posizione attuale dell'utente.

Per far ciò Windows Phone mette a disposizione la classe GeoCoordinateWatcher che fornisce tutte le funzionalità necessarie allo sviluppatore per utilizzare i servizi di location providing del dispositivo (GPS, Wi-Fi, ecc..). La classe GeoCoordinateWatcher astrae la scelta del location provider utilizzato per l'acquisizione dei dati, permettendo allo sviluppatore di poter decidere solo la precisione che vuole utilizzare, che può essere scelta tra High oppure Default (utilizzando una precisione alta si aumenta il consumo di batteria del dispositivo e l'acquisizione di dati dal location provider può richiedere più tempo). Come detto precedentemente attraverso la pagina delle impostazioni l'utente può modificare tale precisione.

Una volta ricavata la posizione dell'utente attraverso l'evento PositionChanged (viene scatenato quando il location provider rileva un cambiamento di posizione di almeno 20 metri, cioè

del valore di MovementThresold), l'applicazione deve ricavare i nomi e le coordinate geografiche dei punti di interesse vicini all'utente. Questa ricerca viene eseguita facendo uso delle api di Google Places; la decisione di utilizzare queste ultime piuttosto che i servizi di ricerca di Bing deriva dal fatto che, eseguendo test con entrambi, con le Api di Google Places ho ottenuto risultati più precisi e in quantità maggiore, inoltre la struttura della richiesta per la ricerca di luoghi di Google Places è perfetta per lo scopo dell'applicazione.

La richiesta per i luoghi desiderati deve essere eseguita mediante una richiesta http all'url seguente:

https://maps.googleapis.com/maps/api/place/nearbysearch/*output*?*parameters*

Passando nella query-string i parametri:

- output: Tipo di output (Json o Xml)
- key: Api Key ottenuta precedentemente visitando l'api console e creando un nuovo Api Project
- location: latitudine\longitudine della posizione attuale dell'utente
- radius: range in metri nel quale eseguire la ricerca dal punto centrale
- sensor: se la richiesta arriva da un device che fa utilizzo o meno di servizi di location providing
- type: tipologia del punto di interesse desiderato tra quelli selezionabili di Google Places, Google Places divide questi punti di interesse in varie categorie, questa applicazione fa utilizzo delle

seguenti: food (per il bonus relativo all'azione mangia), hospital (per il bonus relativo all'azione cura), park (per il bonus relativo all'azione gioca), "museum, art_gallery e church"(per il bonus relativo all'azione fotograga).

- Keyword: termine utilizzato da google per eseguire una ricerca tra tutti i luoghi da lui indicizzati, nell'applicazione viene fatto uso di questo campo solo nel caso della ricerca della parola chiave "hotel" (per il bonus relativo all'azione riposo), in quanto non ho trovato nessuna categoria tra quelle offerte da Google Places adatta alla ricerca di tali punti di interesse.

Il seguente codice si occupa della creazione dell'URL, dell'esecuzione della richiesta e della gestione della risposta.

```
//fa la richiesta http a google places con i parametri passati in ingresso e aggiunge l'handler per la gestione della risposta
private void HttpRequestToGooglePlaces(double latitudine, double longitudine, double range, string chiaveDiRicerca)
{
    string urlRichiesta = BuildURLGoogleApi(latitudine, longitudine, range, chiaveDiRicerca);
    WebClient clientWeb = new WebClient();
    clientWeb.DownloadStringAsync(new Uri(urlRichiesta));
    clientWeb.DownloadStringCompleted += new DownloadStringCompletedEventHandler(googlePlacesRequestEnded);
}
```

```
//Viene invocato quando ritorna una risposta per una richiesta http alle google places api
void googlePlacesRequestEnded(object sender, DownloadStringCompletedEventArgs e)
{
    string s = e.Result;
    GoogleApisPlaceResponse APIResponse = JsonConvert.DeserializeObject<GoogleApisPlaceResponse>(s);
    var prova = sender;
    //aggiunge tutti i luoghi trovati alla lista
    foreach (LuogoDiInteresse el in APIResponse.results)
    {
        if (el != null)
        {
            if (el.types.Contains("hotel"))
            {
                (App.Current as App).hotels.Add(el);
                (App.Current as App).geoCibo = true;
            }
            if (el.types.Contains("food"))
            {
                (App.Current as App).ristoranti.Add(el);
                (App.Current as App).geoCibo = true;
            }
            if (el.types.Contains("museum") || el.types.Contains("art_gallery") || el.types.Contains("church"))
            {
                (App.Current as App).palestre.Add(el);
                (App.Current as App).geoCibo = true;
            }
            if (el.types.Contains("park"))
            {
                (App.Current as App).hotels.Add(el);
                (App.Current as App).geoCibo = true;
            }
            if (el.types.Contains("hospital"))
            {
                (App.Current as App).ospedali.Add(el);
                (App.Current as App).geoOspedale = true;
            }
        }
    }
}
```

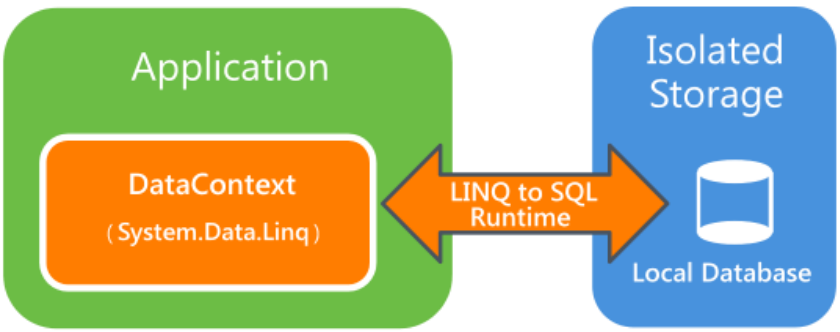
L’event-handler che gestisce la risposta si occupa anche di eseguire un parsing del file Json ottenuto dal server per ottenere una lista luoghi di interesse. Eseguendo un controllo sul campo type di ogni elemento della lista l’applicazione stabilisce che tipo di punto di interesse è stato trovato.

Sia la localizzazione della posizione attuale dell’utente che la gestione di richieste e risposte delle API di Google Places sono gestite in un thread separato da quello principale dell’applicazione, che viene lanciato solo se l’utente ha dato i permessi all’utilizzo dei servizi di geolocalizzazione, la scelta di far esguire queste operazioni in un thread secondario influenza in maniera positiva i tempi di caricamento delle pagine (ad esempio, la pagina relativa alla visualizzazione dei punti di interesse può caricare sia i pushpin che tutti gli elementi della lista senza dover

eseguire la ricerca dei punti di interesse, perché questa è già stata eseguita dal backGroundWorker).

3.3.2 ORGANIZZAZIONE DEI DATI E GESTIONE DI AGGIORNAMENTI E NOTIFICHE

Con Windows Phone 7.1, Microsoft propone una soluzione per la gestione di database locali basata sulla versione di SQL Compact per Windows Phone. Il database è contenuto nell'Isolated Storage (area di archiviazione riservata all'applicazione e identificata dall'Application Guid), il che significa che l'accesso ai dati è riservato unicamente all'applicazione che lo ha creato. L'approccio supportato da Windows Phone 7.1 è chiamato **Code-First**: il mapping tra il database e gli oggetti viene fatto direttamente da codice. Al primo avvio, l'applicazione crea il database in base a come sono state definite le classi corrispondenti alle tabelli ed al data context.



Ho fatto utilizzo di SQL Server Management Studio per la creazione del file del database, in seguito, attraverso Visual Studio, ho creato una nuova connessione dati a tale file per il progetto.

Grazie ad SQL Server Compact Toolbox (estensione di Visual Studio) ho creato il Data Context ed una classe per ogni tabella del database.

I dati da inserire nelle tabelle delle anagrafiche e in quella della relazione tra di esse sono contenuti in tre file di testo, al primo avvio, dopo la creazione del database, l'applicazione legge i file di testo e per ogni riga analizzata, un nuovo record viene inserito in una tabella.

Esempio della struttura di una riga del file di testo per le relazioni:

```
personaggio1,+torta,+crostata,-gelato
```

I nomi dei personaggi e quelli degli alimenti, anche se non corrispondono alle chiavi primarie delle relative tabelle sono valori univoci all'interno di esse. Il carattere '+' o quello '-' davanti al nome di un alimento indica che per il personaggio presente nella riga rispettivamente favorisce o sfavorisce quell'alimento. Una volta ricavate le chiavi primarie corrispondenti al nome del personaggio ed a quello dell'alimento è possibile creare un nuovo record per la tabella di relazione Cibi-Personaggi (in questo caso viene inserito un record per ogni alimento presente nella riga).
Per la memorizzazione dei dati relativi alle impostazione ed alle statistiche attuali del personaggio ho preferito non utilizzare il database, sia per il fatto che sarebbe stato necessario creare due tabelle (una per le impostazioni e l'altra per le statistiche) che contenevano un unico record, ma soprattutto perché, dovendo i dati relativi alle statistiche essere letti e modificati da più processi, la memorizzazione di questi come coppie

chiave-valore all'interno dell'isolatedStorageSettings rende tale accesso più semplice.

L'aggiornamento delle statistiche del personaggio viene eseguito in due maniere differenti: se l'applicazione è in uso dall'utente, questi valori vengono aggiornati frequentemente. Ho deciso di effettuare questi aggiornamenti in un thread separato da quello dell'applicazione, in modo tale che siano indipendente dalle operazioni che l'utente esegue. Ho utilizzato la classe BackGroundWorker che si presta perfettamente allo scopo:

```
BackgroundWorker threadAggiornamento = new BackgroundWorker();
threadAggiornamento.DoWork += new DoWorkEventHandler(AggiornaStatistiche);
if(!threadAggiornamento.IsBusy)
    threadAggiornamento.RunWorkerAsync();
```

L'event-handler AggiornaStatistiche relativo all'evento DoWorkEventHandler si occupa dell'aggiornamento delle statistiche. Il controllo if si occupa di non eseguire una seconda volta il thread se quest'ultimo è già stato lanciato. Dopo aver aggiornato le statistiche il thread verrà bloccato con il metodo .sleep() per il tempo che deve trascorrere tra un aggiornamento e l'altro (pochi minuti).

Quando l'applicazione è però, potrebbero sorgere problemi relativi alla correttezza dei dati in quanto sia il thread che si occupa dell'aggiornamento delle statistiche, che quello principale dell'applicazione accedono all'isolated storage per eseguire delle operazioni di lettura e scrittura sui dati.

Ad esempio:

Il valore iniziale di livello salute è 21:

Thread per la gestione degli aggiornementi	Thread principale dell'applicazione
Legge livello di salute (21)	Legge livello di salute (21)
Livello di salute = Livello di salute - 1	Livello di salute = Livello di salute + 5
	Scrittura per livello di salute (26)
Scrittura per livello di salute (20)	

In questo caso l'aggiornamento del valore eseguito dal thread principale andrebbe perso se l'accesso all'isolated storage non fosse sincronizzato.
Per ovviare a questo problema è stata utilizzata la classe Mutex.
Questa classe garantisce l'accesso ad una risorsa condivisa a solo un thread per volta. Se il primo thread prende possesso del mutex, il secondo non può accedere alla risorsa condivisa finchè il primo non esegue tutte le operazioni su di essa ed infine libera il mutex.
La classe personaggio gestisce tutti gli accessi alle letture e alle scritture delle statistiche del personaggio memorizzate sull'Isolated Storage.
La seguente porzione di codice illustra l'implementazione dei metodi per la lettura e la scrittura di una statistica nell'application storage della classe Personaggio:

```
public object GetValue(string key)
{
    Mutex mut = new Mutex(false,"isolated storage");
    object obj = new object();
    mut.WaitOne();
    if (IsolatedStorageSettings.ApplicationSettings.Contains(key))
        obj = IsolatedStorageSettings.ApplicationSettings[key];
    else
        obj = null;
    mut.ReleaseMutex();
    return obj;
}

public void SetValue(string key, string value)
{
    Mutex mut = new Mutex(false, "isolated storage");
    mut.WaitOne();
    if (IsolatedStorageSettings.ApplicationSettings.Contains(key))
        IsolatedStorageSettings.ApplicationSettings[key] = value;
    mut.ReleaseMutex();
}
```

Il parametro key è presente in entrambi i metodi, esso rappresenta la stringa che identifica la statistica che si vuole leggere o scrivere dall'isolated storage.

Come si osserva dal codice i due oggetti della classe Mutex sono stati inizializzati con lo stesso nome "isolated storage", in questo modo l'accesso ai dati dell'isolatied storage per la lettura e scrittura delle statistiche del personaggio avviene in maniera esclusiva e si evitano problemi di sincronizzazione.

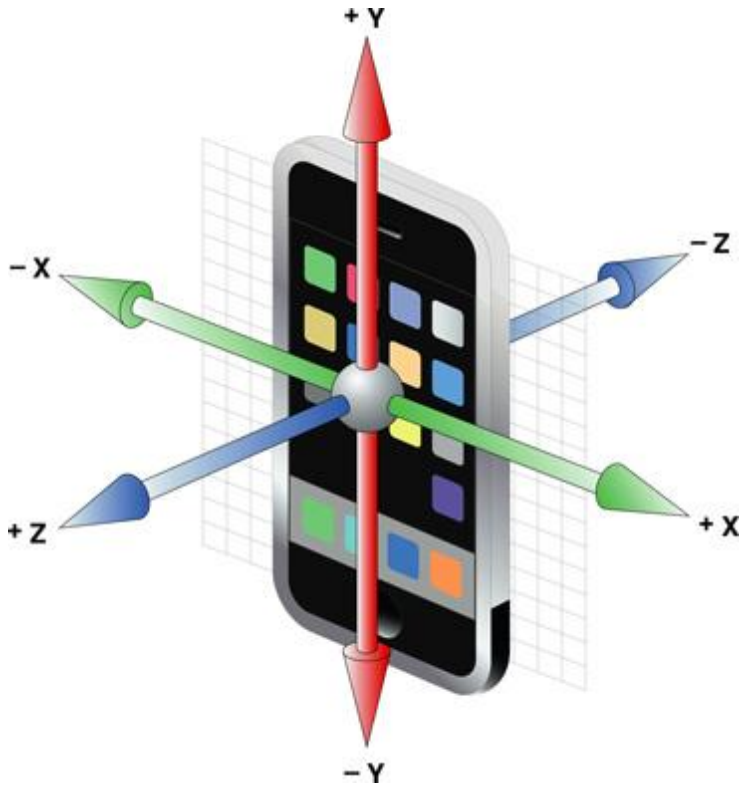
Se l'applicazione invece non è in esecuzione, le statistiche devono comunque venire costantemente aggiornate anche se il lasso di tempo che trascorre tra gli aggiornamenti è più ampio. Windows Phone 7.5 introduce il supporto per i background agents, questi ultimi permettono l'esecuzione di codice in un thread separato dall'applicazione quando quest'ultima non è in esecuzione e possono essere di due tipi: PeriodicTask oppure ResourceIntensiveTask, il tipo utilizzato da questa applicazione è PeriodicTask: il background agent viene richiamato circa ogni 30 minuti (questo lasso di tempo non può essere modificato), può eseguire operazioni che non richiedono un consumo di memoria ingente ed accedere allo stesso isolated storage dell'applicazione, questo tipo di

background agent si presta perfettamente allo scopo dell'aggiornamento offline ed a quello dell'invio di notifiche. Ho aggiunto al soluzione del progetto sottoprogetto con template Windows Phone Scheduled Task Agent, il codice interno alla chiamata OnInvoke() si occupa dell'esecuzione dell'aggiornamento delle statistiche e dell'invio di notifica push se una di tali statistiche è bassa (il livello dell'esperienza non viene controllato).

3.3.3 IMPLEMENTAZIONE DELLE PARTI RELATIVE ALL'ACCELETOMETRO

Le pagine dell'applicazione che fanno utilizzo della lettura di dati dal sensore dell'accelerometro sono due: la pagina relativa all'azione mangia e la pagina dell'interfaccia principale di gioco durante il mini-game.

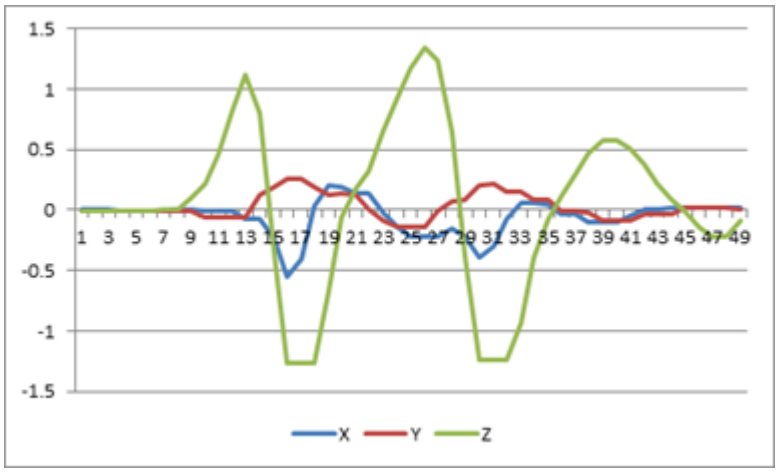
Verrà brevemente illustrato il sistema di coordinate utilizzato da Windows Phone per rappresentare le misure. La classe AccelerometerReadingEventArgs fornisce i dati ogni qual volta l'evento ReadingChanged è scatenato, essa contiene tre proprietà X, Y e Z, che rappresentano i valori dell'accelerazione nelle rispettive direzioni, con in particolare con l'asse Z orientato nel verso uscente dallo schermo.



La pagina relativa all'azione mangia fa utilizzo dei dati letti dal sensore dell'accelerometro per rilevare quando l'utente esegue una gesture "Shake" (ovvero quando scuote il dispositivo).
La gesture Shake è un movimento continuo nello spazio 3D attraverso uno o più assi, la cui direzione subisce parecchi cambiamenti nel tempo, essa viene rilevata se l'utente scuote il dispositivo in una o più direzioni, I seguenti parametri sono utili al riconoscimento della gesture:

- Il movimento deve essere continuo
- La forza con la quale il dispositivo viene scosso è significativa (se non supera una certa soglia la gesture non viene rilevata)

Esempio di lettura di dati utile al riconoscimento di una gesture Shake (asse Z):



Per il rilevamento di questa gesture l'applicazione esegue un controllo sui valori letti in input dall'accelerometro.

In Windows Phone, la classe Accelerometer fornisce all'applicazione l'accesso ai dati letti dall'accelerometro, ogni volta che l'evento ReadingChanged viene scatenato, una nuova lettura viene eseguita, il tempo che intercorre tra due letture è definito dalla proprietà TimeBetweenUpdate (in questa applicazione viene utilizzato quello di default che è 2 ms).

L'applicazione memorizza i valori dei parametri X,Y e Z ricavati dalla lettura precedente delle proprietà di AccelerometerReadingEventArgs e li confronta con quelli della lettura attuale. Per controllare se la forza con la quale il dispositivo viene scosso è sufficiente, l'algoritmo per l'identificazione della gesture esegue la differenza in valore assoluto tra la lettura attuale di un parametro e quella precedente (deltaX, deltaY, deltaZ), se almeno una di queste differenze supera una certa soglia (sogliaScuotimento) vorrà dire che l'utente ha scosso il dispositivo con forza sufficiente. Più il valore di questa soglia è alto, più l'utente dovrà scuotere velocemente il dispositivo. Ecco la porzione di codice che esegue l'identificazione della gesture:

```
private bool ControllaShake(Vector3 letturaCorrente, Vector3 letturaPrecedente)
{
    bool shake = false;
    double deltaX = Math.Abs((letturaPrecedente.X - letturaCorrente.X));
    double deltaY = Math.Abs((letturaPrecedente.Y - letturaCorrente.Y));
    double deltaZ = Math.Abs((letturaPrecedente.Z - letturaCorrente.Z));
    if (deltaX > sogliaScuotimento)
        shake = true;
    if (deltaY > sogliaScuotimento)
        shake = true;
    if (deltaZ > sogliaScuotimento)
        shake = true;

    return shake;
}
```

Anche la pagina di gioco sfrutta la lettura di dati dai sensori dell'accelerometro. Questa pagina fa un utilizzo combinato delle caratteristiche del frame work XNA e dell'accelerometro per implementare il mini-game relativo all'azione di gioco. L'evoluzione dell'interfaccia di gioco nel tempo pagina viene gestita da un loop che ad intervalli regolari (33 volte al secondo) esegue il codice che dovrà disegnare sullo schermo.

L’interfaccia del mini-game è la seguente:

Mantenendo il dispositivo in modalità landscape ed inclinandolo verso destra o verso sinistra, l’utente potrà così spostare il proprio personaggio verso destra o verso sinistra per cercare di raccogliere quanti più cibi possibili prima che tocchino terra.

Lo spostamento dell’immagine del personaggio (sia la velocità con la quale cambia posizione, sia l’accelerazione) dipende quindi dall’inclinazione che l’utente da al proprio dispositivo. Durante l’esecuzione del loop di gioco, l’Event-Handler OnReadingChanged utilizza i valori dell’asse Y dell’accelerometro e del tempo trascorso tra due letture per il calcolo dell’accelerazione, della velocità e della nuova posizione del personaggio. In primo luogo viene calcolata l’accelerazione del personaggio tenendo conto anche di quella calcolata nella chiamata precedente, per il calcolo dell’accelerazione viene utilizzata la formula dell’accelerazione di un corpo su di un piano inclinato senza attrito: a = g*sin(t) dove g è un parametro costante e sin(t) è il seno dell’angolo di inclinazione del dispositivo, cioè

AccelerometerReadingEventArgs.Y (valore negativo in caso di inclinazione verso destra).

In seguito viene calcolato il tempo trascorso dall'ultima lettura, questo parametro, insieme all'accelerazione precedentemente calcolata, viene utilizzato per il calcolo della la velocità del personaggio alla fine dell'intervallo, in questo caso viene utilizzata la formula del moto rettilineo uniformemente accelerato : v = a*t.

Viene poi calcolato lo spazio percorso dal personaggio tenendo conto della posizione precedente del personaggio, della media tra la velocità precedente e quella appena calcolata e del tempo trascorso, questo valore viene poi moltiplicato per una costante che serve ad aumentare o diminuire la sensibilità dello spostamento. Infine la nuova posizione del personaggio viene calcolata sottraendo alla posizione precedente il valore dello spazio percorso e vengono memorizzati i parametri appena calcolati sostituendo i precedenti. Ecco la porzione di codice che si occupa dell'aggiornamento della posizione del personaggio sullo schermo:

```
void OnReadingChanged(object sender, AccelerometerReadingEventArgs e)
{
    // numero di secondi trascorsi dall'ultima lettura
    double time = (e.Timestamp.Ticks - _time) / 10000000.0;
    // accelerazione media in questo intervallo tenendo conto della precedente
    double ay = ((e.Y * gravity) + _ay) / 2.0;
    // velocità del personaggio alla fine dell'intervallo di tempo
    double vy = _vy + (ay * time);
    // nuova posizione del personaggio
    double sy = _sy - ((((_vy + vy) / 2.0) * time) * _mul);
    //Controllo delle collisioni con i bordi
    //l'immagine rimane ancorata allo schermo e la velocità viene azzerata
    if (sy < 20)
    {
        sy = 20;
        vy = 0;
    }
    else if (sy > 780)
    {
        sy = 780;
        vy = 0;
    }
    // salvataggio dei parametri attuali
    _time = e.Timestamp.Ticks;
    _ay = ay;
    _vy = vy;
    _sy = sy;
    Dispatcher.BeginInvoke(() => AggiornaPosizionePet(sy));
}
```

L’algoritmo inoltre esegue un controllo delle eventuali collisioni del personaggio con i bordi dello schermo: se il personaggio tocca il bordo destro o sinistro dello schermo, la sua velocità viene azzerata.

Ad ogni passaggio del ciclo di gioco vengono inoltre rilevate le eventuali collisioni della texture del personaggio con quelle dei cibi, nel caso in cui l’area di schermo ricoperta dalla texture del personaggio vada a sovrapporsi con quella ricoperta della texture di un alimento, il codice rileva che il giocatore è riuscito a raccogliere un cibo prima che quest’ultimo toccasse terra.

3.3.4 GESTIONE DELLE TEXTURE 2D E DEI MOVIMENTI

Le texture 2D che vengono utilizzate nell'interfaccia di gioco sono principalmente due:

- Il personaggio: questa texture ha due funzioni diverse in base al fatto che il mini-game sia in corso oppure no. Nel caso in cui il mini-game sia in corso la texture di quest'ultimo è statica ed il suo movimento è obbligato a destra o sinistra. Nel caso in cui il mini-game non sia in corso la texture del personaggio deve spostarsi nel punto in cui l'utente ha eseguito un tap sullo schermo, dando l'impressione che cammini.
- Alimenti: questa texture viene utilizzata durante lo svolgimento del mini-game, l'utente non può interagire con il suo spostamento.

Sono state create appositamente due classi per la gestione di queste texture, entrambre ereditano dalla classe "Immagine2D".

I metodi ereditati dalla classe padre sono i seguenti:

- Rectangle GetConfiniDiCollisione: Il metodo ritorna un oggetto della classe Rectangle la cui altezza e larghezza corrispondono con quelle della texture attualmente caricata.
- public Rectangle GetConfini(): Il metodo ritorna una coppia di valori che rappresentano la posizione sullo schermo del punto superiore destro ed inferiore sinistro della texture sullo schermo.
- CenterAtLocation(Vector2 centro): questo metodo accetta in ingresso un oggetto di tipo Vector2, e modifica il valore del campo pubblico Position.
- LoadContent(ContentManager content, string nomeImmagineDaCaricare): questo metodo è virtuale, quindi la

sua implementazione è svolta nelle classi ereditarie, accetta in ingresso un oggetto della classe ContentManger e una variabile string, si occupa della modifica della texture corrente dell'immagine.

- Draw(SpriteBatch spriteBatch): Metodo che si occupa di disegnare sul video la texture attualmente caricata nella classe.

Classe Alimento2D

Questa classe è utile alla gestione della texture di uno degli alimenti che cadono durante il mini-game. La texture della classe è scelta in maniera casuale, il metodo LoadContent prima seleziona dal database la lista dei nomi di alimenti (ad ogni alimento presente in anagrafica corrisponde un'immagine), poi seleziona in maniera casuale un elemento della lista e esegue il caricamento della texture corrispondente. Infine inizializza la posizione della texture ad un punto fuori dallo schermo. Durante il ciclo di gioco la posizione della texture di un alimento deve cambiare costantemente (deve cadere verso il basso), il metodo Update() si occupa di ciò, infatti ogni volta che viene richiamato dal ciclo di gioco, esso incrementa la posizione verticale della texture (l'incremento dipende dal valore del parametro velocità, più è alto, più la velocità con la quale gli alimenti cadranno sarà alta) e se questa è fuori dai confini dello schermo, la riporta in cima, in modo tale che sia possibile utilizzare la stessa texture durante il ciclo di gioco per aumentare il risparmio di memoria.
Ecco la sua implementazione:

```
public void Update()
{
    random = new Random();

    if (Position.X == -1 && Position.Y == -1)
    {
        int x = random.Next(0, 800);
        Position.X = x;
        Position.Y = 0;
    }
    if (Position.Y > 480)
    {
        Position.Y = -1;
        Position.X = -1;
    }
    else
    {
        Position.Y += velocita;
    }
}
```

Il metodo Collision è utile al rilevamento di collisioni tra la texture dell'alimento e quella del personaggio, il metodo riceve in ingresso la posizione della texture del giocatore e le sue dimensioni e restituisce una variabile di tipo bool, il cui valore è true nel caso in cui le posizioni della texture passata in input al metodo e quella dell'alimento collidano oppure false in caso contrario. Ad ogni passaggio del ciclo di gioco viene controllata se la posizione del personaggio è in collisione con la posizione di una delle texture degli alimenti in caduta.

Classe Personaggio2D

Questa classe è utile alla gestione della texture del personaggio. Nell'interfaccia di gioco l'utente può eseguire un tap nel punto in cui vuole che il personaggio si sposti, la texture del personaggio quindi inizierà a muoversi fino tale punto. Durante lo spostamento, le texture del personaggio vengono cambiate costantemente per dare l'impressione all'utente che il suo personaggio cammini. Ogni volta che viene rilevato un tap sullo schermo, il ciclo di gioco richiama il metodo WalkToPosition() della classe Personaggio2D, questo metodo aggiorna il campo positionToReach che rappresenta il punto che il personaggio deve raggiungere alla fine dello spostamento. In seguito il ciclo di gioco richiama il metodo Update() della classe che si occupa dello spostamento della texture e dello spostamento della e del cambio di texture.

Il metodo Update() mette a confronto il valore del campo positionToReach con quello della posizione attuale della texture, se queste sono differenti, come prima cosa incrementa o decrementa il valore X della posizione attuale per allinearlo con il valore di positionToReach, in seguito esegue un allineamento del valore Y, ne deriva che, durante il raggiungimento del punto in cui l'utente ha eseguito il tap, la texture si sposta prima orizzontalmente, poi verticalmente.

Il metodo Update() richiama a sua volta il metodo privato ChangeWalkTexture(), quest'ultimo si occupa di cambiare la texture del personaggio in base alla direzione in cui si sta muovendo, per dare l'impressione all'utente che il personaggio cammini (La texture del personaggio viene cambiata ogni 6 frame, in modo tale da non rendere troppo rapido l'effetto del camminare).

Implementazione del metodo Update():

```
public void Update(ContentManager content)
{
    if (PositionToReach != Position)
    {
        if (PositionToReach.X != Position.X) //spostamento orrizzontale
        {
            //spostamento verso destra
            //mi sposto di 5 in 5 tranne l'ultimo passo se non è multiplo di 5
            if (Position.X < PositionToReach.X) //destra
            {
                ChangeWalkTexture(2, content);
                if (Position.X + 5 > PositionToReach.X)
                    Position.X = PositionToReach.X;
                else
                    Position.X += 5;
            }
            else //spostamento verso sinistra
            {
                ChangeWalkTexture(4, content);
                if (Position.X - 5 < PositionToReach.X)
                    Position.X = PositionToReach.X;
                else
                    Position.X -= 5;
            }
        }
        else //spostamento vertivale
        {
            if (Position.Y < PositionToReach.Y) //basso
            {
                ChangeWalkTexture(3, content);
                if (Position.Y + 5 > PositionToReach.Y)
                    Position.Y = PositionToReach.Y;
                else
                    Position.Y += 5;
            }
            else //alto
            {
                ChangeWalkTexture(1, content);
                if (Position.Y - 5 < PositionToReach.Y)
                    Position.Y = PositionToReach.Y;
                else
                    Position.Y -= 5;
            }
        }
    }
```

3.3.5 INTEGRAZIONE CON FACEBOOK

La pagina relativa all'azione lavoro deve dare la possibilità all'utente di scattare una foto e di memorizzarla nell'isolated storage dell'applicazione, inoltre grazie all'integrazione dell'applicazione con Facebook, l'utente può loggarsi con il suo profilo Facebook e condividere la foto appena scattata per ottenere un bonus maggiore all'esperienza.

Il metodo Show() della classe CameraCaptureTask permette all'applicazione di lanciare l'applicazione relativa alla fotocamera in un processo separato, dopo che la foto viene scattata ed accettata, l'evento Completed viene invocato, l'event handler per questo evento si occupa di salvare lo stream di dati corrispondente alla foto nell'isolated storage dell'applicazione come immagine Jpeg.

Per integrare l'applicazione con Facebook il progetto fa utilizzo della Facebook C# SDK, questa SDK fornisce una serie di classi e metodi che questa applicazione utilizza per implementare le seguenti funzioni:

- Esecuzione del login

  Una volta che l'utente clicca sul tasto di login, il componente web Browser indirizza l'utente alla pagina di login di facebook, l'url della pagina è stato ottenuto mediante il metodo GetLoginUrl la classe FacebookClient passando come parametri le extendedPermissions (permessi per condividere post ed ottenere informazioni relative all'utente loggato). Una volta che l'utente ha eseguito il login, nell'havent handler relativo all'evento Navigated l'applicazione crea un nuovo oggetto FacebookOAuthResult eseguendo un parsing dell'url al quale il webBrowser viene reindirizzato ed ottenendo in questo modo l'AccessToken.

L’AccessToken viene in seguito utilizzato dal metodo LoginSuceeded per ricavare l’UserID il quale verrà poi passato come parametro al metodo asincrono FacebookClient.GetAsinc(), utilizzato per ricavare nome, cognome ed immagine del profilo dell’utente che ha eseguito il login. AccessToken e UserID vengono poi memorizzati nell’isolatedStorage, in questo modo ad un secondo avvio dell’applicazione l’utente non dovrà eseguire nuovamente inserire le proprie credenziali per il login (se l’access token è ancora valido).

- Pubblicazione di un post sul profilo dell’utente attualmente loggato

  Il metodo SharePhoto() crea l’oggetto fbUpl della classe FacebookMediaObject ed inizializza i campi FileName e ContentType con le stringhe relative al nome del file da postare e all’estensione dell’immagine. In seguito l’immagine acquisita da fotocamera viene convertita in un array di byte il quale viene salvato dentro all’oggetto fbUpl. Come ultima cosa viene richiamato il metodo asincrono PostAsync() della classe FacebookClient il quale pubblica il post sul profilo dell’utente (a questo metodo vengono passati come parametri l’oggetto fbUpl ed il messaggio da pubblicare nel post). Se il post è stato pubblicato senza errori, verrà visualizzata una MessageBox che informerà l’utente dell’avvenuta pubblicazione.

- Esecuzione del logout

  Per l’esecuzione del logout, bisogna in primo luogo ricavare l’url della pagina relativa. Il metodo FacebookClient.GetLogoutUrl ci

fornisce questo URL includendo in esso due parametri: l'access token e l'URL della pagina alla quale si vuole essere reindirizzati alla fine del Logout. Una volta ottenuto l'url il web browser navigherà all'indirizzo indicato, invalidando l'access token.

Quando la pagina viene caricata, se l'access token è presente nell'isolated storage l'applicazione esegue una richiesta asincrona a Facebook per ottenere i dati dell'ultimo utente loggato, questa richiesta è eseguita dentro un costrutto try catch, in quanto non è possibile prevedere se l'access token memorizzato sia ancora valido. Nel caso la richiesta non vada a buon fine, l'utente dovrà rieseguire il login per ottenere un altro access token.

4 CONCLUSIONI E SVILUPPI FUTURI

Lo sviluppo di questo progetto, innanzitutto, mi ha permesso di imparare come sviluppare un'applicazione Windows Phone da zero, cosa che non ho avuto modo di imparare durante le lezioni e che si è rivelata molto interessante.
Il progetto presentato, inoltre, mi ha permesso di approfondire meglio alcuni aspetti fondamentali nel mondo dello sviluppo di applicazioni mobile quali la geolocalizzazione o l'integrazione con Facebook.

Le maggiori difficoltà riscontrate durante lo sviluppo riguardano la parte relativa all'integrazione con Facebook in quanto, per poter capire il corretto funzionamento di tutte le funzioni messe a disposizioni dall'SDK, è stato necessario consultare a fondo la documentazione relativa ed osservare varie implementazioni scritte da terze parti per poterne capirne il corretto funzionamento.

Alcuni sviluppi futuri dell'applicazione potrebbero riguardare l'implementazione di una parte multiplayer grazie alla quale gli utenti possano sfidarsi a vicenda, o collaborare, per ottenere bonus aggiuntivi per i propri personaggi. Per mettere in comunicazione gli utenti tra loro, l'applicazione potrebbe far uso del bluetooth, oppure per gli utenti distanti tra loro potrebbe essere necessaria l'implementazione di una applicazione lato server che comunichi con quelle client (installate sui dispositivi degli utenti).
Inoltre, prima di una eventuale pubblicazione sullo Store, l'applicazione necessiterà, oltre di varie migliorie all'interfaccia grafica, anche di una conversione per poter essere sfruttata dall'ultimo sistema operativo

mobile di casa Microsoft: Windows Phone 8 (la cui SDK è uscita poco meno di un mese fa).

5 BIBLIOGRAFIA

1) D. Betts, F. Boerr, S. Densmore, J. G.Salazar, A. Homer, "Windows Phone 7 Developer Guide 2011"

2) A. Boaretto, G. Noci, F. Maria Pini "Mobile Marketing"

3) T. B. Jones, M. Perga, M. Sync, "Windows Phone 7 in action"

4) IDC: Android will peak in 2012, Windows Phone to eclipse iOS in 2016

"http://www.androidguys.com/2012/06/06/idc-android-will-peak-in-2012-windows-phone-to-eclipse-ios-in-2016/"

5) Wikipedia, Sistema operative per dispositive mobile "http://it.wikipedia.org/wiki/Sistema_operativo_per_dispositivi_mobili"

6) Google Places API documentation "https://developers.google.com/places/documentation/"

7) Facebook C# SDK documentation "http://facebooksdk.codeplex.com/documentation?version=94"

8) Mobile Phones Tracking "http://en.wikipedia.org/wiki/Mobile_phone_tracking"

9) Windows Phone Developer Center "http://dev.windowsphone.com/en-us/"

10) "Introduzione allo sviluppo con XNA Game Studio" "http://msdn.microsoft.com/it-it/library/bb203894(v=xnagamestudio.40).aspx"