UNIVERSITA' DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN

INGEGNERIA INFORMATICA

LIBROPIU':
PROGETTAZIONE E REALIZZAZIONE DI UNA LIBRERIA DIGITALE CON iOS

RELATORE: PROF. GIORGIO MARIA DI NUNZIO

LAUREANDO: ANDREA ANTONINI

Anno Accademico 2012/2013

Indice

Elenco delle figure

Elenco dei listati codice

Capitolo 1

Introduzione

L'obiettivo dell'esperienza di tirocinio svolta presso la ditta BAZZACCO S.R.L., nell'anno accademico in corso (2012/2013), con sede a San Martino di Lupari (Pd), è stato quello di sviluppare un'applicazione che gestisca il funzionamento di una libreria digitale. I libri digitali, detti anche "electronic book", sono dei file (di vario tipo, tra i quali pdf, doc, html) acquisiti ed elaborati da un software che possono essere scaricati da internet, acquistati o prelevati gratuitamente online, e quindi essere letti attraverso un computer, un tablet pc o un e-book reader (lettore dei libri in formato digitale). In altre parole si accende il computer o il reader, si sceglie il libro desiderato, lo si scarica e lo si inizia a leggere, oppure lo si ripone nella libreria virtuale, la quale può contenere milioni di volumi pronti per essere sfogliati e riposti con semplici click. La libreria, quindi è un database che contiene tutte le opere disponibili (di tipo scolastico e non scolastico) corredate ciascuna da una possibile recensione, prezzo, sconto e isbn.
Il codice isbn (dall'inglese International Standard Book Number , "numero di riferimento internazionale del libro"), è una sequenza numerica di 13 cifre usata internazionalmente per la classificazione dei libri (è ancora utilizzata la codifica antecedente il 2007, costituita da un numero di cifre pari a 10, in cui l'ultimo carattere può contenere la lettera maiuscola "X"). Ogni codice isbn identifica in modo univoco ogni specifica edizione di un libro, (non però per le semplici ristampe, che mantengono lo stesso codice dell'edizione cui si riferiscono) e, una volta assegnato non può più essere riutilizzato.
Tutto questo viene sviluppato perchè già dall'anno scolastico 2013/2014 saranno messi a disposizione degli studenti[1] delle medie e superiori libri digitali consultabili direttamente da tablet (il motivo principale), ma soprattutto perché la vendita di questi dispositivi è cresciuta notevolmente nell'ultimo anno riscuotendo un notevole successo. Tale applicazione verrà sviluppata avendo come riferimento tablet basati su sistema operativo iOS, mediante il tool grafico XCode.
Libropiù, il nome del progetto, richiederà un anno di tempo per poter essere sviluppato e testato nelle sue parti. Esso rappresenta un'innovazione in questo campo che ormai da parecchi anni si cerca di far emergere, ma che per vari motivi non ha mai avuto il giusto slancio anche tra le parti commerciali in gioco.
Il mio lavoro in questo progetto, è stato quello di provvedere in primo luogo ad un'analisi del progetto nelle sue parti e successivamente il passaggio all'implementazione di una parte: la registrazione di un utente al servizio per poter scaricare il libro.

Nel secondo capitolo verrà analizzato il progetto nelle sue parti, fornendo una descrizione dei vari elementi che lo compongono, mostrando e descrivendo anche un tool alternativo di realizzazione mediante codice Delphi. Nel terzo capitolo verrà realizzata l'applicazione

[1] Per gli studenti delle elementari bisognerà attendere l'anno scolastico 2014/2015. Per altre informazioni si puo' consultare la norma contenuta nel decreto Digitalia consultabile nella gazzetta ufficiale al sito: http://www.gazzettaufficiale.it/eli/id/2013/01/11/13A00321/sg.

col linguaggio Objective-C, mediante XCode, nella parte che è stata di mia competenza. Sempre nel terzo capitolo sarà fornita una breve analisi sulla codifica crittografica salt a 128 bit utilizzata nel progetto andando a evidenziare i vantaggi che si hanno nel caso di un attacco a dizionario e infine nel quarto capitolo verranno redatte delle conclusioni personali sull'esperienza svolta su questo progetto.

Capitolo 2

Analisi del progetto

In questo capitolo vedremo :

- **L'analisi dei requisiti** che andranno a descrivere, in un linguaggio informale, le entità che descrivono le caratteristiche del sistema
- **Glossario** dei termini
- La procedura di **registrazione di un utente** al servizio
- La procedura di **registrazione di un nuovo dispositivo**
- La procedura di **acquisto di un libro** digitale
- La procedura di **scaricamento di un libro digitale**
- Scelta degli strumenti di sviluppo

2.1 Analisi dei requisiti

Qui di seguito vengono descritte le entità che dopo un'attenta analisi abbiamo evidenziato. Esse rappresentano una descrizione del sistema nelle sue parti, fornendo specificatamente per ognuna una breve descrizione e la funzione che ricopre nel progetto Libropiù. Il progetto Libropiù, è una "nuvola" di contenuti, servizi e strumenti digitali che ha l'e-book, la versione digitale del libro cartaceo. Intorno, migliaia di video, audio, tutorial, attività interattive che rendono le lezioni in classe e lo studio a casa più immediati ed efficaci. Da Libropiù si puo' accedere direttamente alla propria Home Page personale, si possono attivare gli e-book e le proprie risorse. La propria Home Page personale su Libropiù è uno spazio riservato, dove si puo' trovare con facilità tutto quello che serve all'utente: i propri e-book, le proprie risorse e i propri strumenti, ma anche consigli e suggerimenti per la propria formazione e un collegamento diretto al mondo social. Si puo' accedere, direttamente dalla home page di Libropiù, semplicemente inserendo il proprio username e la password nella Login Area (se non sei ancora registrato, clicca qui) e utilizzando il link "La tua Home".
Di seguito viene fornito l'elenco completo delle entità del sistema:

- *Book Store*: Sito web accessibile da browser dove l'utente potrà registrarsi e acquistare i libri digitali.
- *Website DB*: Database management system a supporto del Book Store.
- *PC,Tablet*: Costituiscono gli strumenti attraverso i quali i libri digitali verranno utilizzati dall'utente finale. Applicazioni specifiche verranno istallate sui dispositivi per consentire la lettura e la gestione della libreria personale. Dopo aver acquistato un libro, questo sarà scaricabile sui dispositivi dell'utente finale fino al raggiungimento del numero massimo consentito. Sarà comunque possibile liberare un dispositivo per consentire lo scaricamento su un altro. Nel caso di rottura o smarrimento di un dispositivo sarà necessario contattare il supporto tecnico per liberare le copie scaricate sul dipositivo perso. Le App installate sui dispositivi

potranno prevedere funzionalità di ausilio alla particolare tipologia di libri commercializzati come la condivisione degli appunti o funzioni di aiuto all'insegnante durante le lezioni (ad esempio la verifica della pagina attualmente aperta dallo studente).

- *Device Manager:* Server controparte dell'applicazione di gestione della libreria personale. Dovrà gestire il riconoscimento e la registrazione del dipositivo e l'associazione con l'account utente. Inoltre veicolerà le informazioni sui libri acquistati e scaricabili sul dispositivo oltre allo scaricamento vero e proprio delle chiavi di utilizzo e dei contenuti. Consentirà inoltre la condivisione degli appunti e note tra diversi utenti e dispositivi dell'utente.
- *Master Appliance:* Database management system a supporto del Device Manager e delle applicazioni dedicate alla preparazione dei contenuti da scaricare sui dispositivi degli utenti.
- *Master Book Storage:* Archivio dei contenuti digitali costituiti dai libri in vendita.
- *Crypt Machine Appliance:* Applicazione che prepara i libri per lo scaricamento sui dispositivi. Si collega alla lista dei lavori in attesa di evasione contenuta nella Master Appliance. I lavori sono costituiti da libri che vanno criptati per essere utilizzabili solo su specifici dispositivi. Preleva quindi dal Master Book Storage la copia richiesta e la deposita dopo il processo di protezione in un'area di storage temporanea in attesa dello scaricamento sul device dell'utente. Aggiorna quindi la Master Appliance sulla fine del processo affinchè possa iniziare il download attraverso il Device Manager.

2.2 Glossario

In questa tabella viene assegnata per ogni entità del sistema una frase "caratteristica" per fissare bene la loro funzione all'interno del progetto.

Termine	**Descrizione**
Book Store	Sito Web di registrazione
Website DB	Database a supporto del Book Store
PC,Tablet	Device per usufruire del servizio
Device Manager	Server che associa account-dispositivo
Master Appliance	D.B.M.S a supporto del Device Manager
Master Book Storage	Archivio dei libri in vendita
Crypt Machine Appliance	Software incaricato di criptare i pdf dei libri per poter essere visualizzati in specifici dispositivi

2.3 Registrazione di un nuovo utente al servizio

Qui vengono descritti i passi da seguire per la registrazione di un nuovo utente. Essi rappresentano i passi fondamentali della procedura che implementa ciò a livello di codice. La registrazione provvede ad assegnare un nome utente e password all'utente in modo da permettere la sua autenticazione al servizio.

Di seguito i passi della procedura da eseguire per la registrazione di un nuovo utente:

- da un browser (PC o Tablet), l'utente si collega al Book Store e comunica i dati richiesti;
- i dati vengono memorizzati sul Website DB;
- il device DB ad intervalli regolari sincronizza il proprio database utenti con quello del Website DB.

2.4 Registrazione di un nuovo dispositivo

La registrazione di un nuovo dispositivo rappresenta la fase più importante. Qui vengono registrati i dispositivi che possono utilizzare il servizio del libro "digitale". E' in questa parte che puo' verficarsi, da parte di "estranei" il furto delle credenziali di accesso per l'uso del dispositivo, motivo per cui, nella sezione 3.2, sarà descritta la codifica salt utilizzata nel progetto per la crittografia dei contenuti.

Registrazione di un nuovo dispositivo:

- dall'applicazione installata sul dispositivo utente viene instaurato un collegamento "sicuro" con il Device Manager. Per "sicuro" si intende che sarà presente una procedura di criptazione dei dati che nel nostro caso è rappresentata dalla codifica *salt* a 128 bit.
- Il nome utente e la password creati in fase di registrazione, vengono utilizzate come autenticazione al fine di poter stabilire a quale profilo utente associare il dispositivo.
- Il Device Manager raccoglie le informazioni sul dispositivo e lo associa al profilo dell'utente provvedendo ad aggiornare anche il Website DB.

2.5 Acquisto di un libro digitale

La fase di acquisto (si intende a pagamento effettuato), attraverso una procedura, fà si che venga inserito nella "libreria" dell'utente il libro da lui scelto, in modo che possa essere successivamente fruibile nel suo contenuto attraverso la fase finale di download.

Di seguito i passi della procedura da eseguire per l'acquisto del libro digitale:

- attraverso un browser l'utente si autentica sul Book Store tramite nome utente e password creati in fase di registrazione;
- sceglie ed acquista un libro digitale dal catalogo dei libri;

- il libro entra a far parte della libreria dell'utente memorizzata sul Website DB;
- il Device DB aggiorna il proprio database dei libri acquistati e trova il nuovo acquisto che l'utente ha effettuato;
- infine inserisce il libro nell'elenco dei libri scaricabili dai dispositivi dell'utente.

2.6 Scaricamento di un libro digitale

Lo scaricamento del libro "digitale", come già detto, rappresenta l'ultima fase per poter effettivamente leggere e consultare il libro. Questo avviene mediante un'interrogazione al database (Website Db) verificando quali libri sono pronti al download per l'utente autenticato al servizio.

Di seguito i passi della procedura da compiere per poter scaricare i libri digitali disponibili:

- Dall'applicazione istallata sul dispositivo l'utente si collega al Device Manager autenticandosi con username e password;
- il Device Manager fornisce l'elenco di libri acquistati con indicazioni su quali possono essere scaricati sul dispositivo;
- l'utente sceglie quali libri scaricare sul dispositivo;
- il Device Manager comunica al Device DB quali libri l'utente ha intenzione di scaricare;
- il Device DB compila una lista di lavori in base ai libri e ai dispositivi sui quali devono essere fruibili;
- La /le Crypt Machine interrogano il Device DB circa i possibili lavori di protezione richiesti e procedono, secondo una logica fifo, a criptare i libri con chiavi generate a partire dalle informazioni hardware sui dispositivi di destinazione;
- al termine del lavoro di protezione la Crypt Machine deposita il libro digitale caratterizzato per lo specifico dispositivo utente in un'area di storage temporanea e comunica al Device DB le chiavi necessarie alla decrittazione;
- infine il Device Manager si accorge interrogando il Device DB quali libri sono pronti per il download e lo comunica all'applicazione utente che puo' iniziare lo scaricamento.

2.7 Scelta degli strumenti di sviluppo

Nello sviluppo dell'applicazione, come già detto viene utilizzato il classico XCode[2] con l'utilizzo di interfaccia grafica, ma per completezza si vuole citare un altro tools, che si chiama Rad Studio Xe4[3]. La tecnica di programmazione RAD che sta per "Rapid Application Development", permette al programmatore di risparmiare molto tempo sulla stesura del codice e la sua ottimizzazione, mettendo a disposizione dei componenti, ovvero delle classi chiamate Unit, che svolgono in maniera molto efficiente procedure, funzioni e chiamate a sistema. Rad Studio Xe4 utilizza il Delphi come linguaggio, ed è stato rilasciato dalla Embarcadero nell'aprile di quest'anno. Il Delphi è un linguaggio

[2] https://developer.apple.com/
[3] http://www.embarcadero.com/

procedurale basato sul Pascal orientato agli oggetti, le cui componenti principali sono le VCL (Visual Component Library), esclusivamente sotto dominio Windows, e da poco introdotte le FMX (Firemonkey Project), un'astrazione delle VLC che permettono la compilazione nativa per diverse piattaforme dello stesso codice sorgente. La peculiarità di questo linguaggio è che può compilare in un singolo eseguibile, autonomo rispetto a librerie esterne, semplificando la distribuzione e i problemi delle diverse versioni delle dll. Vediamo ora un breve esempio, descrivendo i vari passi da compiere per scrivere un'applicazione, avendo prima ovviamente istallato il pacchetto software nel proprio pc.

Passo 1:Creare un nuovo progetto FireMonkey per Ios.

- Seleziona **File > New > FireMonkey Mobile Application - Delphi**:
- Seleziona **Blanck Application.**

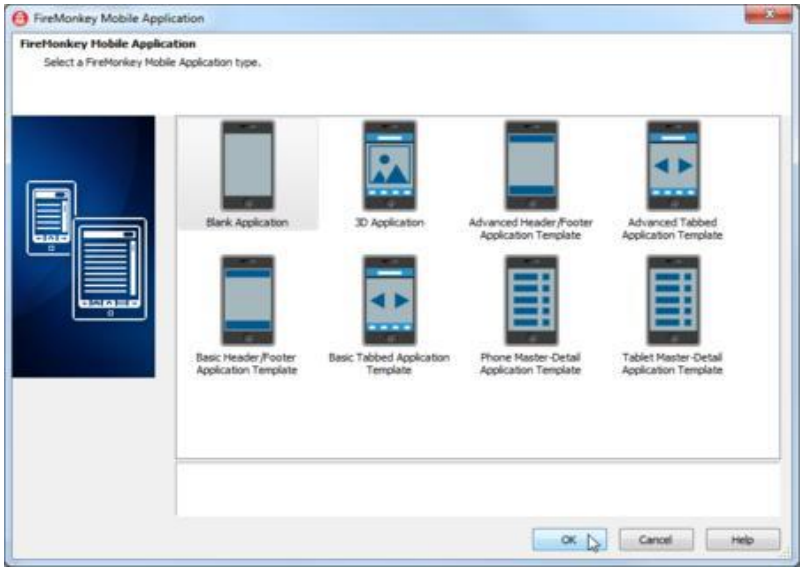

Figura 3.1 Firemonkey Blank Application

- **FireMonkey Mobile Form Designer** mostra una nuova form per Ios.

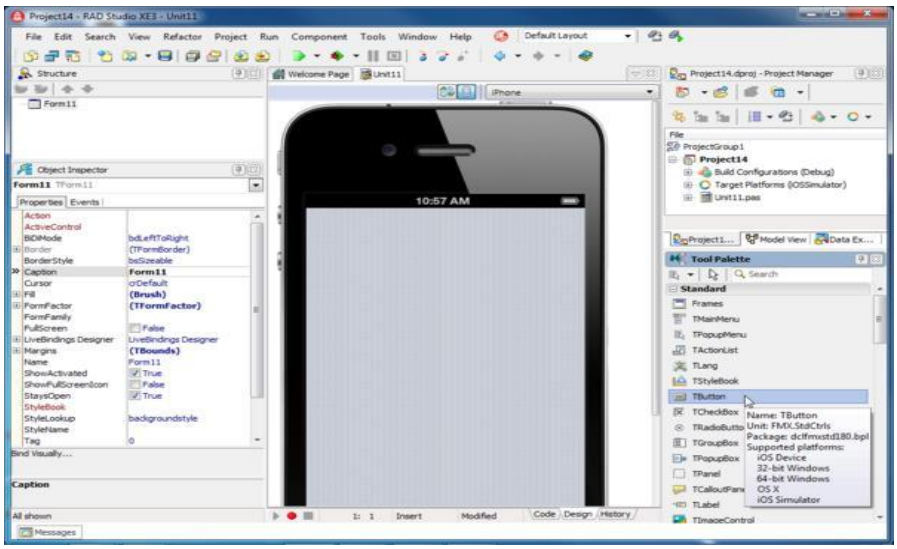

Figura 3.2 Firemonkey Form Designer

Da qui in poi è possibile cominciare a piazzare dalla tool palette (in basso a destra) i componenti desiderati per poter cominciare a costruire la nostra applicazione. Per andare a scrivere il codice desiderato, basterà selezionare l'oggetto e il relativo evento (OnClick, On DoubleClick) dall'Object Ispector (in basso a sinistra).

Ora dobbiamo testare l'applicazione sul nostro mac. Per fare cio' abbiamo bisogno di un oggetto che si chiama paserver, un thread messo in esecuzione sul nostro mac che intercetta la compilazione dal nostro pc Windows (ovviamente i pc sono in rete) e di un simulatore di dispositivo Ios già impostabile da Rad Studio. Questo è fatto in modo per poter testare gradualmente la nostra applicazione e solo alla fine trasferire il file eseguibile prodotto nel dispositivo Ios interessato. Alla fine di tutto il processo è possibile trasferire il file eseguibile prodotto sul nostro dispotivo per poter poi essere commercializzato.

Mi porto sulla cartella del mio Mac dove risiede il file *PaServer* precedentemente scaricato dal sito produttore di Rad Studio. Lo avvio da terminale, e una volta lanciata da Rad Studio l'applicazione mediante il tasto Run questo è il risultato: viene lanciata l'applicazione che ho implementato.In questa figura un esempio di "Hello World".

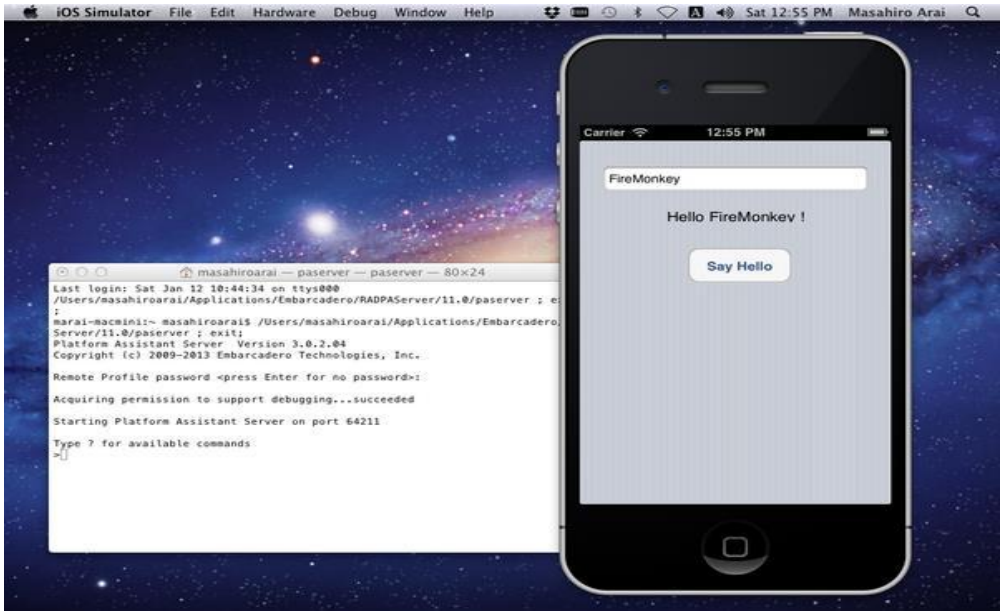

Figura 3.3 Simulazione applicazione Rad Studio sul Mac

Il vantaggio della programmazione con questo tool si vede chiaramente. Il programmatore non si preoccupa di conoscere la sintassi di Objective-C, chiaramente più difficile e non di facile comprensione. Scrivendo mediante sintassi basata su codice Delphi è possibile sviluppare applicazioni iOS in un tempo più snello e quindi con una messa in produzione più veloce. Tuttavia, essendo il tool di sviluppo ancora abbastanza "nuovo" sul mercato è stato deciso di continuare lo sviluppo di LibroPiu' mediante XCode, come idea iniziale di progetto. Per una trattazione completa di questo strumento si può consultare la documentazione[4] citata nel sito a piè pagina.

[4] http://docs.embarcadero.com/products/rad_studio/radstudioXE4/iOS%20Tutorial%20en.pdf

Capitolo 3

Realizzazione di Libropiù

In questo capitolo vedremo:

- Analisi del file sorgente per la registrazione di un nuovo utente al Book Store.
- Cenni della codifica salt utilizzata per la criptazione dei contenuti, in modo da evitare lo scaricamento dei libri da persone non autorizzate.
- Schermata applicazione Libropiu': esempio.

L'applicazione che si sta sviluppando è molto vasta e ho deciso di commentare questa parte perchè mi sembra la più significativa e interessante (oltre ad essere parte di mia competenza), cosa che altrimenti risulterebbe pesante proponendo l'analisi di pagine e pagine di codice.

3.1 Analisi del file sorgente per la registrazione di un nuovo utente al Book Store.

Ora riporterò il file con estensione (.h) con le definizioni dei metodi e commentero' il file di relativa implementazione, consultabile in appendice A.3.

File RegisterLoginView.h:

```
1.  #import <UIKit/UIKit.h>
2.  #import <QuartzCore/QuartzCore.h>
3.  @class NuovoUtenteView;
4.  @interface RegisterLoginView : UIViewController
5.  @property (strong, nonatomic) IBOutlet UIButton *NuovoUtenteButton;
6.  @property (strong, nonatomic) IBOutlet UIButton *LoginButton;
7.  @property (strong, nonatomic) IBOutlet UITextField *UserName;
8.  @property (strong, nonatomic) IBOutlet UITextField *Password;
9.  @property (nonatomic, strong) NuovoUtenteView *NuovoUtenteView;
10. -(IBAction) NuovoUtenteButtonPush: sender;
11. -(IBAction) LoginButtonPush: sender;
12. @end;
```

Osservazioni e considerazioni:

Nella programmazione ad oggetti, ogni attributo di una classe dovrebbe essere sempre privato per poter implementare il concetto di incapsulamento. Per poter accedere agli attributi, solitamente, vengono creati dei metodi di set e di get da utilizzarsi per conoscerne e modificarne il valore. La scrittura dei metodi set e get è un operazione abbastanza noiosa e che richiede diverso tempo, in particolar modo se abbiamo a che fare con classi con molti attributi. L'Objective-C viene incontro a questa esigenza, agevolando la scrittura

di questi metodi utilizzando i costrutti @**property** e @**synthesize**. Ora analizziamo le righe di codice scritte nel file *RegisterLoginView.h.*

Riga (3)-importo la classe NuovoUtenteView.h ricorsivamente già pronta e implementata.

Riga (4)-per prima cosa creiamo l'interfaccia che conterrà i nostri metodi e argomenti.

Ora vengono descritti gli attributi di tale classe:

Riga (5)-viene definito l'attributo UIButton(bottone) per registrare un nuovo utente.

Il termine *nonatomic* e *strong* hanno rispettivamente questo significato:

- indica che il metodo creato avrà la caratteristica di atomicità. Ciò è indispensabile se si usano, nell'applicazione, i thread per garantire la consistenza dei dati.
- incrementa di uno il contatore che gestisce i riferimenti all'oggetto. Questo è il meccanismo di retain/realase (Appendice A.2)

Righe (6)…(9)-vengono definiti gli attributi UIButton (bottone) e TextField (campo testo) per la registrazione di un nuovo utente o per il login di un utente già esistente.

Righe (10)…(11)-viene scritta la firma dei metodi: la registrazione del nuovo utente e il login di uno già inserito nel dbms.

In tutte le dichiarazioni degli attributi (bottoni, testo editabile) compare la parola chiave **IBOutlet.** Questa parola chiave fà si' che vi sia un collegamento tra l'oggetto, per esempio il bottone e il suo evento. In questo modo sarà possibile attivare il metodo ad un particolare evento: doppio click, click, ecc…, riferimento utilizzato quando si utilizza l'**Interface Builder**, strumento che consente di utilizzare interfaccie grafiche in XCode.

Ora passiamo al file di implementazione.

Osservazioni e considerazioni:

In questa sezione implementativa si deve implementare una procedura che permetta il login dell'utente se già esiste, oppure ne permetta la registrazione se è la prima volta che utilizza il servizio. Nel caso l'utente si debba autenticare si controllerà la consistenza dei dati inseriti, dapprima controllando che username e password siano diversi da stringa vuota e successivamente controllando che sia disponibile la connessione ad internet per permettere il login vero e proprio dell'utente. Una volta verificata la presenza dell'utente nel dbms, si procederà alla verifica di libri disponibili allo scaricamento precedentemente ordinati, che saranno prelevati mediante download in caso affermativo di disponibilità. Nel caso della registrazione di un nuovo utente, mediante un form l'utente procederà alla compilazione di alcuni dati richiesti per poi essere inserito nella banca dati, condizione che lo porterà a usufruire del servizio del libro digitale. In questa fase avviene la crittografia della credenziali, descritta nel paragrafo 3.3.
L'utente dispone di tutte queste informazioni, mediante la propria Home Page, sempre consultabile per vedere la propria situazione riguardo ordini e acquisti. Le procedure implementate sono consultabili nell'appendice A.4. Ho scelto di riportare solo il codice

relativo alle procedure che ora descriverò, perché il blocco di codice che implementa tutte le funzionalità della Homepage sono oltre quattrocento righe, cosa un po' pedante da leggere e analizzare.
La procedura *ControllaOrdiniNuovi()* controlla la presenza di nuovi ordini da parte dell'utente, interrogando mediante uno script fatto in Php il database, mostrando il numero di ordini disponibili. La procedura CaricaLibreria(), mostra i libri disponibili per l'utente, predisponendoli per il download, interrogando il database mediante una query di selezione.

Un'altra parte di cui mi sono occupato è la rotazione visiva dell'applicazione quando per esempio ruoto l'Ipad o l'Iphone, che in gergo si chiama "rotazione della vista". Questo è il codice da scrivere nella parte implementativa di CountRotateAppViewController:

```
1. -(BOOL) shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation{
2. // Return YES per rotazione
3. return YES;//No per la rotazione non automatica
4. }
```

Questo è un metodo della classe "UIViewController", che implementa già la rotazione automatica dello schermo. Ancora una volta non dovremo fare veramente niente, in quanto sono le librerie del SDK a fornirci tutto pronto. Piccola nota: se non volessimo implementare l'auto-rotazione, ci basterebbe ritornare come valore "NO", oppure semplicemente non inserire il metodo nell'applicazione.

3.2 Cenni alla codifica Salt utilizzata per la crittografia dei contenuti.

Derivare una chiave crittografica significa definire una funzione di derivazione della chiave (in inglese Key derivation function o KDF) che fa derivare una o più chiavi segrete da informazioni segrete e/o altre informazioni, che invece sono note. Un algoritmo di derivazione della chiave, a partire da un valore arbitrario in ingresso (una stringa o un array di larghe dimensioni) deriva in modo crittograficamente sicuro (ovvero non è possibile abbreviare il calcolo con una qualche scorciatoia) una chiave di dimensioni adatte alla cifratura. Questo fa sì che sia possibile imporre un dato tempo computazionale per generare una chiave, nota la password (o una qualsivoglia informazione segreta), in modo che un attaccante che provi un attacco a forza bruta o anche solo un attacco a dizionario si trovi rallentato (in modo non bypassabile) nelle operazioni. Unendo alla password un valore pseudocasuale, che non ha bisogno di restare segreto (seed o salt), in ingresso alla funzione di derivazione della password si fa in modo che l'avversario non possa nemmeno precomputare le chiavi corrispondenti alle password più provabili, perché saranno diverse al cambiare del salt, e quindi diverse per ogni file cifrato. Le funzioni di derivazione di chiave sono funzioni di hash crittografiche, spesso usate in congiunzione con parametri non segreti per derivare una o più chiavi da una stessa informazione segreta. Un suo utilizzo può evitare che un attaccante che entri in possesso della chiave derivata apprenda informazioni sensibili sul valore segreto effettivamente in input. Una KDF può anche essere usata per assicurare che le chiavi derivate abbiano altre proprietà

desiderabili, come ad esempio evitare chiavi deboli in alcuni specifici sistemi di crittografia. Le funzioni di derivazione di chiavi sono spesso usati come componenti di protocolli di key-agreement tra più parti. Esempi di alcune funzioni di derivazione di chiavi includono KDF1, definito nello IEEE P1363, e funzioni simili in ANSI X9.42. Funzioni di derivazione delle chiavi sono usate anche per derivare delle chiavi da una password segreta o una passphrase. Un particolare uso di una funzione di derivazione di chiave può essere il seguente:

DK=KDF(Key,Salt,Iterations) dove:

- DK è la chiave derivata
- KDF è la funzione di derivazione della chiave
- Key è la chiave originale o la password
- Salt è un valore casuale che svolge il ruolo si salt crittografico
- Iterations si riferisce al numero di iterazioni della funzione

La chiave derivata è usata al posto della chiave originale o della password, come chiave del sistema. Il valore del salt e il numero di iterazioni (se non è prefissato) sono memorizzati nel valore hash della password o inviati come testo in piano insieme al messaggio criptato. La difficoltà di forzare la chiave con un attacco di brute-force aumenta all'aumentare del numero di iterazioni. Un limite pratico sul numero di iterazioni è la riluttanza degli utenti a tollerare un ritardo percepibile nell'effettuare il login nel sistema o nel vedere il messaggio decriptato. L'uso di un salt crittografico evita, come detto prima,di precomputare un dizionario di chiavi derivate.
Nella crittanalisi e nella sicurezza informatica, un attacco a dizionario è una tecnica di attacco alla sicurezza di un sistema o sottosistema informatico mirata a "rompere" un codice cifrato o un meccanismo di autenticazione provando a decifrare il codice o a determinare la passphrase cercando tra un gran numero di possibilità. In pratica si tenta di accedere a dati protetti da password (sia remoti, come ad esempio account su siti web o server di posta, database server; sia locali, come documenti o archivi protetti da password) tramite una serie continuativa e sistematica di tentativi di inserimento della password, solitamente effettuati in modo automatizzato, basandosi su uno o più dizionari. In contrasto con un metodo forza bruta (o attacco brute force), dove tutte le possibili password sono ricercate in maniera esaustiva, un attacco a dizionario prova solamente quelle ritenute più probabili, tipicamente contenute in una lista (detta dizionario). Generalmente, questi attacchi, detti per questo "a dizionario", hanno successo perché la maggior parte delle persone ha la tendenza a scegliere password semplici da ricordare (e quindi semplici da scoprire, ad esempio il proprio nome, quello dei propri figli, date di nascita) e tendenzialmente sceglie parole prese dalla propria lingua nativa.

Il nome della codifica salt (sale in italiano), sta proprio ad indicare un’aggiunta di una sequenza casuale di bit utilizzata assieme ad una password come input a una funzione unidirezionale, di solito una funzione hash (una funzione specifica che data la mia chiave originale calcola una chiave aggiunta) il cui output è conservato al posto della sola password, e può essere usato per autenticare gli utenti. Usando dati sale si complicano gli

attacchi a dizionario, quella classe di attacchi che sfruttano una precedente cifratura delle voci di un elenco di probabili parole chiave per confrontarle con l'originale: ogni bit di sale utilizzato raddoppia infatti la quantità di memorizzazione e di calcolo necessari all'attacco.

Si supponga che la chiave segreta di un utente venga rubata da un database sotto forma di hash, e che sia noto che la password originale fosse una delle 200.000 parole della lingua italiana. Sapendo che il sistema utilizza un valore salt lungo 32 bit, gli hash pre-calcolati dell'attaccante non sono più di alcuna utilità: in questo caso, un utente malintenzionato dovrebbe calcolare l'hash di ogni parola con ognuno dei (4.294.967.296) possibili valori sale, fino a trovare una corrispondenza. Il numero totale di tentativi possibili può essere ottenuto moltiplicando il numero di parole nel dizionario con il numero di valori sale possibili:

$$2^{32} * 200000 = 858993459200000$$

Per completare un attacco a forza bruta, l'attaccante dovrebbe calcolare 858993459200000 di hash, invece di soli 200.000. Anche sapendo che la password stessa è semplice, il valore segreto del sale rende molto più difficile individuare la password. Nel nostro caso abbiamo utilizzato una salt a 128 bit quindi avremo:

$$2^{128} * 200000 = 68056473384187692692674921486353642291200000$$

possibili tentativi che il malintenzionato dovrebbe provare per poter “hackerare” lo scaricamento del libro digitale. Da qui si può capire quindi come l’uso di questa codifica sia estremamente utile al fine di evitare spiacevoli inconvenienti da parte dell’utente che utilizza il servizio.

3.3 Schermata applicazione Libropiù.

Di seguito viene riportata la schermata di autenticazione di Libropiù.

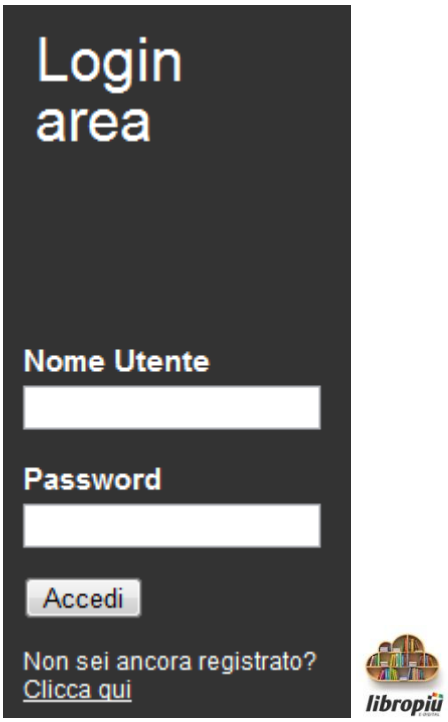

Figura 3.1 Schermata applicazione Libropiu'

Ora per provare l'applicazione lasciamo bianchi i campi **nome utente**, **password** e premiamo il tasto entra, ovviamente quando verrà venduta l'applicazione verranno creati i profili utente e ognuno inserirà il suo *Nome Utente e Password*. Se non si è registrati si verrà mandati a una schermata di compilazione di alcuni dati per registrarsi.
In questo modo il sistema provvederà a visualizzare una prenotazione di libri scaricabili dal *Book Store*.
Ecco la schermata della propria situazione dei libri ordinati e possibile scaricamento.

N.	Stato	Titolo	Editore	Prezzo €	Ordinato q.ta	Ordinato Data	Disponibile q.ta	Disponibile Data	Ritirato q.ta	Ritirato Data	Pagato q.ta	Pagato Data	Note
1	Ritirato	**MOBY DICK (PICCARDI) X MEDIA E SUP.** Isbn: 9788828601494 - Autore: MELVILLE	EINAUDI SCOL.	10,54	1	18/11/11	1	01/12/11	1	06/12/11	0		
2	Disponibile	**UOMINI CON FIGURE (DOSIO) X MEDIA** Isbn: 9788828600251 - Autore: PIUMINI	EINAUDI SCOL.	9,04	1	18/11/11	1	06/12/11	0		0		
3	Ordine	**MONDO DI MALU X MEDIA** Isbn: 9788845051722 - Autore: ARGILLI	FABBRI SCOL.	8,93	1	05/12/11	0		0		0		Copertina
4	Pagato	**PROGETTO EDUCAZIONE TECNICA BIBL.1** Isbn: 9788839506719 - Autore: AA.VV.	PARAVIA	4,29	1	18/11/11	1	01/12/11	1	05/12/11	1	05/12/11	
5	Ordine	**NUOVO UNIVERSO DI SEGNI 2** Isbn: 9788835091035 - Autore: FINAMORE	SCUOLA (LA)-STUDIUM	10,48	1	05/12/11	0		0		0		Copertina

Figura 3.2-Schermata ordini libri utenti

Come si può vedere l'utente ha la situazione dei libri disponibili allo scaricamento, di quelli già ritirati e pagati. Voglio sottolineare, che lo scaricamento è un servizio aggiuntivo che si dà al cliente in quanto lo si vuole abituare gradualmente alla consultazione dei libri in questo formato elettronico, dandogli sempre di default anche la copia cartacea del libro. Facendo doppio click sullo stato "Disponibile" parte lo scaricamento del contenuto digitale del titolo del libro corrispondente, portando lo stato del libro a "Ritirato".

Capitolo 4

Conclusioni

L'obiettivo principale del tirocinio ha riguardato lo sviluppo di una parte del progetto Libropiù: la registrazione dell'utente al servizio del libro digitale per poter scaricare il libro. Questo tipo di progetti (come è lo proprio il progetto Libropiù) che sfruttano l'interazione con un database, sono un aspetto fondamentale nel mondo della rete in quanto, le applicazioni che ne derivano, vengono utilizzate nelle più comuni operazioni che si fanno in internet, come ad esempio l'acquisto di un biglietto aereo o il controllo delle proprie operazioni bancarie.
L'uso dell'e-book ("electronic book", il libro in formato digitale) si presta ad essere uno strumento fondamentale di lettura e consultazione di documenti nella rete, permettendo la rapidità di ricerca dei contenuti desiderati accendendovi comodamente da un tablet pc, o da un dispositivo in grado di connettersi ad internet (Smartphone, IPhone).
Il linguaggio utilizzato ha permesso di ridurre il tempo di sviluppo del progetto grazie alla sua varietà di framework messi a disposizione al programmatore. Soprattutto, grazie al paradigma di programmazione dell'Objective-C, si è potuto sfruttare bene il concetto della programmazione ad oggetti creando delle classi apposite per la gestione dei vari moduli del progetto. Grazie a questo approccio si ha avuto modo di poter scrivere delle procedure solide che nella fase di collaudo hanno risposto positivamente quasi a tutti i test eseguiti e si procederà a eventuali correzioni dei bug non appena nei prossimi mesi si riscontreranno problemi particolari nella messa in vendita.
Già presa in considerazione è la cooperazione con importanti case editrici come la Mondadori e la Zanichelli, in modo da garantire il servizio nella maniera più efficiente possibile. Le possibilità di sviluppo di questo progetto sono molto ampie, in quanto l'intero settore di progettazione che riguarda applicazioni che sfruttano la rete è in continua evoluzione. Per quanto riguarda il progetto, sono previsti alcuni ampliamenti tra i quali lo sviluppo su piattaforma Android e Windows Phone in previsione dell'incremento della vendita di dispositivi con questi sistemi operativi. Possibili ulteriori sviluppi riguardano sistemi di distribuzione dei contenuti "in loco" ossia direttamente all'interno delle librerie attraverso tecnologie wireless. In questo caso per massimizzare la velocità di scaricamento, considerando anche il contemporaneo acquisto di più libri da più persone, si dovranno valutare sistemi di storage locali su disco.

Appendice A

Storia del linguaggio Objective-C

Nei primi anni ottanta, la pratica comune dell'ingegneria del software era basata sulla programmazione strutturata. Questa modalità era stata sviluppata per poter suddividere programmi di grandi dimensioni in parti più piccole, principalmente per facilitare il lavoro di sviluppo e di manutenzione del software. Ciononostante, col crescere della dimensione dei problemi da risolvere, la programmazione strutturata divenne sempre meno utile, dato che conduceva alla stesura di un numero sempre maggiore di procedure, ad uno spaghetti code e ad uno scarso riuso del codice sorgente.

L'Objective-C fu creato principalmente da Brad Cox e Tom Love all'inizio degli anni ottanta alla Stepstone. Entrambi erano stati introdotti a Smalltalk durante la loro permanenza al Programming Technology Center della ITT Corporation nel 1981. Cox aveva iniziato ad interessarsi ai problemi legati alla riusabilità del software e si accorse che un linguaggio simile a Smalltalk sarebbe stato estremamente valido per costruire potenti ambiente di sviluppo per i progettisti di ITT. Cox iniziò così a modificare il compilatore C per aggiungere alcune delle caratteristiche di Smalltalk. Ottenne così ben presto una implementazione funzionante di una estensione ad oggetti del linguaggio C che chiamò OOPC (Object-Oriented Programming in C). Nel frattempo Love fu assunto da Schlumberger Research nel 1982 ed ebbe l'opportunità di acquisire la prima copia commerciale di Smalltalk-80 che influenzò in seguito lo sviluppo della loro creatura. Per dimostrare che il linguaggio costituiva un reale progresso, Cox mostrò che per realizzare componenti software intercambiabili erano necessari pochi adattamenti pratici agli strumenti già esistenti. Nello specifico, era necessario supportare gli oggetti in modo flessibile con un insieme di librerie software che fossero usabili e consentissero al codice sorgente (e ad ogni risorsa necessaria al codice) di essere raccolto in un solo formato multipiattaforma. Cox e Love formarono infine una nuova impresa, la Productivity Products International (PPI), per commercializzare il loro prodotto che accoppiava un compilatore Objective-C con una potente classe di librerie.

Nel 1986 Cox pubblicò la sua descrizione dell'Objective-C nella sua forma originale nel libro *Object-Oriented Programming, An Evolutionary Approach*. Sebbene fosse attento a puntualizzare che la questione della riusabilità del software non poteva essere esaurita dal linguaggio di programmazione, l'Objective-C si trovò spesso ad essere confrontato, caratteristica per caratteristica, con gli altri linguaggi.

A.1 Caratteristiche del linguaggio

Objective-C è il principale linguaggio utilizzato per chi sviluppa applicazione native per iOS. E' dinamico perché posticipa a runtime la maggior parte delle azioni sugli oggetti. La conoscenza di come trattare a runtime gli oggetti deriva dal Runtime: una libreria che viene staticamente linkata ad ogni programma Objective-C. In sostanza il Runtime agisce come una specie macchina virtuale in cui "vivono" gli oggetti Objective-C. Il Runtime mette a disposizione delle API che consentono ad esempio di conoscere per ogni oggetto a quale classe appartiene, i metodi e le proprietà che possiede; queste caratteristiche avvicinano più Objective-C a linguaggi tipo il Ruby o Python piuttosto che C++. Essendo, quindi, un superset di C in Objective-C si possono tranquillamente utilizzare tutti i metodi e le funzioni del C in maniera nativa. Ne consegue che è possibile compilare un qualsiasi programma scritto in C con un compilatore Objective-C. Esso è la base da apprendere per utilizzare le librerie che Apple mette a disposizione e che consentono lo sviluppo di applicazione su :

- OSX
- iPhone
- iPodTouch

L'IDE più utilizzato per programmare con questo linguaggio è XCode, uno strumento compatto e potente che mette a disposizione moltissime risorse utili al programmatore.

Un pò di termini:

Classe: è una descrizione astratta, un prototipo che definisce il comportamento di un oggetto.

Oggetto: è un'entità di una classe con un proprio stato e comportamento.

L'Objective-C richiede che l'interfaccia e l'implementazione di una classe siano dichiarati in blocchi di codice differenti. Per convenzione l'interfaccia è messa in un file con suffisso ".h", mentre l'implementazione in un file con suffisso ".m".
L'interfaccia descrive le azioni (i metodi) della classe e nasconde l'implementazione che definisce come i metodi vengono realmente eseguiti.

Quindi per scrivere una applicazione in prima approssimazione ho bisogno di due file:

Estensione del file	**Funzionalità**
Miofile.h	Definisce l'interfaccia con dichiarazioni e metodi
Miofile.m	Definisce come vengono implementati i metodi di interfaccia

A.2 Definizione di classi, interfacce, metodi

Indicazioni operative di utilizzo del linguaggio.

Dichiarazione di un'interfaccia(NomeDellaClasse.h):

```
1.  //definizione dell'interfaccia: "NomeDellaClasse.h"
2.  #import "NomeDellaSuperclasse.h"
3.  @interface NomeDellaClasse : NomeDellaSuperclasse
4.  {
5.  //variabili d'istanza
6.  int variabileIntera;
7.  float variabileFloat;
8.  ...
9.  }
10. //metodi di classe
11. + metodoDiClasse1
12. + metodoDiClasse2
13. + ...
14. //metodi di istanza
        - metodoDiIstanza1
        - metodoDiIstanza2
        - ...
15. @end
```

Il segno meno (-) denota i metodi d'istanza, mentre il segno più (+) quelli di classe (analoghi alle funzioni statiche del C++).
Dichiarazione dell'implementazione(NomeDellaClasse.m):

```
1.  #import "NomeDellaClasse.h"
2.  @implementation NomeDellaClasse
3.  + metodoDiClasse1
4.  {
5.  // implementazione
6.  ...
7.  }



8.  + metodoDiClasse2
9.  {
10. // implementazione
11. …
12. }

13. -metodoDiIstanza1
14. {
15. // implementazione
16. ...
17. }
18. -metodoDiIstanza2
19. {
20. // implementazione
21. ...
```

```
}
...
@end
```

Si può riassumere il **ciclo di vita** di un oggetto nel seguente modo:

- Alloc: viene allocata la memoria necessaria a predisporre l'oggetto.
- Init: viene inizializzato l'oggetto invocando il suo costruttore (l'oggetto prende vita)
- Use: utilizzo dell'oggetto
- Dealloc: L'oggetto viene deallocato dalla memoria.

La gestione di un ciclo di vita di un oggetto, si basa sull'utilizzo di un contatore che tiene traccia di quanti oggetti esterni hanno un riferimento ad esso. Ogni volta che si crea un riferimento ad esso il contatore si incrementa, quando viene rimosso si decrementa. Se il contatore raggiunge lo zero l'oggetto viene rimosso dalla memoria. Da notare che è lo stesso metodo utilizzato dai router in una rete per una errata ricezione del pacchetto tra gli host che la compongono. Questo meccanismo è noto come **retain/release**.

A.3 Implementazione della registrazione di un nuovo utente al book store

File *RegisterLoginView.m:*

```
1. #import "RegisterLoginView.h"
2. #import "NuovoUtenteView.h"
3. @interface RegisterLoginView ()
4. @end
5. @implementation RegisterLoginView
6. @synthesize NuovoUtenteButton;
7. @synthesize LoginButton;
8. @synthesize NuovoUtenteView;
9. @synthesize UserName;
10. @synthesize Password;
11. -(id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
12. {
13. self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
14. if (self) {
15. // Custom initialization
16. }
17. return self;
18. }
19. (void)viewDidLoad
20. {
21. [super viewDidLoad];
22. NuovoUtenteButton.layer.cornerRadius = 5; // this value vary as per your desire
23. NuovoUtenteButton.clipsToBounds = YES;
24. LoginButton.layer.cornerRadius = 5; // this value vary as per your desire
25. LoginButton.clipsToBounds = YES;
26. }
27. (void)didReceiveMemoryWarning
28. {
29. [super didReceiveMemoryWarning];
30. // Dispose of any resources that can be recreated.}
31. -(IBAction) NuovoUtenteButtonPush: sender {
32. if (self.NuovoUtenteView == nil) {
33. self.NuovoUtenteView = [[NuovoUtenteView alloc] initWithNibName:@"NuovoUtenteView"
    bundle:nil];
34. }
35. [self.navigationController pushViewController:self.NuovoUtenteView animated:YES];}
36. -(IBAction) LoginButtonPush: sender {
37. if (([UserName.text isEqualToString:@""]) || ([Password.text isEqualToString:@""])) {
38. UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Attenzione"
39. message:@"Inserire un Username e Password validi."
40. delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
41. [alert show];
42. }
43. else {
44. UIActivityIndicatorView* spinner = [[UIActivityIndicatorView alloc]
    initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhite];
45. [spinner startAnimating];
```

```
46. if (![NetworkHelper isConnectedToNetwork])
47. {
48. UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Assenza di Connessione"
49. message:@"Per accedere al servizio e' necessaria una connessione di rete. Si prega di utilizzare una connessione WiFi o 3G."
50. delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
51. [alert show];
52. }
53. NSString *url = [NSString stringWithFormat: @"%@json/login.php?u=%@&p=%@", URLLibropiu, UserName.text, Password.text];
54. NSString *retJson = [NetworkHelper getStringFrom:url usingVerb:kVerb_GET withParams:nil headers:nil];
55. NSDictionary *retlogin = [retJson JSONValue];
56. NSString *ret = [retlogin objectForKey:@"ret"];
57. if ([ret isEqualToString:@"0"]) {
58. UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Errore di Login"
59. message:@"Username o Password errati"
60. delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
61. [alert show];}
62. else {
63. arrayLogin = retlogin;
64. NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory , NSUserDomainMask, YES);
65. NSString *documentsDir = [paths objectAtIndex:0];
66. NSString *fullPath = [documentsDir stringByAppendingPathComponent:@"login.plist"];
67. BOOL fileExists = [[NSFileManager defaultManager] fileExistsAtPath:fullPath];
68. [retlogin writeToFile: fullPath atomically:YES];
69. if (fileExists) {
70. [[self navigationController] popViewControllerAnimated:YES];
71. }
72. Else
73. {
74. [[self navigationController] popViewControllerAnimated:YES];
75. }
76. }
77. }}@end
```

A.4 Implementazione del controllo di nuovi ordini di un utente

```
1. (void) ControllaOrdiniNuovi
2. { LabelUtente.text = [NSString stringWithFormat: @"Bentornato: %@", [arrayLogin
   objectForKey:@"nom_cli"]];
3. if ([NetworkHelper isConnectedToNetwork])
4. { NSString *url = [NSString stringWithFormat: @"%@json/contaordini.php?id=%@",
   URLLibropiu, [arrayLogin objectForKey:@"id"]];
5. NSError* error = nil;
6. NSString *conta = [NSString stringWithContentsOfURL:[NSURL URLWithString:url]
   encoding:NSUTF8StringEncoding error:&error];
7. int larghezza = 8 + (10 * conta.length);
8. [UIApplication sharedApplication].networkActivityIndicatorVisible = NO;
9. CGRect rect= self.AreaDownloadButton.frame;
10. rect.origin.x = rect.origin.x + rect.size.width - 12;
11. rect.origin.y = rect.origin.y - 4;
12. rect.size.width = larghezza;
13. rect.size.height = 18;
14. if (self.OrdiniAttivi == nil)
15. {  self.OrdiniAttivi = [[UILabel alloc ] initWithFrame:rect ];
16. [self.OrdiniAttivi setAdjustsFontSizeToFitWidth:YES];
17. self.OrdiniAttivi.layer.cornerRadius = 10;
18. self.OrdiniAttivi.layer.borderWidth = 2;
19. self.OrdiniAttivi.layer.borderColor = [UIColor whiteColor].CGColor;
20. self.OrdiniAttivi.backgroundColor = [UIColor redColor];
21. self.OrdiniAttivi.textColor = [UIColor whiteColor];
22. self.OrdiniAttivi.font = [UIFont fontWithName:@"HelveticaNeue-Bold" size:(15.0)];
23. self.OrdiniAttivi.textAlignment = NSTextAlignmentCenter;
24. [self.view addSubview:self.OrdiniAttivi];
25. }
26. else {
27. self.OrdiniAttivi.frame = rect;
28. }
29. if ([conta isEqualToString: @"0"]) {
30. self.OrdiniAttivi.hidden = true;
31. }
32. else {
33. self.OrdiniAttivi.hidden = false;
34. self.OrdiniAttivi.text = conta;
35. }
36. }
```

A.5 Implementazione del caricamento della libreria per un utente

```
1. -(void) CaricaLibreria
2. {
3. LibriAcquistati = [[NSMutableArray alloc] init];
4. int index = 0;
5. int x = OFFSETXCOPERTINA;
6. int y = 0;
7. int w = LARGHEZZACOPERTINA;
8. int h = ALTEZZACOPERTINA;
9. int riga = 0;
10. int indexriga = 0;
11. int testindex = 0;
12. if (self.interfaceOrientation == UIInterfaceOrientationPortrait ||
    self.interfaceOrientation ==UIInterfaceOrientationPortraitUpsideDown) {
13. testindex = NRCopertineVerticale;
14. }
15. else {
16. testindex = NRCopertineOrizzontale;
17. }
18. sqlite3 *db;
19. if(sqlite3_open([[DBFunction databasePath] UTF8String], &db) == SQLITE_OK)
20. {
21. sqlite3_stmt *stmt
22. NSString *query = @"SELECT * FROM downloads";
23. const char *insert_stmt = [query UTF8String];
24. NSString *documentsDirectory =
    [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
    NSUserDomainMask, YES) objectAtIndex:0];
25. NSString *imagepath = [documentsDirectory
    stringByAppendingPathComponent:@"copertine"];
26. if(sqlite3_prepare_v2(db, insert_stmt, -1, &stmt, NULL) == SQLITE_OK)
27. {
28. NSMutableString *cod_isb;
29. NSMutableString *tit_ope;
30. NSMutableString *aut_001;
31. NSMutableString *des_edi;
32. NSMutableString *nom_fil;
33. while(sqlite3_step(stmt) == SQLITE_ROW)
34. {   cod_isb = [NSString stringWithUTF8String:(char *)sqlite3_column_text(stmt, 1)];
35. tit_ope = [NSString stringWithUTF8String:(char *)sqlite3_column_text(stmt, 2)];
36. aut_001 = [NSString stringWithUTF8String:(char *)sqlite3_column_text(stmt, 3)];
37. des_edi = [NSString stringWithUTF8String:(char *)sqlite3_column_text(stmt, 4)];
```

```
38. nom_fil = [NSString stringWithUTF8String:(char *)sqlite3_column_text(stmt, 13)];
39. NSMutableDictionary *rigalibro = [[NSMutableDictionary alloc] initWithObjectsAndKeys:cod_isb, @"cod_isb", tit_ope, @"tit_ope", aut_001, @"aut_001", des_edi, @"des_edi", nom_fil, @"nom_fil", nil];
40. [LibriAcquistati addObject:rigalibro];
41. rigalibro = nil;
42. //Load Image From Directory UIImage *imageCopertina = [Funzioni loadImage:[NSString stringWithUTF8String:(char *)sqlite3_column_text(stmt, 1)] ofType:@"jpg" inDirectory:imagepath];
43. if (indexriga > 0) x = x + OFFSETXCOPERTINA + w;
44. y = OFFSETYCOPERTINA + (riga * (h + OFFSETYCOPERTINA));
45. UIButton *buttonCopertina = [UIButton buttonWithType:UIButtonTypeCustom];
46. buttonCopertina.frame = CGRectMake(x, y, w, h);
47. [buttonCopertina setImage:imageCopertina forState:UIControlStateNormal];
48. [buttonCopertina.imageView setContentMode:UIViewContentModeScaleAspectFit];[buttonCopertina setTag:TAGButtonCopertine + index];
49. [buttonCopertina addTarget:self action:@selector(buttonCopertinaClicked:) forControlEvents:UIControlEventTouchUpInside];
50. [self.scrollView addSubview:buttonCopertina]
51.  index++;
52. indexriga++;
53. if (indexriga >= testindex) {
54. indexriga = 0;
55.  riga++;
56. x = 10;
57. }
58. }
59. sqlite3_finalize(stmt);
60. }
61. }
62. sqlite3_close(db);
63. }
```

Bibliografia

[1] Wikipedia: La teoria della codifica salt, http://it.wikipedia.org/wiki/Sale_(crittografia)

[2] Steven F. Daniel, Xcode 4 iOS Development Beginner's Guide, Packt Publishing Limited

[3] Massimiliano Bossi : Guida PHP, http://www.mrwebmaster.it/php/guide/guida-php/

[4] Maurizio Codogno : Salt e Hashing, http://www.ilpost.it/mauriziocodogno/

[5] Wikipedia : Storia del linguaggio Objective-C, http://it.wikipedia.org/wiki/Objective-C

[6] Gabliele Gigliotti : HTML 5 e CSS 3, Apogeo (2011)

[7] Andi Gutmans, Stig Bakken, Derick Rethans : PHP 5 Guida completa, Apogeo, (2005)

[8] Marco Cantù, Peter W. A. Wood, Derick Rethans : A Guide to the New Features of Delphi 2010, Brossura, (2010)

[9] The IEEE P1363 Home Page: Standard Specifications For Public-Key Cryptography, http://grouper.ieee.org/groups/1363/

[10] Chiavi pubbliche, private e derivazioni della chiave (KDF), http://iw1gli.altervista.org/guide/guida1/chiavi.html