UNIVERSITA' DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea in Ingegneria Informatica

QUESTIONARI UNIWEB: UN APPROCCIO PER PIATTAFORMA IOS

Laureando

Gheorghe Alexandra

Relatore

Dott. Carlo Fantozzi

ANNO ACCADEMICO 2011/2012

Un ringraziamento a tutte le persone che hanno reso possibile la realizzazione di questo progetto: Alessandro Furlati, Fabrizio Bassi e Ali Donmez, collaboratori del Cineca e al mio relatore Carlo Fantozzi.

Un ringraziamento anche alla mia famiglia che mi ha sempre incoraggiato e aiutato e a tutte le persone che mi hanno sostenuto.

Indice

Capitolo 1

SOMMARIO

Negli ultimi anni hanno avuto uno sviluppo importante gli smartphone e i loro sistemi operativi, e con essi ha preso vita anche il concetto di "app", in modo da soddisfare ogni esigenza del consumatore. Di conseguenza molti servizi disponibili sul web hanno dovuto migrare nel ramo mobile.

In questo elaborato si presenterà l'mplementazione di un servizio UNIWEB su piattaforma iOS. In particolare si presenterà la realizzazione di un modulo che andrà integrato nell'app Uniweb.

UNIWEB è un servizio informativo messo a disposizione nei vari atenei italiani, di cui possono usufruire sia gli studenti che i docenti. Il sistema permette agli studenti di accedere a vari servizi, come iscrizione agli esami, visualizzazione del libretto online o richiesta per la domanda di borsa di studio via web, utilizzando username e password della posta elettronica di Ateneo. Analogamente i docenti accedono al servizio per aprire e chiudere le liste di esame oppure per pubblicare e verbalizzare i vari esiti degli esami. Tra le funzionalità c'è anche la possibilità di compilare i questionari Uniweb, questionari realizzati da Cineca in colaborazione con i vari Atenei per poter raccogliere informationi sui servizi offerti da quest'ultimi.

Il modulo sviluppato gestisce il servizio sui questionari Uniweb direttamente su dispositivi mobili Apple.

Capitolo 2

INTRODUZIONE

2.1 Che cosa è UNIWEB

L'università è un istituto di formazione superiore con origini antiche (Accademia platonica) che con il trascorre degli anni ha saputo tenere il passo con l'evoluzione della società umana.

Al giorno d'oggi, in piena era della tecnologia,viviamo circondati da computer, smartphone e connessioni internet ed ancora una volta l'Università si è rivelata all'altezza, introducendo i servizi informatici.

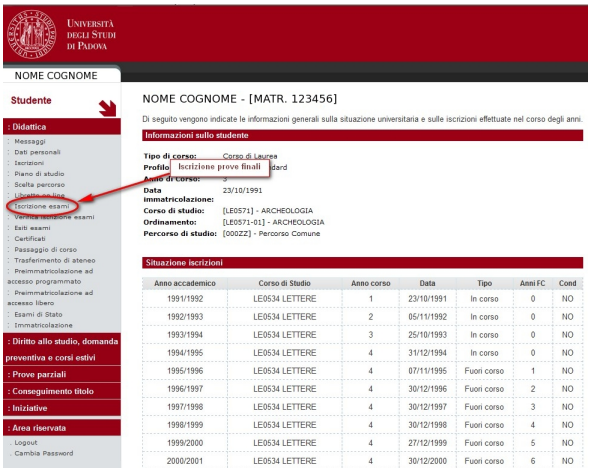
Si è trattato di un passo molto importante: l'introduzione dei servizi informatici ha prodotto una gestione centralizzata del patrimonio di dati e della programmazione didattica. Essa ha dato più trasparenza, velocità e facilità d'accesso ai servizi tradizionali e la possibilità di introdurne altri di innovativi. In questo modo si sono create banche di dati contenenti i dati degli studenti, professori e attività didattiche da essi svolte per poter avere un controllo migliore della gestione degli utenti e dei vari servizi offerti dall'università.

Nel 1969 quattro università italiane, tra cui anche l'Università Padova hanno costituito un "Consorzio interuniversitario per il Calcolo Automatico dell'Italia Nord-orientale",chiamato Cineca, con lo scopo di creare una struttura dedicata al supercalcolo.

Attualmente il consorzio è formato da 54 università, enti nazionali e dal Ministero dell'Università e della Ricerca. Con il passare degli anni le attività del consorzio si sono articolate nel settore del trasferimento tecnologico, nella gestione e nello sviluppo di reti e servizi telematici e nella produzione di sistemi informativi rivolti al mondo universitario.

Cineca ha creato un'azienda, Kion, dedicata alla progettazione e realizzazione di applicazioni nell'area dei sistemi informativi. Attualmente Kion è diventata la software factory del Consorzio e si occupa dei sistemi per la gestione della didattica e dei servizi agli studenti universitari.

Cineca e Kion seguono e sostengono costantemente l'evoluzione del mondo accademico e in questo modo ha preso vita il progetto Uniweb.



Schermata anonima del portale Uniweb visualizzata da uno studente

Uniweb è un prodotto nato per offrire un sistema per la gestione della programmazione didattica, per la valutazione e il controllo della qualità della formazione. Sostiene lo sviluppo da parte dell'Ateneo di nuovi servizi multimediali (via web, email) per docenti e studenti facilitando l'integrazione con i servizi già esistenti ed il patrimonio di dati. Inoltre promuove l'integrazione tra la piattaforma dei servizi web e sistemi di e-learning. Esso mette a disposizione strumenti per amministrare i programmi di mobilità internazionale per studenti/ docenti e sistemi che affiancano l'Ateneo nelle attività di orientamento degli studenti diplomati.

Per uno studente, Uniweb è il "sistema gestionale" della sua carriera accademica, anche per quanto riguarda tasse e mobilità internazionale.

Facendo una panoramica delle funzionalità offerte agli studenti troviamo, nell'aria "Didattica", la voce "Dati personali", dove lo studente può modificare direttamente online i suoi dati, ad esempio in caso di cambiamento di residenza; le voci "Piano di studio" e "Scelta percorso" presentano l'elenco degli esami obbligatori ed i vari esami a scelta. La voce "Libretto online" mostra una copia del suo libretto con i vari esami già sostenuti e verbalizzati. In stretta relazione con quest'ultima troviamo "Iscrizione esami" e "Esiti esami". Per potersi iscrivere ad uno dei esami presenti nel suo libretto, lo studente deve prima compilare il questionario della Valutazione didattica. Inoltre tramite Uniweb può gestire i suoi certificati, eventuali passaggi di corso, trasferimenti di ateneo oppure l'esame di stato. Tramite area "Diritto allo studio e corsi estivi" lo studente può accedere alle tasse, autocertificazione(ISEE) o corsi estivi. Il sistema prevede un'area per la mobilità internazionale e una per la domanda di conseguimento del titolo.

Come ci saremmo aspettati, Uniweb presenta anche un'area per i questionari, dove lo strudente può compliare i questionari per la valutazione di fine anno oppure i questionari per Erasmus.

In conclusione possiamo affermare che Uniweb rappresenta il punto di incontro tra lo studente ed il suo Ateneo.

2.2 Che cosa è un questionario

2.2.1 Questionario

Che cos'è un questionario? La definizione di questionario è la seguente: per questionario si intende una "serie di domande scritte, poste su un determinato argomento e a scopo di indagine o ricerca; il modulo in cui tali domande sono formulate. "

Il questionario consente la misurazione di un fenomeno in modo quantitativo e di conseguenza deve essere standardizzato.

- Standardizzazione degli stimoli: le domande di un questionario sono poste nello stesso ordine e con gli stessi termini a tutti i soggetti. Questa caratteristica permette di raccogliere in maniera uniforme le informazioni sui temi oggetto di indagine e di confrontare le risposte tra loro.

- Strutturazione delle risposte: i questionari strutturati sono costituiti da domande e risposte.

In altre parole il questionario è uno strumento di raccolta delle informazioni, definito come un insieme strutturato di domande. Principalmente un questionario può essere di tre tipi: *chiuso*, quando è possibile scegliere una o più risposte fra quelle indicate; *aperto*, quando la risposta non è predeterminata, ma viene lasciato uno spazio in cui è possibile indicare liberamente una risposta, che può essere di tipo numerico oppure testuale; ed infine *scalato*, quando la risposta viene indicata su una scala graduata, che pone ai suoi estremi due risposte totalmente opposte fra di loro e in cui la scelta dell'intervistato viene espressa scegliendo un valore della suddivisione della scala in cui la preferenza gradita corrisponde alla corretta distanza dagli estremi della scala.

Il poeta Johann Wolfgang Goethe affermava che " Comunicare l'un l'altro, scambiarsi informazioni è natura; tenere conto delle informazioni che ci vengono date è cultura", quindi avere delle informazioni e utilizzarle non può che produrre un miglioramento costante.

Avendo a disposizione uno strumento come i questionari l'Ateneo può facilmente ottenere informazioni sui servizi offerti. Conoscendo l'importanza di questo strumento, Cineca ha deciso di utilizzarlo in modo da poter raccogliere più informazioni per poter affiancare al meglio gli Atenei, analizzando le risposte dei campioni intervistati. Il valore di questo strumento non deve essere sottovalutato, in quanto tramite di esso l'Ateneo ottiene informazioni riguardo il servizio che esso offre e può quindi evolvere tenendo conto anche delle richieste degli studenti.

2.2.2 Descrizione della struttura di un questionario in Uniweb

Il potere di un questionario sta nel fatto che esso non è solo uno strumento di misura qualitativo, ma anche uno strumento di comunicazione.

Prima di costruire un questionario si devono stabilire gli argomenti e si deve fare un'analisi sul dove devono essere collocate le varie domande nel questionario in modo da evitare di creare ambiguità al rispondente. Per ottener il risultato voluto le domande devono rispettare l'ordine stabilito nella sequenza degli argomenti in modo da evitare il condizionamento della risposta e distorsioni dei dati ottenuti, per esempio se si vuole un'opinione spontanea sulla soddisfazione nel lavoro è bene non anteporre domande sulle caratteristiche specifiche del lavoro svolto che potrebbero focalizzare l'attenzione su alcuni aspetti particolarmente gradevoli o sgradevoli.

Il linguaggio utilizzato per componere il questionario è un aspetto critico per la riuscita di un questionario.

Consapevoli del potere dell'informazione ottenuta tramite questo strumento, Uniweb contiene un'intera area dedicata ai questionari. Per preparare i questionari offerti agli studenti è stato fatto uno case study per poter rende il più utile possibile questo strumento. In quanto dai risultati forniti dai vari questionari si possono ricavare nuove problematiche nel sistema o fornire nuovi ambiti di indagine.

Gli argomenti pricipali che vengono trattati nei questionari presenti su Uniweb sono : la valutazione didattica sui corsi proposti dall'Ateneo, la valutazione di fine anno e la valutazione sull'esperienza Erasmus.

Il modulo dei questionari è una serie di funzionalità di ESSE3 orientate alla configurazione, alla gestione e alla compilazione di questionari elettronici sul web dell'Ateneo. Attualmente un modulo di questo tipo funziona soltanto sulla piattaforma web e l'implementazione sulle piattaforme mobile non è stata ancora realizzata.

Ovviamente, dato che un questionario può essere applicato a tutte le funzionalità gestite dal portale di Ateneo, la caratteristica di tali questionari è la capacità di avere il massimo di configurabilità possibile, sia come dati di input, sia come strutturabilità interna, sia come dati di output, ossia tutti quei dati che possono essere collegati ad un questionario compilato e oggetto di statistiche successive.

Questo modulo viene realizzato da Kion in stretta colaborazione con gli Atenei, in modo da addattarsi perfettamente alla situazione dell'università. Tutti gli Atenei che si appoggiano al servizio Uniweb presentano un modulo dei questionari, però il contenuto dei questionari varia da università a università.

2.2.2.1 Esempio: il questionario della valutazione della didattica

Il questionario della valutazione didattica è in assoluto lo strumento più adatto per valutare dal punto di vista qualitativo e quantitativo un corso universitario messo a dispozione dall'Ateneo.

Questo questionario è uguale per tutti i corsi accademici proposti da un certo Ateneo, in modo da renderli confrontabili, per tutti gli intervistati, in modo da poter ottenere un quadro generale della situazione. L'intervistato in questo caso è lo studente, però la sua identità resta anomima, in modo da non metterelo in soggezione. Il questionario per un corso compare su Uniweb alla chiusura ufficiale del corso e la sua compilazione è obbligatoria, in quanto blocca l'iscrizione dell'utente all'appello del corso.

Il quadro che questo tipo di questionario offre è molto dettagliato in quanto esso viene visualizzato in modo dinamico, cioè certe risposte posso portare alla visualizzazione di nuove domande o all'opposto nascondere certe domande. Ad esempio se uno studente alla domanda "Hai frequentato questo insegnamento?" risponde con "Non ho mai frequentato il corso." le domande del tipo "Le conoscenze preliminari da te possedute sono risultate sufficienti per la comprensione degli argomenti trattati nel corso?" non vengono visualizzate per evitare di avere delle risposte casuale e quindi avere un'opinione più accurata e affidabile.

2.3 La piattaforma iOS

Logo della piattaforma iOS

2.3.1 Introduzione e storico

Apple Inc. (più comunemente conosciuta come Apple) è un'azienda informatica statunitense che produce sistemi operativi, computer e dispositivi multimediali, tra qualli il lettore di musica digitale iPod, lo smartphone iPhone ed il tablet iPad. L'iPhone e l'iPad usano un sistema operativo sviluppato internamente, chiamato iOS.

Il sistema operativo iOS (fino a giugno 2010 iPhone OS) è una derivazione di UNIX (famiglia BSD) e usa un microkernel XNU Mach basato su Darwin OS.Esso occupa meno di mezzo gigabyte della memoria interna del dispositivo. Il sistema operativo è stato rilasciato per la prima volta nel giugno 2007 e non aveva un nome ufficiale fino al rilascio della prima beta dell'iPhone SDK il 6 marzo 2008. Esso è implementato sui dispositivi iPod Touch, iPhone e iPad.

2.3.2 Tool disponibili

Nel 2008 è stato rilasciato un SDK (software development kit) che permette agli sviluppatori di creare applicazioni per iPhone e iPod touch, e testarle in un simulatore dei dispositivi.

All'interno dell'SDK è contenuto l'iPhone Simulator, un programma usato per emulare il "look and feel" dell'iPhone nel desktop dello sviluppatore. Originariamente chiamato Aspen Simulator, è stato rinominato con la beta 2 dell'SDK. Da notare che l'iPhone Simulator non è un emulatore ed esegue codice generato per un target x86. L'SDK richiede un Mac Intel con Mac OS X Leopard. Altri sistemi operativi, inclusi Microsoft Windows e vecchie versioni di Mac OS X, non sono supportati.

Nell'SDK troviamo i seguenti strumenti che Apple mette a disposizione:

- DashCode , tool per creare pagine web, widget e altro;
- Instruments, che permette di analizzare e valutare le prestazioni delle nostre applicazioni;
- Interface builder, un tool grafico per realizzare interfacce grafiche;
- Quartz Composer, tool per creare animazioni ed effetti grafici;
- Xcode, l'ambiente di sviluppo vero e proprio.

Xcode è un ambiente di sviluppo integrato (Integrated development environment, IDE) sviluppato da Apple Inc. per agevolare lo sviluppo di software per Mac OS X e iOS. Estende e rimpiazza il precedente tool di sviluppo della Apple, Project Builder. Xcode lavora in congiunzione con Interface Builder. Esso include GCC, che è in grado di compilare codice C, C++, Objective C/C++ e Java. Una delle caratteristiche tecnologicamente più avanzate di Xcode è che supporta la distribuzione in rete del lavoro di compilazione. Usando Bonjour e Xgrid è in grado di compilare un progetto su più computer riducendo i tempi. Supporta la compilazione incrementale; è inoltre in grado di compilare il codice mentre viene scritto, in modo da ridurre il tempo di compilazione. Dalla versione 3.1, Xcode è anche lo strumento per sviluppare le applicazioni native per iPhone e iPod touch. Dalla versione 3.2, è possibile sviluppare applicazioni anche per iPad.

2.3.3 L'innovazione

Apple ha lanciato sul mercato il concetto di "app", dicitura abbreviata per indicare un'applicazione software per i suoi dispositivi mobile. L'azienda ha messo a disposizione un portale, "App Store" da dove gli utenti posso scaricare le app per i loro dispositivi mobile.

La piattaforma iOS è stata la prima piattaforma costruita intorno alle app. Essendo progettata in questo modo, essa presenta delle caratteristiche uniche che riescono ad offire all'utente un'esperienza memorabile.

Tra le caratteristiche principali che le app offrono c'è la risposta ai gesti e non ai click, come riportato anche sul sito ufficiale della Apple. L'interfaccia multi-touch dà agli utenti una sensazione di interazione totale con i loro dispositivi, sensazione che rafforza la sensazione di manipolazione diretta degli oggetti sullo schermo, in quanto i gesti da compiere sono semplici e molto intuitivi.

2.4 Obiettivo del lavoro

Lo scopo del progetto è quello di sviluppare un'applicazione nativa per IPhone e IPad, che dovrà gestire i questionari messi a disposizione dal sistema informativo Uniweb direttamente dal dispositivo mobile. L'applicazione dovrà essere completamente funzionante ed independente ed andrà integrata come modulo nella vera e propria app di Uniweb.

Essendo un'applicazione progettata per dispositivi Apple, dovrà essere costruita rispettando le caratteristiche principali che questa piattaforma offre in modo da poter sfruttare a pieno tutte funzionalità associate ad un'applicazione di questo genere. L'applicazione sarà sviluppata utilizzando come strumento Xcode e compilata in Objective C.

Siccome dobbiamo creare un questionario, la modalità più semplice e più pulita per visualizzarlo è utilizzare le UITableView, che rappresentano un mezzo per la visualizzazione e la modifica di elenchi gerarchici di informazioni che renderanno ll'app semplice, fluibile e efficiente.

L'applicazione avrà come target iOS 4 e quindi sarà installata su device con display retina e multi-touch che renderà la compilazione del questionario un'esperienza piacevole. Gli unici gesti che l'utente dovrà esseguire saranno "Tap", per selezionare e "Drag" per effettuare lo scroll.

Nei capitoli successivi dell'elaborato ci sarà una presentazione più dettagliata dei questionari Uniweb forniti da Cineca e Kion. Si farà una presentazione della struttura di questi questionari e le modalità con cui essi vengono gestiti nella casa madre. Ci sarà anche un capitolo dedicato alla realizzazione dell'applicazione sia dal punto di vista della progettazione sia da quello della costruzione vera e propria utilizzando il tool Xcode.

Capitolo 3

QUESTIONARI UNIWEB

Lo scopo di questo progetto è la realizzazione della gestione dei questionari Uniweb su dispositivi Apple.

La funzionalità di gestione dei questionari esiste già nel portale web di Uniweb, quindi essa deve essere solamente trasposta nelle piattaforme mobile.

Anche se le piattaforme web e mobile differiscono molto la logica di funzionamento e la struttura interna dei questionari non deve cambiare. La grande differenza consiste nella gestione della loro visualizzazione.

Tutti i dati dei questionari sono gestiti tramite un database che risiede su uno dei web service messi a disposizione da Kion.

Un web service è un sistema software progettato per supportare l'interoperabilità tra diversi elaboratori su di una medesima rete. Tra le caratteristiche principali che questo sistema ha possiamo elencare:

- **l'interoperabilità tra diverse applicazioni software su diverse piattaforme hardware;**
- **l'utilizzo di protocolli open sourse;**
- **l'utilizzo dei protocolli HTTP per il trasporto dei messaggi.**

Quando il web service viene interrogato sui questionari esso risponde con file JSON. I file JSON contengono un elenco ordinato di valori, rappresentabile nei vari linguaggi come un array, vettore, dizionario o sequeza.

Le informazioni sui questionari ed i questionari stessi sono gestiti tramite un database, i file JSON mandati dal web service come risposta ad una query contengono le righe di una tabella del db. Il software web che si occupa della gestione dei questionari, dopo aver ricevuto il file JSON, effettua il parse di esso e interpretando le informazioni visualizza il questionario.

Lo stesso schema logica deve seguiere anche l'applicazione mobile. L'app che andremo a creare deve appoggiarsi allo stesso web service, a cui si appoggia anche il portale web, quindi la struttura interna sarà la stessa.

3.1 Struttura interna dei questionari Uniweb

Il modulo dei questionari Uniweb presenti sul portale con lo stesso nome è una serie di funzionalità di ESSE3 orientate alla configurazione, alla gestione e alla compilazione. Il modulo che andrà a funzionare sui dispositivi mobile deve presentare le stesse funzionalità, quindi la struttura interna resterà la stessa.

Data l'applicabilità di un questionario generale, la struttuta interna di esso deve essere costruita seguendo delle regole generali, imposte da Kion in collaborazione con l'Ateneo in cui andrà visualizzato, dopo precendenti indagini.

Analizziamo ora i tipi generali di questionari e le proprietà associate ad essi.

3.1.1 Questionario Configurato:

Un questionario configurato è il questionario creato e strutturato per essere sottoposto alle future compilazioni. Questo questionario comprende:

- **Struttura generale:Testata + Righe.**
  - ➤ **La testata include: un codice univoco, una descrizione, un contesto di utilizzo, uno stato (bozza o attivo) ;**
  - ➤ **Le righe includono: una strutturazione gerarchica di elementi (i tasselli di un questionario) e proprietà varie degli elementi (formato, TAG, categorie, obbligatorietà, note, filtri applicati) :**
    - **PAGINE (ognuna contiene uno o più paragrafi) ;**
    - **PARAGRAFI (ognuno contiene una o più domande) ;**
    - **DOMANDE (ognuna contiene uno o più risposte) ;**
    - **RISPOSTE.**

3.1.2 Questionario Compilato:

Un questionario compilato è l'insieme delle risposte date a un questionario configurato da parte di un compilatore. In termini informatici, rappresenta una delle "istanze" possibili collegate alla "classe" del questionario configurato. Naturalmente, possono esistere N compilazioni di uno stesso questionario configurato. Tali compilazioni dipenderanno dal tipo di condizioni filtranti che avremo collegato al questionario configurato.

3.1.3 Metadati o TAGS:

I metadati o TAGS sono delle etichette di sistema a cui associare dei valori, numerici, data o alfanumerici, che racchiudono le informazioni che intendiamo collegare ai questionari. Possono essere suddivisi in:

• TAG DI CONFIGURAZIONE. Sono i tag che vengono collegati ad un questionario configurato. Possono essere definiti come i tag che servono a contestualizzare una configurazione. Ad esempio, per i questionari sui concorsi, ogni questionario è collegato con un concorso, quindi ognuno di essi sarà caratterizzato dalla chiave univoca del concorso che verrà collegata al questionario sotto forma dei 2 TAGS Anno accademico e ID del test. Per i questionari di valutazione della didattica, inoltre, dato che il questionario configurato è unico per ogni contesto di applicazione (ossia per ogni attività didattica oggetto di valutazione), non è stato necessario avere nessun tag di configurazione.

• TAG DI COMPILAZIONE. Sono i tag che vengono collegati ad un questionario compilato. Sono tutte quelle informazioni che ogni compilazione intende portarsi dietro a fini statistici e non di processo, come invece sono "concepiti" i tag di configurazione.

• TAG UTENTE (Log di compilazione). Sono i tag che sono collegati ad ogni soggetto compilatore. Sono le informazioni che ogni soggetto compilatore si porta dietro ogni volta che accede al sistema per effettuare una compilazione. Questi tag sono differenziati dai tag di Compilazione perché i questionari possono essere anonimi. In caso di anonimato, l'utente compilatore (e così anche tutte le informazioni collegate sotto forma di tags) sono fisicamente disgiunti, anche a livello di base dati, con il questionario che ha compilato (e quindi con tutte le informazioni collegate a quest'ultimo), in modo da non poter avere la possibilità di identificarne l'autore.

• TAG di ELEMENTO. Sono i tag che sono collegati ad un elemento di un questionario e possono essere utilizzati a fini statistici o tecnici (di solito questi ultimi). Sono usati generalmente nel significato più generico di TAG, ossia un etichetta che permette di identificare un elemento per includerlo in logiche di processo particolari.

3.1.4 Motore di compilazione generale:

Una volta avuto accesso alla compilazione del questionario (uno qualsiasi, ma nell'esempio parliamo del solito questionario di valutazione), entra in gioco il cosiddetto "motore di compilazione generale" che legge la configurazione del questionario e la traduce in qualcosa di logico e leggibile. La prima pagina contiene una domanda con risposte condizionanti per le pagine successive.

3.1.5 Parametri di configurazione dei questionari Uniweb:

Non esistono attualmente parametri di configurazione globali per tutti i tipi e contesti di questionari. Esiste però un parametro di configurazione standard che gestisce i questionari di valutazione della didattica. Questo parametro è CHECK_QUEST_VAL_DID, che serve a controllare la presenza dei questionari di valutazione per abilitare l'accesso alla prenotazione appelli.

In conclusione i questionari Uniweb che andranno visualizzati nell'applicazione dovranno rispettare questa configurazione. L'applicazione dovrà implementare le stesse funzionalità del software web che gestisce il portale Uniweb trasposte nel linguaggio Objective-C e adattate alla piattaforma mobile. Una delle conseguenza più importante sarà la gestione della graffica, il modo in cui il questionario andrà visualizzato sullo schermo del device mobile.

Capitolo 4

L'APPLICAZIONE iOS

4.1 Descrizione degli obiettivi

L'obiettivo del progetto è quello di realizzare un modulo da integrare nell'app di Uniweb, già in corso di sviluppo presso Kion. Questo modulo deve gestire i questionari proposti dal sistema informativo madre.

4.2 Situazione di partenza

L'analisi delle esigenze è stata frutto della collaborazione con Cineca e Kion, in quanto il loro scopo è di ampliare l'utilizzo del sistema Uniweb rendendolo disponibile anche sulle piattoforme mobile.

Nell'ottica di sviluppare un'app Uniweb, essi hanno scelto di gestire le varie funzionalità tramite moduli tra i quali anche il modulo che gestisce i questionari, che risulta essere il fulcro di questo elaborato.

Nel colloquio con Cineca sono state tracciate le funzionalatà principali che il modulo deve fornire. Per realizzare tale progetto Kion ha fornito un web service indispensabile per ottenere le informazioni necessarie allo sviluppo del progetto.

4.3 Requisiti del progetto

Dopo aver deciso le funzionalità che l'applicazione deve raggiungere è necessario fare uno schema generale dei requisiti del progetto.

Nell'applicazione dobbiamo implementare:

- le richieste HTTP al web service;
- l'interpretazione dei file JSON restituito dal web service;
- la costruzione del questionario seguendo la sua struttura interna di essi, proposta da Kion.

4.3.1 Le richieste HTTP

Per gestire le richieste dobbiamo utilizzare oggetti di tipo NSURL-Request, oggetti che rappresentano una richiesta di caricamento di un url, in modo independente dal protocollo utilizzato e dal formato dell'url.

NSURLRequest racchiude in sè due informazioni chiave associate a una richiesta di caricamento:

- url da caricare;
- la politica da utilizzare per consultare il contenuto della cache dell'url;

Questo oggetto è stata progettato per poter essere esteso in modo da supportare anche altri tipi di protocolli. Esiste anche la sottoclasse mutabile(modificabile) chiamata NSMutableURLRequest.

Una delle funzionalità essenziali che l'app deve fornire è l'interrogazione di un web service per ottenere i dati necessari. Questa funzionalità viene raggiunta costruendo una request HTTP, che ha come url la query utilizzata per l'interrogazione del web service.

Per eseguiere questa funzionalità la relazione tra device e web service deve essere di tipo client-server e l'app sarà il software che andrà a creare la richiesta ed interpretare la risposta.

Gestione della richiesta:

```
NSURL *url = [NSURL URLWithString: @"http://..."];
NSMutableURLRequest *request = [[NSMutableURLRequest alloc]
initWithURL:url cachePolicy:NSURLRequestReloadIgnoringCacheData
timeoutInterval:20];
```

Per definire l'url creiamo un oggetto di tipo NSURL, che riceve come parametro una stringa composta concatenando il nome del web service al quale dobbiamo accedere e la query da eseguire su di esso. Questo url è inserito come parametro nella nostra request. Nella richiesta è stato settato come timeoutInterval 20 secondi, cioè il tempo massimo che si può aspettare per una risposta è di 20 secondi.

Gestione della risposta:

Per gestire la risposta dobbiamo creare un oggetto di tipo NSData. Gli oggetti che fanno parte di questa classe sono oggetti dati, object-oriented wrappers per i buffer di byte. Utilizziamo anche un oggetto della classe NSURLConnection,che fornisce supporto per il caricamento della richiesta. Dopo che l'oggetto dataweb service è stato "riempito" con la risposta del web service la convertiamo in una stringa, oggetto appartenente alla classe NSString. Effettuiamo la conversione in una stringa per poter gestire al meglio la risposta, in quanto il web service ritorna file di tipo JSON.

```
NSData *dataWebService;
[dataWebService setLength:0];
dataWebService = [NSURLConnection sendSynchronousRequest:request
returningResponse:nil error:nil];
NSString *responseString;
responseString = [[NSMutableString alloc] initWithData:dataWebService
encoding:NSUTF8StringEncoding];
```

4.3.2 I file JSON

Il web service utilizzato dall'applicazione ritorna file di tipo JSON.

I file di tipo JSON vengono utilizzati per lo scambio dei dati in applicazioni client-server ed il nome è l'acronimo di JavaScript Object Notation. Questo tipo di file è molto utilizzato per la semplicità con quale si può effettuare il parsing e a causa della diffusione della programmazione in JavaScript nel mondo del Web. I file JSON vengono utilizzati come un'alternativa ai file xml. JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

JSON è basato su due strutture:

- Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
- Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Esempio di file JSON:

```
{ "type": "value1","value2": "item1", "item2": [ {"value": "pippo",
"action": "pluto"}, {"value": "popye", "action": "olive"} ] }
```

Visto la semplicità di questi tipi di file essi vengono utilizzati anche nelle applicazioni per IPhone. Ufficialmente non è stato rilasciato alcun framework che si occupa della gestione di questo tipo di file, però in rete troviamo progetti, che poi riportati nel nostro si comportano come una libreria vera e propria. Un esempio di un tale progetto può essere scaricato dal social network dei programmatori www.github.com. Utilizzare questo tipo di libreria non viene considerato come codice non valido nei controlli eseguiti da Apple nel caso in cui l'app andrà sul market.

Nel nostro progetto è stato aggiunta una cartella che contiene le classi che si occupano della gestione dei file di questo tipo.

Nella sezione precedente avevamo converito la risposta da NSData a NSString proprio per poter gestire ancora con più facilità i dati. Per fare il parsing della risposta basta fare sfruttare una delle proprietà messe a disposizione delle classe che gestiscono questo tipo di dati e otterremo un NSDictionary con tutti i dati.

```
// parse della stringa ottenuta
NSDictionary *json = [stringaJson JSONValue];
```

4.3.3 Struttura interna del questionario

Per costruire il questionario, ci dobbiamo appoggiare ai file JSON ritornati dal web service.

Il file JSON contine la traccia della sttruttura del questionario e i dati(le domande e le risposte) che esso deve contenere.

Un esempio di uno dei valori contenuti nell'elenco presente nel file JSON è il seguente:

{ "QUESTIONARIO_ID": , "QUESITO_ID": , "TIPO_ELEM_COD": ,

"TIPO_FORMATO_COD": , "DES": ,

"ORD_VIS": , "PARENT_QUESITO_ID": }

Per costruire il questionario che andrà visualizzato dobbiamo analizzare i valori associati alle chiavi riportare sopra.

- **"QUESTIONARIO_ID": contiene un numero intero che identifica univocamente il questionario.**
- **"QUESITO_ID": contiene anche esso un numero intero che però identifica univocamente il quesito.**

- "TIPO_ELEM_COD": contiene una stringa in base alla quale possiamo determinare il tipo di elemento del quesito; i valori possibili sono i seguenti: Q_pagina, Q_paragrafo, Q_domanda oppure Q_risposta.
- "TIPO_FORMATO_COD":contiene il tipo di formato del quesito. Il formato del quesito dipende anche del tipo di elemento. Se il quesito è una domanda allora il formato può essere:

• TL_DOM_DFS (domanda con risposte a dominio finito e scelta singola). Questo formato richiede, nella compilazione, la comparsa di un Radiobutton a fianco di ogni risposta "figlia" , in modo da permettere la scelta di una e una sola risposta alla domanda.

• TL_DOM_DFM (domanda con risposte a dominio finito e scelta multipla). Questo formato richiede la comparsa di una Checkbox a fianco di ogni risposta "figlia" , in modo da permettere la scelta di una o più risposte alla domanda.

• TL_DOM_OFS (domanda con risposte a dominio finito e scelta singola con layout orizzontale) .

• TL_DOM_OFM (domanda con risposte a dominio finito e scelta multipla con layout orizzontale). Questi ultimi due formati sono simili ai rispettivi precedenti, con la differenza che dispongono le domande in riga e le risposte possibili in colonna, in modo da creare visivamente una tabella a matrice. Ovviamente questo layout ha ragione di essere solo se si tratta di una lista con risposte su domande adiacenti (l'esempio tipico sono risposte fisse tipo si/no oppure basso/medio/alto o logicamente simili).

• TL_DOM_LIB (domanda con risposte libere). Questo formato si usa con risposte singole a testo libero.

Se il quesito è una risposta allora il formato può essere:

• TL_RSP_TFB (risposta a testo fisso). Questo tipo di risposta è il più semplice: un testo fisso da leggere e eventualmente da selezionare. Evidentemente non deve essere "figlia" di una domanda TL_DOM_LIB.

- **TL_RSP_ALF (risposta libera di tipo alfanumerico).**

- **TL_RSP_DTA (risposta libera di tipo data).**

- **TL_RSP_NUM (Risposta libera di tipo numerico). Negli ultimi due formati, il sistema effettuerà un controllo sintattico dei testo immesso, in modo da circoscrivere il tipo di dato immesso a quelli specificati dal formato (una data e un numero rispettivamente).**

➤ **"DES": contiene una stringa che rappresenta la descrizione del quesito;**

➤ **"ORD_VIS": contiene un numero intero che rappresenta l'ordine di visualizzazione dei quesiti;**

➤ **"PARENT_QUESITO_ID": contiene un numero intero e rappresenta l'id del quesito con il quale l'attuale quesito ha una relazione(di solito di apparteneza al paragrafo/pagina).**

Analizzando questi dati potremo costruire graficamente il questionario scelto.

4.4 Progettazione

Avendo fissato l'obiettivo che deve raggiungere l'applicazione e le modalità di sviluppo, la fase successiva consiste nella progettazione vera e propria dell'app.

Lo sviluppo del progetto è iniziato con uno schema generale delle funzionalità che deve raggiungere e le modalità per realizzarle.

Ricordiamo che i requisiti principali del progetto sono i seguenti:

- l'applicazione deve essere installata su un device che ha una connessione internet;

- utilizzando la connessione internet deve accedere ai dati presenti nel web service, tramite una richiesta HTTP;

- dopo aver raggiunto il web service deve interpretare la risposta ricevuta da esso;

- gestendo opportunamente la risposta deve gestire la visualizzazione sul dispositivo dei questionari.

4.4.1 Creazione del progetto

In Xcode è stato creato un nuovo progetto di tipo Single View Application, con Product name : Questionari Uniweb, settando come Device family : iPhone e Deployment target : 4.0(l'app può essere installata e utilizzata sui device che hanno come sistema operativo iOS 4 oppure le versioni successive ad essa).

Dopo aver creato il progetto con le caratteristiche sopra elencate si sono dovuti aggiungere altri framework accanto a quelli di default. I framework di default di un progetto standard con queste caratteristiche sono : UIKIt.framework, Foundation.framework, CoreGraphics.framework.

I framework aggiuntivi che sono stati utilizzati sono i seguenti:

- CFNetwork.framework: fa parte della famiglia del framework CoreService e fornisce una libreria per i protocolli di rete, semplificando in questo modo una serie di attività, come lavorare con Http;

- SystemConfiguration.framework: fornisce funzioni che determinano la raggiungibilità di host di destinazione,sia in modo sincrono che asincrono, e servizi di rilevamento degli errori;

- MobileCoreServices.framework: contiene i servizi di sistema fondamentali.

Creando un tale progetto, come classi di default avremo : QuestionariUniwebAppDelegate.h, che contiene l'interfaccia della classe stessa e QuestionariUniwebAppDelegate.m, la classe dove vengono implementati i metodi definiti nell'interfaccia e anche i file QuestionariUniwebViewController.h, QuestionariUniwebViewController.m e QuestionariUniwebViewController.xib. Le classi di tipo ViewController sono responsabili della gestione delle activity dell'applicazione e di tutti i comandi che verranno forniti dall'utente.

4.4.2 Gestione della connessione internet

Per raggiungere le funzionalità stabilite, il progetto ha bisogno di una connessione internet. Per gestire questo aspetto abbiamo aggiunto nel progetto delle classi, fornite da Apple, che si occupano della gestione della connessione: Reachability.h/.m e CheckConnection.h/m.

La classe Reachability mette a disposizione dei metodi per verificare se esiste una connessione internet e di che tipo.

La classe ConnectionCheck, verifica utilizzando i metodi della classe Reachability se è disponibile una connessione internet e lo stato di essa.

4.4.3 La gestione dei dati

Dopo aver eseguito una richiesta e ottenuto una risposta, i dati sono mantenuti in un oggetto di tipo NSDictionary, contente elementi con chiave e valore.

La gestione dei dati ha rappresentato una sfida dal punto di vista concettuale, in quanto essa si poteva realizzare in modi diversi, tutti corretti, però si doveva scegliere la soluzione più efficiente pensando sempre al modo in cui i dati verrano utilizzati nell'applicazione.

Dopo varie analisi e tentativi come soluzione per la gestione dei dati ne erano rimaste solo due: la gestione dei dati tramite un database e quella tramite indici.

4.4.3.1 Gestione dei dati tramite un database SQL

Un database SQL rappresenta un insieme di dati collegati secondo un modello relazionale, in modo tale da consentire la gestione dei dati stessi. Per l'interfacciamento con l'utente si usano delle particolari applicazioni software dedicate basate su un'architettura di tipo client-server e sulle query. La query rappresenta l'interogazione di un database nel compiere determinate operazioni.

Se il numero delle occorrenze da gestire è molto elevato, la gestione tramite un database si rivela ad essere la soluzione vincente. Proprio per questo i servizi offerti da una gestione tramite database sono stati adattati anche sulle piattaforme mobile.

Nella piattaforma iOS viene utilizzato SQLite. SQLite è una libreria software che implementa una versione light del motore database SQL. I vantaggi di questo software sono molti, tra qui il fatto che è un software open source, il codice è maturo e propone un ambiente controllato e portatile.

iOS non fornisce classi per rendere l'utilizzo di SQLite più semplice. Lo sviluppatore,pe utilizzare le API di SQLite deve aggiungere la libreria SQLite al suo progetto e rincorrere al linguaggio C, mentre il resto dell'applicazione deve essere sviluppato in Objective-C.

Per poter utilizzare SQLite nel progetto bisogna aggiungere un framework, formito da Apple, libsqlite3.dylib.

Dopo aver aggiunto il framework si procede con la "costruzione" del database all'interno dell'app. Per gestire il database viene creata una nuova classe che gestisce : la creazione del file .db, la creazione delle varie tabelle, l'inserimento dei valori dento le tabelle, le varie query eseguite. L'interfaccia di questa classe è la seguente:

```
#import <UIKit/UIKit.h>
#import "/usr/include/sqlite3.h"
#import "MakeDB.h"
@interface QuestionarioSceltoViewController : UIViewController
{ NSString *databasePath;
sqlite3 *questionarioDB; }
-(void)findData;
@end
```

La creazione del database avviene eseguendo le successive istruzioni:

```
NSString *docsDir;
NSArray *dirPaths;
dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
docsDir = [dirPaths objectAtIndex:0];
// Build the path to the database file
databasePath = [[NSString alloc] initWithString: [docsDir string-
ByAppendingPathComponent: @"questioario.db"]];
NSFileManager *filemgr = [NSFileManager defaultManager];
// Do any additional setup after loading the view from its nib.
if ([filemgr fileExistsAtPath: databasePath ] == NO)
{ const char *dbpath = [databasePath UTF8String];
//CREAZIONE DELLE DUE TABELLE DEL MIO DATABASE
DEL QUESTIONARIO
if (sqlite3_open(dbpath, &questionarioDB) == SQLITE_OK)
```

```
{ char *errMsg; const char *sql_stmt1 = "CREATE TABLE IF
NOT EXISTS DOMANDE (QUESITO_ID INTEGER PRIMARY
KEY, TIPO_FORMATO_COD TEXT, ORD_VIS INTEGER, PA-
RENT_QUESITO_ID INTEGER,DES TEXT)";

if (sqlite3_exec(questionarioDB, sql_stmt1, NULL, NULL, &errM-
sg) != SQLITE_OK)

{ NSLog(@"Failed to create table"); }
```

Il popolamento delle tabelle:

```
sqlite3_stmt *statement;

const char *dbpath = [databasePath UTF8String];

if (sqlite3_open(dbpath, &questionarioDB) == SQLITE_OK)

{ for(int i = 0; i< [[[QuestionarioData instance]arrayIDRisposte]
count]; i++)

{ NSString *insertSQL = [NSString stringWithFormat: @"INSERT
INTO RISPOSTE (QUESITO_ID,TIPO_FORMATO_COD,ORD_VIS,PARENT_QUESI
VALUES (\"%@\", \"%@\",\"%@\",\"%@\",\"%@\")", [[[Que-
stionarioData instance]arrayIDRisposte] objectAtIndex:i],

[[[QuestionarioData instance]arrayFormRisposte]objectAtIndex:i] ,

[[[QuestionarioData instance]arrayOrdineRisposte]objectAtIndex:i],

[[[QuestionarioData instance]arrayParentRisposte]objectAtIndex:i],

[[[QuestionarioData instance]arrayRisposte]objectAtIndex:i] ];

const char *insert_stmt = [insertSQL UTF8String];

sqlite3_prepare_v2(questionarioDB, insert_stmt, -1, &statement,
NULL); }

if (sqlite3_step(statement) == SQLITE_DONE)

{ NSLog(@"Dati aggiunti con successo"); isData = TRUE; }

else

{ NSLog(@"Errore"); isData = FALSE; } sqlite3_finalize(statement);

sqlite3_close(questionarioDB); }
```

Per eseguire un query sul database utilizziamo :

```
const char *dbpath = [databasePath UTF8String];

sqlite3_stmt *statement;

if (sqlite3_open(dbpath, &questionarioDB) == SQLITE_OK)

{ for(int i = 0; i < [[[QuestionarioData instance]arrayIDDomande]
count];i++)

{ NSString *querySQL = [NSString stringWithFormat: @"SELECT
quesito_id FROM domande "];

const char *query_stmt = [querySQL UTF8String];

if (sqlite3_prepare_v2(questionarioDB, query_stmt, -1, &statement,
NULL) == SQLITE_OK)

{ if (sqlite3_step(statement) == SQLITE_ROW)

{ NSLog(@"l'id domande db=%@",[[NSString alloc] initWithUTF8String:(const
char *) sqlite3_column_text(statement,0)]); } sqlite3_finalize(statement);
}

sqlite3_close(questionarioDB); } }
```

In questo esempio la query :"SELECT quesito_id FROM domande " stampa i quesito_id contenuto nella tabella domande.

Siccome le tabelle vengono popolati con i dati ritornati dal web service dopo le richiste, essi devo essere sempre aggiornati. Il metodo più semplice ed efficiente è quello di cancellare i dati e riscriverli ogni volta. Per eseguiere la cancellazione si utilizza l'istruzione:

```
[filemgr removeItemAtPath:databasePath error:NULL];

[filemgr release];
```

4.4.3.2 Gestione dei dati tramite indici

La gestione dei dati tramite indici si è rivelata la soluzione più semplice e stabile, in quanto utilizziamo oggetti apparteneti alle classi native della piattaforma, come NSArray oppure NSMutableArray.

Per memorizzare i dati creiamo una classe di tipo NSObject, chiamata QuestionariData. In questa classe dichiariamo i vari NSMutableArray dentro i quali verrano memorizzati i dati. Per poter essere utilizzati anche all'esterno della classe, questi oggetti saranno una "@property" della classe. Inoltre sviluppiamo anche un metodo che ritornerà un'istanza di questa classe.

Gli oggetti di questa classe saranno accessibili da tutte le altre classi del progetto sia in lettura, per visualizzare il questionario; sia in scrittura, per riscrivere i dati dopo una nuova richiesta al web service. Inoltre i dati contenuti in questi oggetti non vengono cancellati se l'applicazione viene messa in background.

Dopo aver ricevuto una risposta, effettuato il relativo parsing e incapsulato i dati dentro il dizionario, tali dati saranno selezionati e salvati negli array costruiti prima.

```
// Parse della stringa
NSDictionary *json = [stringaJson JSONValue];
//salvo i dati i un file unico
[[QuestionarioData instance] setCONT_COD:[json valueForKey:@"CONT_COD"]];
if([[QuestionarioData instance] CONT_COD]!= nil)
{ [[QuestionarioData instance] setIsQuestionario:TRUE];
[[QuestionarioData instance] setDES:[json valueForKey:@"DES"]];
[[QuestionarioData instance] setNOTE:[json valueForKey:@"NOTE"]];
[[QuestionarioData instance] setQUESTIONARIO_COD:[json valueForKey:@"QUESTIONARIO_COD"]];
[[QuestionarioData instance] setQUESTIONARIO_ID:[json valueForKey:@"QUESTIONARIO_ID"]]; }
```

Come soluzione definitiva della gestione dei dati è stato scelto l'utilizzo degli array, in quanto sono degli oggetti facile da gestire. Per ottenere i dati di un array basta fare una scansione, invece con l'utilizzo del database bisognerebbe aprire e chiudere la connessione con il database, e la query eseguita non restituisce tutti i valori presenti in una colonna della tabella, ma ben si il valore prensente in una sola riga.

Siccome dobbiamo costruire un questionario composto da domande e risposte che hanno varie proprietà dobbiamo accedere molte volte ai dati e per questo la soluzione con gli indici è risultata quella più efficiente.

4.4.4 QuickDialog

QuickDialog è una libreria open source, scaricabile dal sito github.com, che offre una serie completa di funzioni per la creazione di finestre di dialogo in stile iPhone. Offre molti elementi graffici, tra cui i pulsanti on/off selettori, date, cursori, i controlli in stile radio, la scrittura del testo, tra gli altri. Inserendo questa libreria nel progetto si possono creare form per la nostra app senza dover trattare direttamente con UITableViews, delegati e fonti di dati, evitando di spendere molto tempo per la creazione e gestione di tabelle, righe e relativi controlli.

Per sfruttare al meglio questa libreria, si devono utilizzare tre diversi tipi di classi:

- QuickDialogController - sottoclasse di UITableViewController che è responsabile della visualizzazione della finestra di dialogo. Nella applicazione veranno create delle sottoclassi di questa classe, una per ogni finestra di dialogo che si gestisce;non si andrà mai a creare oggetti di questo tipo direttamente con alloc / init.

- QRootElement - è stato pensato come l'elemento principale della finestra di dialogo: un insieme di sezioni e celle che possono essere utilizzati per visualizzare dei dati utili per l'utente. Ogni QuickDialogController può visualizzare solo un RootElement alla volta, ma quest'ultimo può contenere altri elementi dello stesso tipo. Gli elementi sono sempre raggruppate in sezioni nell'elemento radice.

- QElement - un elemento di un UITableViewCell.

QuickDialog mette a disposizione anche altri tipi di elementi. Uno degli altri tipi elementari è QSection, che rappresenta una sezione dalla tabella. Come la QRootElement, anche essi hanno varie proprietà: title/footer e headerView/footerView.

Un altro vantaggio di questa libreria è che si occupa della gestione delle tastiere e dei datePicker. Se nel nostro dialogo c'e un QEntryType e poi un QDateTimeElement non ci si deve preoccupare della gestione del cambio di tastiere, tutto avviene in automatico.

Altri tipi di elementi che hanno un comportamento analogo al quello di QSection sono :

- QRadioSection - visualizza direttamente una lista di elementi, senza dover caricare un'altra view;

- QSortingSection.

Tra i più usati elementi messi a disposizione da QuickDialog possiamo nominare anche: QLabelElement, QBooleanElement, QButtonElement, QDateTimeElement, QEntryElement, QTextElement, QWebElement.

Tutti questi elementi possono utilizzare le seguenti proprietà:

- key: utilizzata per settare il nome di ogni cella, in modo da tener traccia dell'elemento e poi leggere il valore inserito in esso;

- controllerAction: una stringa contenente il nome del metodo che il controller andrà a chiamare quando l'elemento sarà selezionato;

- onSelected: un metodo che verrà chiamato quando l'elemento sarà selezionato.

4.4.4.1 Esempio di utilizzo di QuickDialog

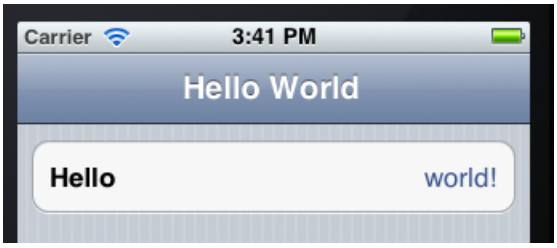
Se nel nostro progetto dopo aver implementato la libreria scriviamo il seguente codice:

```
QRootElement *root = [[QRootElement alloc] init];
root.title = @"Hello World";
root.grouped = YES;
QSection *section = [[QSection alloc] init];
QLabelElement *label = [[QLabelElement alloc] initWithTitle:@"Hello" Value:@"world!"];

[root addSection:section];
[section addElement:label];

UINavigationController *navigation = [QuickDialogController
controllerWithNavigationForRoot:root];
[self presentModalViewController:navigation animated:YES];
```

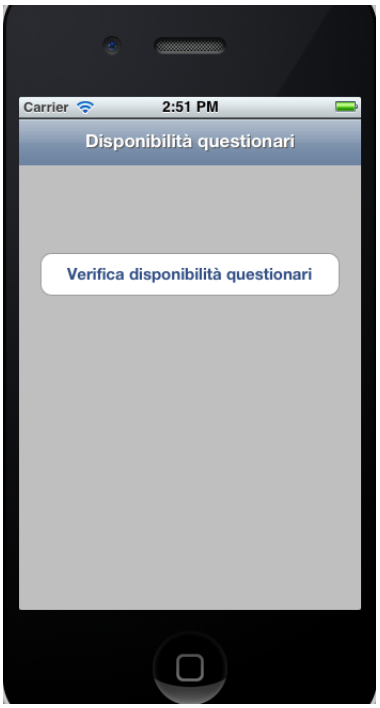
Otteremo il seguente form :



Schermata di un app che implementa la libreria QuickDialog e prodotta dal codice riportato come esempio

4.5 Sviluppo dell'applicazione

4.5.1 QuestionariUniwebViewController



La prima schermata dell'applicazione

Per gestire le activity utilizzeremo classi di tipo ViewController che estendono UIViewController, classe nativa che fornisce le fondamenta per la gestione delle view. Quando si definisce una nuova sottoclasse di tipo UIViewController si devono specificare le view che il controller andrà a gestire. Una classe di tipo UIViewController fornisce supporto per il caricamento automatico della view, quando si accede alla proprietà di visualizzazione.

La prima activity dell'app utilizza una istanza di QuestionariUniwebViewController. Per costruire la view dal punto di vista grafico andremo a modificare il file QuestionariUniwebViewController.xib. Dove andremo a trascinare gli oggetti che vogliamo utilizzare, nel nostro caso un bottone e poi collegarlo all'interfaccia dove avevamo dichiarato l'oggetto e l'azione collegata ad esso.

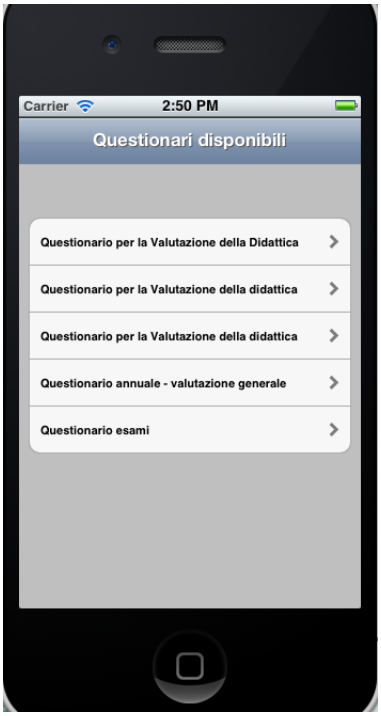
```
@interface QuestionariUniwebViewController :
UIViewController<UIAlertViewDelegate,UITableViewDataSource,
UITableViewDelegate>
{ IBOutlet UIButton *cercaQuestionariBottone; }
-(IBAction)cercaQuestionariFunzione:(id)sender;
@end
```

Il metodo collegato al bottone viene invocato quando l'utente compie il gesto di "Touch up inside", cioè un semplice tap all'interno del rettangolo grafico che rappresenta il bottone.

Nel metodo "cercaQuestionariFunzione", implementato nella classe QuestionariDisponibiliViewController.m viene dichiarato un oggetto del tipo della classe dove risiede la funzione che compie la richiesta verso il web service e poi tramite questo viene chiamato il metodo della richiesta. Se l'esito della chiamata è positivo si procede con il caricamento della prossima view, che elencherà i questionari disponibili.

```
UIViewController *chiamta = [[UIViewController alloc]init];
CercaQuestionari *cercaQuestionari=
[[CercaQuestionari alloc]initWhitView:chiamta ];
[cercaQuestionari cercaQuestionariDisponibili] ;
if([[QuestionarioData instance] isQuestionario])
{ [alertview dismissWithClickedButtonIndex:0 animated:YES];
QuestionariDisponibiliViewController *tabellaQuestionari = [[Que-
stionariDisponibiliViewController alloc] initWithNibName:nil bund-
le:nil];
[self presentModalViewController:tabellaQuestionari animated:NO];
[tabellaQuestionari release]; }
else { [alertview dismissWithClickedButtonIndex:0 animated:YES];
UIAlertView *alert=[[UIAlertView alloc] initWithTitle:titoloErrore
message:messagioErrore delegate:self
cancelButtonTitle:@"OK" otherButtonTitles:nil];
[alert show]; [alert release]; }
```

4.5.2 QuestionariDisponibiliViewController



La secondo schermata dell'app fa uso di un'istanza di QuestionariDisponibiliViewController. Anche tale classe eredita da UIViewController.

Azionando il bottone presente nella prima schermata viene generata una richiesta al web service. La risposta, se la chiamata non è andata in errore, è un file JSON che contiene informazioni sui questionari che sono disponibili per la compilazione e in quel momento viene caricata la seconda schermata. La seconda schermata contiene una UITableView, le cui righe contengono i titoli dei questionari disponibili.

Un oggetto di tipo UITableView rappresenta un oggetto messo a disposizione da Apple per la visualizzazione e la modifica di elenchi gerarchici di informazioni.

Per poter utilizzare questo oggetto, la nostra classe deve implementare i protocolli previsti per il suo funzionamento.

@interface QuestionariDisponibiliViewController :

UIViewController<UITableViewDataSource,UITableViewDelegate,

UIAlertViewDelegate>

Per costruire la tabella e popolarla con le informazioni necessarie dobbiamo utilizzare i metodi standard messi a disposizione. Per il numero di sezioni e il numero di righe utilizziamo i metodi seguenti:

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView { return 1; }

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section { return numeroRighe; }

Per popolare le righe della tabella e per aggiungere ad ogni riga l'accessorio utilizziamo il seguente frammento di codice:

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath

{ static NSString *CellIdentifier = @"Cell";

UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

if (cell == nil)

{ cell = [[[UITableViewCell alloc] initWithFrame:CGRectZero reuseIdentifier:CellIdentifier] autorelease];

cell.textLabel.textColor = [UIColor blackColor];

cell.textLabel.font = [UIFont fontWithName:@"Helvetica Neue" size:11.0];

cell.textLabel.font = [UIFont boldSystemFontOfSize:11.0]; }

NSString *cellValue = [titoliTabella objectAtIndex:indexPath.row];

cell.textLabel.text = cellValue; return cell; }

- (UITableViewCellAccessoryType)tableView:(UITableView *)tableView accessoryTypeForRowWithIndexPath:(NSIndexPath *)indexPath { return UITableViewCellAccessoryDisclosureIndicator; }

Per selezionare una riga della tabella si utilizza il metodo:

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath

Quando una riga viene selezionata, il metodo appena nominato viene chiamato e parte l'azione che deve compiere la riga di indice indexPathRow. Quando una delle righe della tabella viene selezionata parte una richiesta verso il web service per interrogarlo sui dati che il questionario scelto contiene. Se la chiamata va a buon fine, salveremo i dati in degli NSArray. Questi dati verranno utilizzati per la creazione del questionario che conterrà domande e risposte.

4.5.3 La costruzione del questionario

Dopo aver selezionato uno dei questionari disponibili, proposti nella schermata QuestionariDisponibiliViewController, partono due richieste verso il web service; la prima ritorna i dati del questionario (domande, risposte) e la seconda ritorna le condizioni, cioè se ci sono domande che dipendono da altre risposte. Se entrambe le chiamate vanno a buon fine si procede con la "costruzione" del questionario.

Per la costruzione del questionario utilizzeremo la libreria open source QuickDialog, presentata in precedenza.

I dati ritornati dal web service vengono memorizzati in NSMutableArray, in modo da avere l'array degli id dei quesiti, l'array delle descrizioni dei quesiti presenti. Questi array hanno lo stesso numero di elementi e questa proprietà rende molto facile l'operazione di ricerca di un certo quesito.

Dopo aver eseguito la prima richiesta, organizzeremo i dati dentro degli array. L'arrayId conterà un numero che identifica univocamente il quesito. L'arrayTipo_elem_cod conterà una stringa in base alla quale possiamo determinare il tipo di elemento che il quesito può essere; i valori possibili che questo array può contenere sono i seguenti: Q_pagina, Q_paragrafo, Q_domanda oppure Q_risposta. L'arrayTipo_form_cod conterà il tipo di formato del quesito.

Questi sono i principali array che utilizzeremo per la costruzione del questionario.

Come prima cosa andremo a verificare il tipo di elemento del quesito. Se esso è di tipo Q_pagina, non verrà visualizzato nel nostro questionario, in quanto questo è un parametro che viene utilizzato nella costruzione del questionario web; ci limiteremo soltanto a memorizzare i quesiti che fanno parte di questa pagina. Invece se il tipo di elemento è Q_paragrafo, se esso ha una descrizione, essa sarà visualizzata come header title di una sezione.

```
if([[[[QuestionarioData instance]arrayTipo_elem_cod] objectAtIndex:i] isEqualToString:@"Q_PARAGR"])
{ QSection *section = [[QSection alloc]initWithTitle:[[[QuestionarioData instance]arrayQuesiti]objectAtIndex:i]];
[root addSection:section]; }
```

Se il tipo di elemento è Q_domanda, con un switch andremo a verificare il tipo di formato del quesito ed in base ad esso andremo a costruire una nuova sezione dove in un QTextElement ci sarà scritta la descrizione del quesito.

```
QSection *section = [[QSection alloc]init];
QTextElement *domandaLibera= [[QTextElement alloc] initWithText:stringaText ];
domandaLibera.font = [UIFont boldSystemFontOfSize:12];
domandaLibera.color =[UIColor blackColor];
[section addElement:domandaLibera];
[root addSection:section];
```

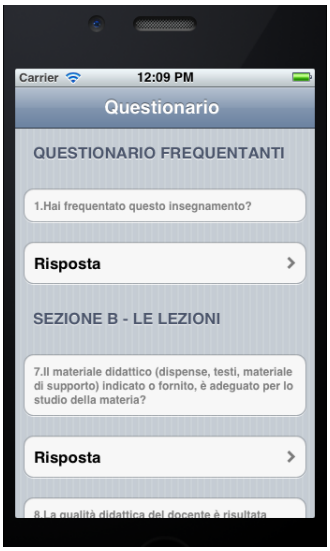
Per i quesiti che sono di tipo Q_risposta si esegue un ragionamento analogo, utilizzando un altro switch. Per le risposte di tipo TL_RSP_TFB andremo a costruire un QRadioElement se la domanda era di tipo TL_DOM_DFS oppure un QSegmentedElement se era di tipo TL_DOM_OFS.

Ovviamente ci sono anche altri controlli, ad esempio per vedere se i quesiti apartengono ad una pagina che dipende da una certa risposta: in questo caso tutte le domande e risposte appartenenti alla pagina andranno nascoste.

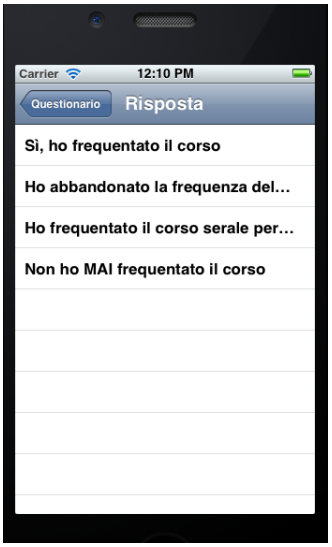
Dopo la visualizzazione del questionario, l'utente potrà procedere con la compilazione del questionario. Se l'utente selezionerà una risposta che sblocca l'apparizione di una Q_pagina il questionario verra ricaricato, in modo da visualizzare anche le domande che prima non erano visualizzate.

In fondo al questionario l'utente troverà due bottoni. Azionando il primo tornerà alla pagina dei questionari disponibili, ma le risposte date verrano salvate. Esse verrano cancellate soltanto nel caso in cui l'utente chiuderà l'applicazione. Invece azionando il secondo bottone tutte le sue risposte verrano salvate in un NSDictionary e mandate al web service di Kion per essere analizzate.

4.5.4 Schermate questionario



Analizzando la schermata notiamo che "Questionari frequentati" è il titolo di una QSezione e rappresenta la descrizione di un quesito di tipo Q_paragr. La prima domanda posizionata in un QTextElement è la descrizione di un quesito di tipo Q_domanda. Cliccando sulla cella contente la parola "Risposta" si aprerà la seconda schermata con dove comparirano le varie risposte.



Questa schermata contiene le risposte appartenenti alla prima domand ed è stata costruita utilizzando un QRadioElement. Non appena viene scelta una risposta l'app carica la pagina precedente.

In questa schermata vengono mostrati i bottoni che compaiano alla fine di ogni questionario. Cliccando il primo,"Torna ai questionari" l'app riporta in primo piano la pagina con i questionari disponibili. Il secondo bottone invia la riaposte al web server.

In questa schermata viene presentato un QSegmentedElement, per poter mostrare graficamente un Q_risposta con il formato TL_RSP_OFS.

Bibliografia

[1] Sito ufficiale delle librerie messe a disposizione da Apple: http://developer.apple.com/library/ios

[2] www.wikipedia.org

[3] "Programming in Objective-C 2.0 : a complete introduction to the Objective-C language for Mac OS X and IPhone development" Stephen G. Kochan

[4] Sito del consorzio: www.cineca.it

[5] Sito web di JSON: www.json.org

[6] Sito web di SQLite: www.sqlite.org

[7] Sito-repository di vari progetti : www.github.com

[8] Sito web della libreria QuickDialog: http://escoz.com/open-source/quickdialog