

UNIVERSITÀ DEGLI STUDI DI UDINE

---

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Triennale in Informatica

Tesi di Laurea

APPLICAZIONE CONTEXT-AWARE  
PER COMPENSORI SCIISTICI

Relatore:

Dott. IVAN SCAGNETTO

Laureando:

MASSIMO PALADIN

---

ANNO ACCADEMICO 2006-2007



# Ringraziamenti

Ringrazio i miei genitori per l'educazione datami, per avermi dato la possibilità di studiare e soprattutto per il sostegno economico, senza il quale non avrei potuto fare niente.

Grazie a Marianna per avermi sopportato durante le sessioni d'esame, con il mio poco tempo e un po' di nervoso, e spero anche che mi possa sopportare in quelle future.

Ringrazio il mio relatore, Ivan Scagnetto per aver accolto la mia idea di progetto e per avermi dato consiglio in ogni problema riscontrato in fase di progettazione e di scrittura della tesi.

Ringrazio anche tutti quelli che mi sono stati vicini in questi anni e che hanno contribuito a questo risultato.

A tutti, un sincero *Grazie*



# Indice

<b>Ringraziamenti</b>	<b>iii</b>
<b>Elenco delle figure</b>	<b>vii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Introduzione . . . . .	1
1.2 Obiettivo della tesi . . . . .	2
1.3 Sinossi dei capitoli . . . . .	4
<b>2 Tecnologie e strumenti utilizzati</b>	<b>5</b>
2.1 Tecnologie . . . . .	6
2.1.1 Telefoni cellulari . . . . .	6
2.1.2 Bluetooth . . . . .	8
2.1.3 JSR-82 . . . . .	10
2.1.4 J2ME - Java 2 Micro Edition . . . . .	10
2.1.5 Java 2 Standard Edition . . . . .	12
2.1.6 PHP - Hypertext Preprocessor . . . . .	12
2.2 Strumenti Utilizzati . . . . .	12
2.2.1 Librerie Avetana . . . . .	12
2.2.2 Librerie BouncyCastle . . . . .	13
2.2.3 Librerie kXML . . . . .	13
2.2.4 Cellulari Nokia 6630 e 5500 . . . . .	13
2.2.5 MySql - Database Management System . . . . .	14
2.2.6 Netbeans 5.5 . . . . .	14
<b>3 Descrizione del progetto</b>	<b>15</b>
3.1 Analisi del progetto . . . . .	15
3.2 Protocolli di comunicazione . . . . .	16
3.2.1 Interconnessione tra client e server . . . . .	16
3.2.2 Tipi di richieste . . . . .	17
3.2.3 Formato delle pagine . . . . .	17

---

<b>4</b>	<b>Client</b>	<b>19</b>
4.1	Requisiti . . . . .	19
4.2	Classi Realizzate . . . . .	19
4.3	Scelte Implementative . . . . .	25
4.3.1	Design . . . . .	25
4.3.2	Multi-Thread . . . . .	25
4.3.3	Sicurezza . . . . .	25
4.4	Funzionamento . . . . .	26
<b>5</b>	<b>Server</b>	<b>31</b>
5.1	Server . . . . .	31
5.1.1	Requisiti . . . . .	31
5.1.2	Classi Realizzate . . . . .	31
5.1.3	Scelte Implementative . . . . .	35
5.1.4	Funzionamento . . . . .	35
5.2	Database dei contenuti . . . . .	36
5.2.1	Requisiti . . . . .	36
5.2.2	Analisi e progettazione . . . . .	37
5.2.3	Implementazione . . . . .	37
5.3	Interfaccia inserimento contenuti . . . . .	38
5.3.1	Requisiti . . . . .	38
5.3.2	Implementazione . . . . .	38
<b>6</b>	<b>Conclusione e futuri sviluppi</b>	<b>43</b>
6.1	Conclusioni . . . . .	43
6.2	Sviluppi futuri . . . . .	44
6.2.1	Piattaforma per la distribuzione dell'applicazione . . . . .	44
6.2.2	Arricchimento dei contenuti . . . . .	44
6.2.3	Realizzazione Servlet di controllo . . . . .	44
6.2.4	Ampliamento per Musei . . . . .	44
	<b>Bibliografia</b>	<b>47</b>

# Elenco delle figure

2.1	Scatternet di due piconet . . . . .	8
2.2	Stack dei protocolli Bluetooth . . . . .	9
2.3	Java2 Platform . . . . .	11
4.1	Diagramma delle classi principali del Client . . . . .	20
4.2	Snapshot della MIDlet - 1 . . . . .	27
4.3	Snapshot della MIDlet - 2 . . . . .	27
4.4	Snapshot della MIDlet - 3 . . . . .	28
4.5	Sequence Diagram di inoltro di una richiesta . . . . .	29
4.6	Richiesta di autorizzazione di accesso al canale Bluetooth . . . . .	29
5.1	Diagramma delle classi principali del server . . . . .	32
5.2	Snapshot di MainFrame . . . . .	33
5.3	Sequence Diagram di gestione di una richiesta . . . . .	36
5.4	Schema ER del database dei contenuti . . . . .	38
5.5	Riassunto dei permessi . . . . .	39
5.6	Testata dell'interfaccia . . . . .	40
5.7	Menù dell'interfaccia . . . . .	40
5.8	Bottom - Parte bassa dell'interfaccia . . . . .	41
5.9	Lista degli utenti . . . . .	41
5.10	Lista delle news . . . . .	41



# Capitolo 1

## Introduzione

### 1.1 Introduzione

I dispositivi mobili ad oggi sono molto importanti, negli ultimi anni hanno avuto una forte espansione commerciale e tecnologica, a partire dai palmari nell'ambito lavorativo fino ai semplici cellulari, che poi così semplici non lo sono più.

Basti pensare che neanche dieci anni fa il cellulare era una cosa per pochi, aveva dimensioni elevate, poca autonomia, poche funzionalità e soprattutto un prezzo proibitivo, veniva comprato solo da chi ne aveva veramente bisogno, ad esempio per motivi di lavoro.

Da allora le cose sono cambiate totalmente, lo sviluppo ha iniziato con il ridurre le dimensioni, con l'aumentare l'autonomia con migliore gestione dell'energia e batterie sempre più sofisticate. I servizi offerti erano pochi, tra cui chiamate, SMS e navigazione internet WAP. Poi pian piano si è passati all'introduzione degli schermi a colori, all'offerta dei messaggi multimediali (MMS), al GPRS, all'inserimento di fotocamere integrate nei dispositivi e poi infine all'UMTS offrendo la possibilità di effettuare videochiamate.

Dall'inserimento delle fotocamere si è iniziato ad inserire memorie integrate sempre più ampie per finire con l'inserire schede di memoria che permettono un'espansione della memoria fino a 2-4gb in modo da permettere di usare il cellulare come lettore MP3 e multimediale.

Nel corso degli sviluppi è aumentata anche l'interconnettività tra i dispositivi, prima con l'infrarosso (poche potenzialità) per poi arrivare al Bluetooth che permette la connessione tra cellulari e altri dispositivi come auricolari.

Con la forte espansione tecnologica i dispositivi mobili sono diventati sempre più sofisticati, la microelettronica ha permesso di racchiudere processori sempre più potenti e con maggiore autonomia in piccole dimensioni, basti pensare

che ad oggi un cellulare di fascia media offre le operazioni di base offerte da un calcolatore (editor di testi, lettore multimediale, personal organizer, Internet browsing...), ora si sta iniziando ad inserire anche ricevitori GPS per trasformare i cellulari in navigatori satellitari.

Ultimamente sta prendendo piede l'uso di veri e propri sistemi operativi per cellulari, primo fra tutti Symbian OS, seguito da Windows Mobile, e adesso si sta iniziando ad integrare Linux per dispositivi embedded, un esempio é il recente annuncio del team "Ubuntu Linux" riguardante lo sviluppo di una distribuzione per cellulari. Altro concorrente che debutterá nel corso del 2007 é Apple che lancerá il suo primo smartphone equipaggiato con un sistema operativo proprietario, per il quale rilascerà delle API che permetteranno a terzi di sviluppare software.

Fattore determinante della forte diffusione è la discesa dei costi di produzione, tanto che con pochi soldi si può comprare un cellulare con ottime funzionalità. Data la grande espansione l'offerta di servizi è aumentata notevolmente, soprattutto a pagamento però, comunque i nuovi servizi sono sempre ben accetti, a maggior ragione se gratuiti e di pubblico interesse, così è nata l'idea di offrire un servizio di informazione agli sciatori, un servizio che i gestori di comprensori sciistici potrebbero offrire alla loro clientela.

## 1.2 Obiettivo della tesi

L'obiettivo della tesi è quello di realizzare una piattaforma *context-aware* per l'offerta di servizi e informazioni ai dispositivi mobili.

Per *context-aware* si intende la fruizione di un servizio che varia a seconda dell'ambiente in cui ci si trova, questo genere di servizi solitamente è riservato a dispositivi come cellulari e PDA. In particolare la classe prescelta che non a caso è anche quella più diffusa è quella dei cellulari, comunque, come vedremo più avanti non escluderemo dalla compatibilità i PDA.

I contenuti che desideriamo offrire sono formati da testo e immagini, vogliamo creare un ambiente in cui l'utente possa navigare tra informazioni utili, quindi i contenuti devono essere mirati e non troppo lunghi, dato che lo schermo di un cellulare non favorisce di sicuro la lettura di lunghi documenti.

Per lo sviluppo abbiamo deciso di specializzare la piattaforma a contenere informazioni riguardanti i comprensori sciistici, quindi l'utenza coinvolta saranno sciatori e non, in visita ai comprensori, quindi le informazioni che andremo a dare agli utenti saranno le news, il meteo, le informazioni riguardanti il comprensorio, le piste da sci, i rifugi e altro, quindi informazioni che possano essere di aiuto al cliente in visita e in vacanza.

L'applicazione sarà adattabile ad altri contesti con piccole modifiche, che sa-

ranno la semplice costruzione dei contenuti relativi al nuovo contesto scelto e alla relativa amministrazione.

Si è deciso di specializzare l'applicazione in un contesto in modo da renderne visibile e reale il funzionamento alle persone, magari ad eventuali interessati, in modo da permettere la valutazione del bisogno e dell'efficacia del sistema per eventuali impieghi reali.

### **Oggetto della specializzazione**

Si è deciso di creare il contesto per i comprensori sciistici perché è un ambiente in ampia crescita, nei comprensori sciistici troviamo gli sciatori e non solo, troviamo anche persone in vacanza che aspettano il passare delle giornate guardando il panorama delle montagne, prendendo il sole, facendo passeggiate e così via, entrambe le categorie sono in aumento negli ultimi anni, e sono entrambe bersaglio per i nostri servizi.

Il sistema che andremo a descrivere potrà gestire informazioni relative ad uno o più comprensori sciistici, questo perché le compagnie di gestione degli impianti sciistici di solito gestiscono più di un comprensorio, tanto che si formano quasi dei monopoli.

## 1.3 Sinossi dei capitoli

Dopo aver introdotto l'obiettivo della tesi inizieremo a descrivere passo passo il progetto, ecco allora la lista dei capitoli con la descrizione del loro contenuto:

**Capitolo 1:** in questo capitolo abbiamo fatto una breve introduzione all'obiettivo della tesi e al suo ambito;

**Capitolo 2:** in questo capitolo descriveremo tutte le tecnologie e gli strumenti usati per lo sviluppo del progetto, argomenteremo anche confronti con tecnologie alternative che non sono state scelte;

**Capitolo 3:** in questo capitolo faremo una prima analisi del progetto, per poi proseguire nei capitoli successivi con una descrizione più accurata;

**Capitolo 4:** in questo capitolo presenteremo la parte client in ogni suo aspetto, eviteremo di andare troppo nei dettagli per consentire la consultazione della tesi anche a persone non specializzate nell'argomento;

**Capitolo 5:** in questo capitolo descriveremo la parte server del progetto, partendo dai requisiti fino alla descrizione del funzionamento;

**Capitolo 6:** in questo capitolo andremo a scrivere le conclusioni riguardanti il progetto e sviluppi futuri consigliati per l'ampliamento del progetto.

# Capitolo 2

## Tecnologie e strumenti utilizzati

### Client

Per la parte Client si è ricorsi alle seguenti tecnologie:

- Telefoni cellulari
- Bluetooth
- JSR-82 Bluetooth API
- Java 2 Micro Edition

e ai seguenti strumenti:

- librerie BouncyCastle per la cifratura RSA
- librerie kXML per il parsing di documenti XML
- cellulari Nokia 6630 e Nokia 5500 per il testing

### Server

Per la parte server invece si sono utilizzate le seguenti tecnologie:

- Bluetooth
- JSR-82 Bluetooth API emulate su J2SE dalle librerie Avetana
- Java 2 Standard Edition

e i seguenti strumenti:

- database MySql per i contenuti

- librerie BouncyCastle per la cifratura RSA
- librerie Avetana per l'uso del Bluetooth via Java da Windows e Linux
- Netbeans 5.5 per la gestione dello sviluppo

## Interfaccia inserimento contenuti

Per l'interfaccia di inserimento dei contenuti sono utilizzate le seguenti tecnologie:

- PHP come linguaggio di scripting per le pagine dell'interfaccia

e i seguenti strumenti:

- database MySql per l'immagazzinamento dei contenuti

## 2.1 Tecnologie

### 2.1.1 Telefoni cellulari

Ora andremo a fare un breve approfondimento sulla categoria scelta per il progetto, in modo da evidenziarne gli aspetti principali che hanno condizionato alcune scelte.

#### Metodi di comunicazione

Ad oggi i cellulari possono comunicare con altri dispositivi mobili e fissi senza l'ausilio di cavi, in particolare con tre modalità:

- via Infrarosso
- via onde radio con tecnologia Bluetooth (802.15)
- via onde radio con tecnologia 802.11b e 802.11g

L'infrarosso è una tecnologia ormai sorpassata, lenta, con molte limitazioni e soprattutto non viene più incluso nei cellulari di nuova concezione.

Le tecnologie 802.11 sono utilizzate per lo più dai laptop e dai pc, hanno grandi potenzialità ma richiedono molte risorse in termini di potenza di trasmissione, di gestione e di inclusione in piccoli cellulari, infatti, ad oggi, nella vasta gamma di cellulari presenti in commercio sono presenti solo una decina di modelli che incorporano questa tecnologia e per di più compaiono in una fascia di prezzo accessibile a pochi.

Ultima tecnologia è il Bluetooth, che è la nostra scelta, è una tecnologia a basso costo, a bassa potenza di trasmissione e integrabile in dispositivi piccolissimi, come gli auricolari. Questa tecnologia non ha una banda di trasmissione molto ampia, 1Mbps lordi (molto lordi), che però sono sufficienti per il nostro progetto e concedono buone prestazioni, questo perché i dati che devono essere scambiati non sono molto pesanti. I cellulari che integrano questa tecnologia hanno una copertura di una decina di metri, quello che basta per la fruizione dei servizi o dei contenuti in questione.

### **Sistemi Operativi dei cellulari**

La maggior parte dei cellulari è equipaggiata con sistemi operativi proprietari e realizzati ad hoc dai costruttori, con due eccezioni: quelli equipaggiati con sistema operativo Windows Mobile che però non sono molto diffusi dato che è un sistema operativo più per PDA; e la seconda eccezione è sostenuta soprattutto da Nokia con l'equipaggiamento di una gran parte dei suoi prodotti con il sistema operativo Symbian, soluzione molto diffusa dato che Nokia detiene circa il 39% del mercato dei cellulari, tanto che sta iniziando ad essere presa in considerazione anche da altri costruttori.

Questa varietà di sistemi fa pensare all'incompatibilità fra essi, e così sarebbe se Sun non avesse sviluppato una versione di Java per dispositivi con poche risorse. Java 2 Micro Edition (J2ME), è il nome del linguaggio di programmazione indipendente dal sistema e dall'architettura di runtime, il codice eseguibile viene eseguito da una virtual machine, che viene realizzata per ogni tipo di sistema mentre il codice è sempre lo stesso. Pare ovvio quindi che sarà la tecnologia da noi scelta per lo sviluppo della piattaforma, questo per garantire la maggiore compatibilità possibile.

### **Compatibilità nei cellulari**

La compatibilità e il supporto di più dispositivi in un progetto così non è da poco, è un fattore determinante per il successo e la diffusione della piattaforma; impiegando le due tecnologie prescelte si garantisce una buona compatibilità, la maggiore che si possa offrire con le tecnologie attualmente in uso, questo perché J2ME ci permette di avere una compatibilità con la maggior parte dei dispositivi presenti in commercio e poi abbiamo il Bluetooth, il quale restringe il campo di compatibilità ma al quale non possiamo rinunciare perché è un requisito d'obbligo per un servizio context-aware.

## 2.1.2 Bluetooth

Il Bluetooth è una tecnologia nata dalla collaborazione di varie aziende (Ericsson, Intel, IBM, Nokia e Toshiba) per la connessione di dispositivi cellulari con altri dispositivi, i requisiti principali erano il basso costo, il basso consumo energetico e la possibilità di includerlo in piccoli spazi. Il risultato è stato un sistema dal costo inferiore ai 5 euro per implementazione, dal consumo nell'ordine di pochi milliwatt e la possibilità di includerlo in piccoli auricolari per cellulari.

### Architettura

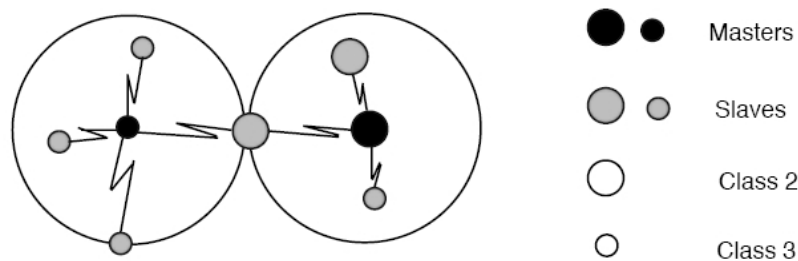


Figura 2.1: Scatternet di due piconet

Il Bluetooth ha un'architettura master slave osservabile in figura 2.1, l'unità fondamentale è la piconet, formata da un nodo master che coordina le comunicazioni all'interno di essa e poi ci possono essere fino a 7 nodi slave attivi (255 in *parked state*), che non possono comunicare direttamente tra di loro ma solo attraverso il nodo master. Due piconet possono unirsi tramite due nodi slave dando luogo ad una Scatternet, il cui routing avviene tra gli slave adiacenti.

### Pila dei protocolli

Lo *standard* Bluetooth non è ben stratificato, lo possiamo vedere dallo stack dei protocolli in figura 2.2, definisce una serie di protocolli che non seguono la struttura del modello ISO/OSI, anche se l'IEEE sta facendo il possibile per standardizzare questa tecnologia in 802.15 e renderlo simile alle altre tecnologie 802 anche se ci sono molti problemi, a partire dall'interferenza con *802.11b-g*.

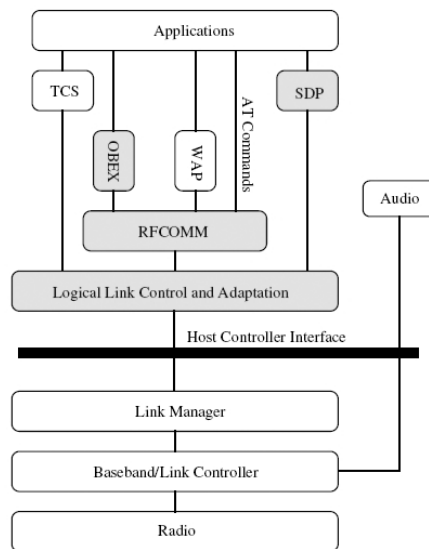


Figura 2.2: Stack dei protocolli Bluetooth

Lo strato radio è quello che si occupa della trasmissione radio e della modulazione, opera nella banda tra i 2,4GHz e i 2,48355 GHz la quale viene divisa in 79 canali da 1MHz, viene usata modulazione GFSK (*Gaussian Frequency Shift Keying*) con un bit per simbolo, raggiungendo la velocità di 1Mbps. Per l'assegnazione equa dei canali viene usata la tecnica FHSS (*Frequency Hopping Spread Spectrum*) con time slots da  $625\mu s$ , con 1600 hops per secondo. Dato che Bluetooth e 802.11 lavorano sulla stessa banda dei 2,4GHz e Bluetooth effettua salti di frequenza più velocemente di 802.11 quest'ultimo ne risente con la conseguenza di forti disturbi con il danneggiamento di connessioni. Questo è un problema che IEEE sta vedendo di risolvere, ma dato che la banda utilizzata è libera da licenze rende difficile la ricerca. Abbiamo varie classi di dispositivi radio, vengono riportate nella tabella 2.1, la maggior parte dei dispositivi è equipaggiata con una radio di classe 3.

Class	Max Output Power	Range
1	100mW (20dBm)	100m+
2	2.5mW (4dBm)	10m
3	1mW (0dBm)	1m

Tabella 2.1: Classi di dispositivi radio

Lo strato baseband è paragonabile al livello MAC, infatti provvede a trasformare i bit in frame e a darli al dispositivo fisico, viene usato TDM (*Time Division Multiplexing*) per definire gli intervalli di comunicazione. Ogni intervallo è di  $625\mu s$ , il master gestisce l'assegnamento degli intervalli, il quale inizia le proprie trasmissioni negli slot pari mentre i nodi slave iniziano negli slot dispari e le comunicazioni possono durare 1,3 o 5 intervalli.

Gli altri strati si occupano di rendere trasparente la comunicazione e offrono vari servizi.

### 2.1.3 JSR-82

Le JSR-82 sono delle API basate sulle specifiche del Bluetooth 1.1 che definiscono l'uso del Bluetooth nella Java 2 Micro Edition (*J2ME*), sono state sviluppate per operare in dispositivi con poca memoria e definiscono le API per i seguenti protocolli:

- L2CAP (Trasmissione di pacchetti)
- RFCOMM (Serializzazione del canale)
- SDP (Identificazione dei servizi disponibili)
- OBEX (Scambio di file)

### 2.1.4 J2ME - Java 2 Micro Edition

J2ME è la piattaforma Java per i dispositivi di consumo come cellulari e PDA che porta la potenza e i benefici di di Java nei dispositivi mobili. Definisce configurazioni e profili per rendere completo l'ambiente di sviluppo ed esecuzione.

Le due configurazioni sono:

- *Connected Limited Device Configuration* - CLDC, è la più piccola delle due, in particolare designata ai cellulari, virtual machine ridotta (KVM)
- *Connected Device Configuration* - CDC, per dispositivi più veloci come PDA di fascia alta, usa una virtual machine completa e supporta un gran parte di J2SE

I tre profili sono:

- *Mobile Information Device Profile* (MIDP), sviluppato per i cellulari, viene combinato con CLDC, le applicazioni risultanti si chiamano MIDlet

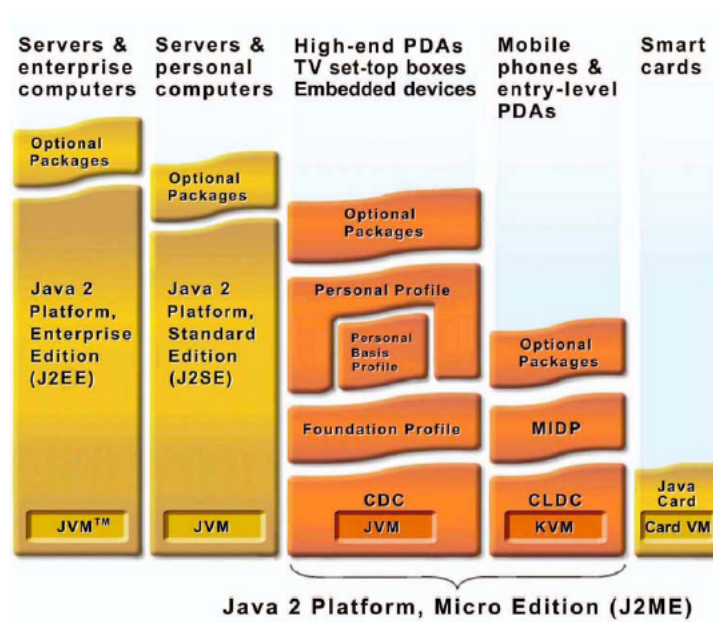


Figura 2.3: Java2 Platform

- *Foundation Profile* (FP), profilo per CDC senza interfaccia utente
- *Personal Profile* (PP), profilo completo di AWT per CDC

### Alternative tralasciate

Altre alternative rispetto a J2ME sarebbero state lo sviluppo con il Compact Framework di Microsoft oppure lo sviluppo per Symbian OS. Lo sviluppo con il toolkit di Microsoft ci ha un po' scoraggiato perché avrebbe vincolato la compatibilità ai soli dispositivi equipaggiati con Windows Mobile, pochissimi smartphone sono dotati di questo sistema operativo, perché è più dedicato ai palmari.

Altrimenti avremmo potuto creare un'applicazione C++ per Symbian OS, questo sistema operativo è più diffuso nei cellulari rispetto a Windows Mobile ma avrebbe in ogni caso ristretto la compatibilità ai soli equipaggiati con Symbian. Abbiamo scelto J2ME rispetto ad altre alternative per il principale motivo che è supportato da quasi tutti i cellulari, a prescindere dal sistema operativo di cui sono dotati, questo ci garantisce la più larga compatibilità.

### 2.1.5 Java 2 Standard Edition

Java è un linguaggio di programmazione orientato agli oggetti, è indipendente dalla piattaforma grazie alla realizzazione di una virtual machine diversa per ogni sistema operativo e architettura. Avremmo potuto sviluppare l'applicazione in C++ ma per garantire una maggiore indipendenza dalla piattaforma abbiamo scelto Java che ci garantisce ampia compatibilità in più sistemi operativi.

### 2.1.6 PHP - Hypertext Preprocessor

PHP è un linguaggio di scripting per il Web, è un linguaggio interpretato che permette la creazione di pagine web dinamiche lato server e non. Alternativa potrebbe essere stata la tecnologia ASP di Microsoft, non supportata però in server Linux, quindi per garantire la più ampia compatibilità è stato scartato. Altra alternativa potrebbe essere stata JSP, strumento molto potente, supporto multi-piattaforma, non è stato scelto solo perché eravamo in possesso di moduli già sviluppati per PHP.

## 2.2 Strumenti Utilizzati

### 2.2.1 Librerie Avetana

La libreria Avetana è una libreria che permette l'uso di una radio Bluetooth da Java sotto i sistemi operativi Linux, Windows e MAC OSX, in Linux si interfacciano alle librerie Bluez ed eseguono un wrapper delle richieste che arrivano da Java alle librerie Bluez. Purtroppo Java da J2SE non permette l'uso di dispositivi Bluetooth e si è dovuti ricorrere a questa libreria a pagamento (di costo contenuto) e ancora in via di sviluppo.

#### Bug rilevato

Durante lo sviluppo è stato rilevato anche un bug nella libreria ed è stato fatto presente alla società sviluppatrice, il bug si presenta sotto sistemi Linux durante l'uso del protocollo RFCOMM che serializza la connessione Bluetooth. In ogni OutputStream seriale di Java compare il metodo flush(), il quale inoltra definitivamente eventuali dati rimasti in coda in attesa di essere trasmessi. Questo metodo della libreria in Linux non funziona, si è dovuti ricorrere ad aspettare un tempo variabile (calcolato euristicamente in seguito a varie prove) prima di chiudere il canale, questo per prevenire il troncamento e l'eliminazione dei dati in coda.

È stato fatto il seguente workaround per garantire il funzionamento in Linux:

---

```
try {
    int milli=((buf.length/40000)+1)*250;
    Thread.sleep(milli);
} catch (InterruptedException ex) {
    ex.printStackTrace();
}
```

---

Non facciamo altro che aspettare 250mS ogni 40Kbyte di dati inviati prima di chiudere la connessione, nei test questo si è rivelato utile perché ha garantito un buon funzionamento.

### 2.2.2 Librerie BouncyCastle

Le Bouncy Castle Crypto API sono delle API per Java leggerissime, consentono cifratura/decifratura con vari algoritmi, possono essere usate sia nella *Java 2 Standard Edition* che nella *Java 2 Micro Edition* proprio grazie alla loro leggerezza.

Sono state scelte proprio per questa ragione, permettono di cifrare e decifrare con RSA in ottimi tempi per un dispositivo con poca potenza di calcolo rispetto ad un calcolatore.

### 2.2.3 Librerie kXML

kXML è una libreria di API che implementano un *pull parser* di codice XML con i vantaggi di SAX e DOM. Questa libreria è molto importante perché riesce a portare la possibilità di fare il parsing di documenti XML nei dispositivi cellulari. Un parser è uno strumento molto potente, senza di esso avremmo dovuto scandire il testo XML token per token e avrebbe reso molto più difficile l'implementazione della lettura.

### 2.2.4 Cellulari Nokia 6630 e 5500

I seguenti cellulari sono stati utilizzati durante i test:

- Nokia 6630
- Nokia 5500 Sport

Sono due cellulari equipaggiati con sistema operativo Symbian, rispettivamente con Symbian S60 e Symbian S60r3, i test hanno prodotto buoni risultati a partire dalla stabilità alle prestazioni.

Sono stati scelti questi due cellulari per il semplice motivo che uno è in mio possesso e l'altro è interno alla mia famiglia.

### **2.2.5 MySql - Database Management System**

MySql è un DMBS relazionale, molto snello e conosciuto per la sua velocità, è dotato di un'interfaccia ODBC che abbiamo utilizzato nel nostro progetto.

Altra alternativa forse più potente sarebbe stato PostgreSQL, però è stato scelto MySql per la sua maggior diffusione e anche perché la quantità dei dati in gioco non è enorme.

### **2.2.6 Netbeans 5.5**

Netbeans è un IDE (Integrated Development Environment) per lo sviluppo software. É dotato di un plugin per dispositivi CLDC per lo sviluppo di MIDlet e per la loro emulazione. Ottima alternativa sarebbe stato l'uso di un altro IDE, Eclipse, che però non offre un buon plugin per lo sviluppo di MIDlet per dispositivi CLDC.

# Capitolo 3

## Descrizione del progetto

### 3.1 Analisi del progetto

Quello che andremo a creare sarà un'applicazione per cellulari che permetterà l'esplorazione di contenuti come un mini-browser che funzioni via Bluetooth, i contenuti saranno distribuiti da un server Bluetooth che dovremo sviluppare, quest'ultimo distribuirà contenuti precedentemente inseriti in una base di dati tramite un'interfaccia web che svilupperemo. Saranno quattro le parti in cui verrà suddiviso il progetto:

- applicazione client per cellulari;
- applicazione server;
- database per i contenuti;
- interfaccia per l'inserimento contenuti.

L'applicazione client sarà un programma per cellulari multilingua che permetterà il browsing tra i contenuti multilingua. In applicazioni reali potrà essere distribuita via web, su richiesta o con una piattaforma automatica di broadcast dell'applicazione (non sviluppata).

L'applicazione server sarà composta da un server Bluetooth residente su un calcolatore che distribuirà i contenuti richiesti che andrà a reperire in una base di dati precedentemente predisposta.

Il database per i contenuti dovrà essere una base di dati che permetterà di contenere le informazioni preannunciate prima nella lista dei contenuti da offrire. L'interfaccia per l'inserimento dei contenuti sarà un'interfaccia web multiutente e multilingua che permetterà l'inserimento dei contenuti descritti in seguito. Queste ultime due parti verranno incluse nella parte relativa alla descrizione del server dato che tutto sommato sono parte integrante di esso.

## Contenuti da gestire

Il sistema che andremo a descrivere potrà gestire informazioni relative ad uno o più comprensori sciistici, questo perché le compagnie di gestione degli impianti sciistici di solito gestiscono più di un comprensorio.

Quindi le informazioni che dovranno essere gestite sono:

- informazioni generali riguardanti le località sciistiche;
- informazioni relative alle piste da sci della località in cui ci si trova e adiacenti;
- eventuali rifugi con o senza alloggi presenti in un comprensorio;
- le news riguardanti il comprensorio in oggetto;
- le previsioni meteo per una migliore programmazione del soggiorno.

Sono tutte informazioni che possono essere visualizzate in un piccolo schermo di cellulare e che possono essere di aiuto per una buona permanenza nella località di vacanza o di visita.

## 3.2 Protocolli di comunicazione

Ora passiamo a parlare della parte comune tra client e server, ovvero del protocollo che useranno per comunicare tra di loro, a partire dal mezzo trasmissivo, al formato delle richieste fino alla sintassi delle risposte.

### 3.2.1 Interconnessione tra client e server

Client e server comunicano usando il mezzo di trasmissione a onde radio Bluetooth definito in 2.1.2, il protocollo usato è RFCOMM, un protocollo che permette di astrarre la connessione tra due dispositivi ad un canale seriale, le procedure di connessione tra i dispositivi rispettano il protocollo definito nelle API Java per il Bluetooth.

Altri due protocolli sono stati presi in considerazione:

**L2CAP:** protocollo di livello inferiore a RFCOMM, questo protocollo consente l'invio di pacchetti con una dimensione prefissata, per l'invio di pacchetti più grandi bisogna arrangiarsi a suddividerli in pacchetti più piccoli, risulta ostico e inutile andare a servirsi di un protocollo di così basso livello quando possiamo avvalerci di un canale seriale.

**OBEX:** alternativa di livello più alto, usata per l'invio di file interi, si appoggia sui protocolli appena enunciati (L2CAP e RFCOMM), questa risultava di livello anche troppo alto perché avremmo dovuto eseguire operazioni in più.

### 3.2.2 Tipi di richieste

Abbiamo tre tipologie di richieste che il client può effettuare verso il server:

- richiesta di autenticazione:  
Client → Server : “VERIFY:<token>”  
Server → Client : “<EncryptedToken>”
- richiesta di un'immagine:  
Client → Server : “img://path/to/image”  
Server → Client : “<image's bytes>”
- richiesta di una pagina:  
Client → Server :  
“<requested page>?[par<sub>1</sub>=<value<sub>1</sub>>]...[par<sub>n</sub>=<value<sub>n</sub>>]”  
Server → Client : “<bytes of requested page>”

### 3.2.3 Formato delle pagine

Le pagine restituite dal server devono essere in linguaggio XML seguendo lo schema HTML, per ora vengono riconosciuti solo i tag essenziali per la creazione di pagine.

Quindi le pagine dovranno avere il seguente schema:

---

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html>
  <head>
    <title>Page's Title</title>
    <command href="href_1" value="value_1" />
    ...
    <command href="href_n" value="value_n" />
  </head>
  <body>
    some text
    <a href="LinkAddress">Link name</a>
    <br />
```

```

</body>
</html>
```

---

Ovviamente all'interno del tag `body` le varie parti possono essere ripetute e mischiate arbitrariamente, unica eccezione per le immagini, per le quali si è deciso di inserirne al massimo una per pagina, altrimenti in uno schermo di cellulare più di un'immagine comprometterebbe la leggibilità.

# Capitolo 4

## Client

### 4.1 Requisiti

Il requisito principale è la realizzazione di un'applicazione per cellulari in Java 2 Micro Edition che permetta di esplorare i contenuti di un server Bluetooth secondo un protocollo ben definito.

Il secondo requisito è che l'applicazione sia molto leggera, quindi non bisognerà appesantirla con tanti accessori grafici, quello che basta per una buona riuscita.

Altro requisito è che l'applicazione sia multilingua, in modo da servire più persone possibile, la selezione della lingua verrà fatta in maniera automatica secondo le impostazioni del dispositivo che comunque sarà modificabile all'interno dell'applicazione.

L'applicazione deve essere sicura e non deve usare server Bluetooth di origine non controllata perché chissà che contenuti potrebbe offrire un intruso.

### 4.2 Classi Realizzate

Di seguito riporteremo una descrizione di tutte le classi realizzate, mentre nella figura 4.1 possiamo vedere il diagramma UML delle principali classi realizzate con i principali metodi, mancano i metodi di servizio per motivi di grafica.

#### Main

Questa classe è una MIDlet, viene invocata all'apertura dell'applicazione, è un Thread che si occupa di avviare tutte le classi necessarie all'esecuzione come il controllo *Control* e il modello *Data*. Questa classe ascolta anche i comandi invocati dall'utente, per ogni tasto premuto viene generato un evento che viene

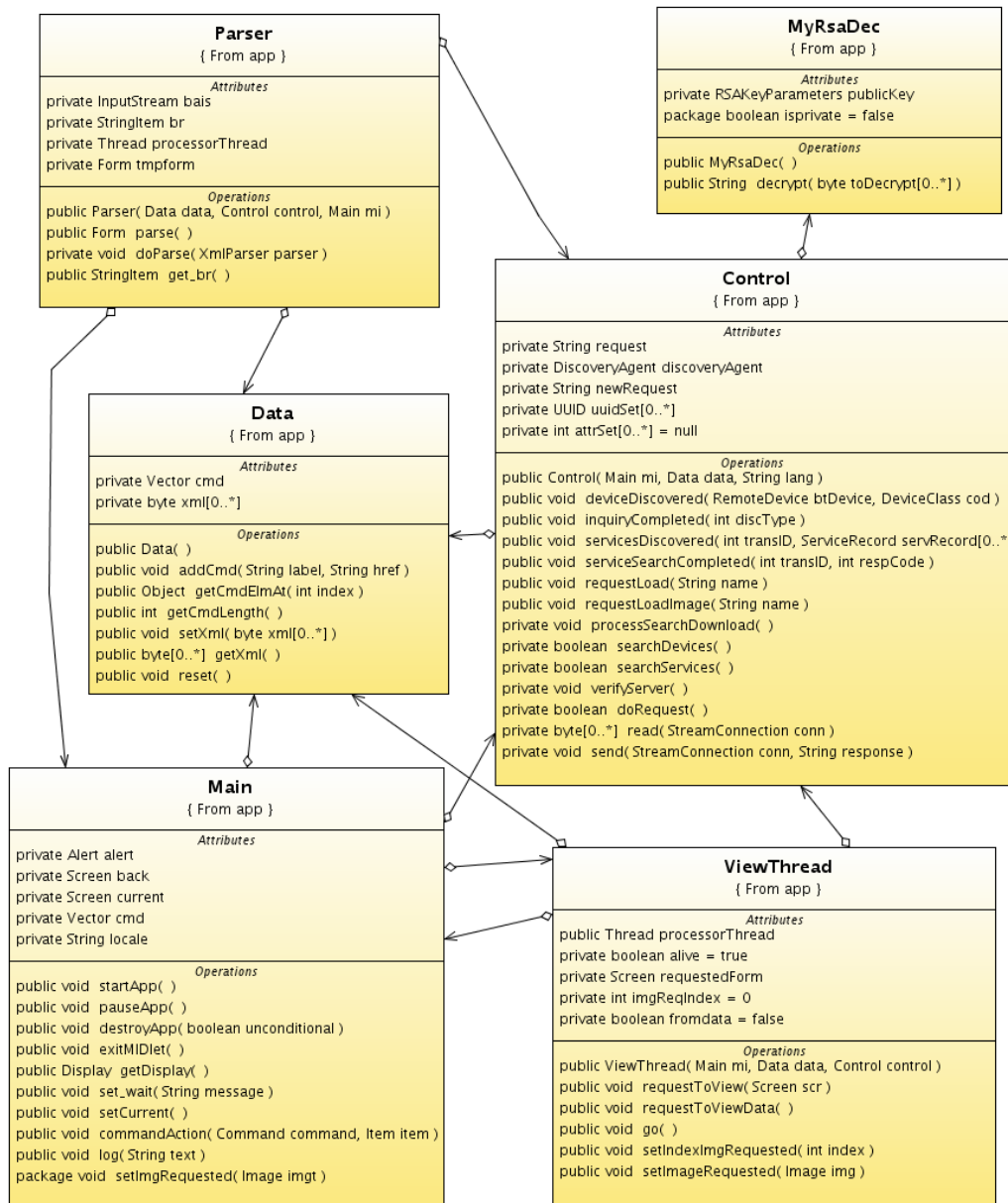


Figura 4.1: Diagramma delle classi principali del Client

gestito direttamente da questa classe. Assieme a `ViewThread` forma la *View* dell'applicazione.

### ViewThread

Questa classe è un `Thread` che si occupa di rendere visibile qualsiasi form su richiesta di *Main*, può visualizzare form già presenti come quelli di servizio o può creare *Parser* per costruire un form a partire dai dati di *Data* e in seguito visualizzarlo.

Questo è un `Thread` indipendente, questo per garantire maggior efficienza nell'esecuzione dell'applicazione.

### Parser

Questa classe si occupa di offrire i metodi necessari per il parsing del contenuto del modello (*Data*) usando la libreria `kXML`, durante il parsing viene costruito il form relativo alla pagina contenuta nel codice XML del modello che poi verrà visualizzato da *ViewThread*.

Il codice seguente mostra come viene effettuato il parsing del modello con la costruzione del relativo form, interessante vedere come le librerie `kXML` rendono molto semplice l'operazione.

---

```
private Form tmpform;

public Form parse(){
    InputStream is=new ByteArrayInputStream(data.getXml());
    try{
        InputStreamReader xmlReader = new InputStreamReader(is);
        XmlParser parser = new XmlParser( xmlReader );
        doParse(parser);
    }catch(Exception e){
        // Errore
    }
    return tmpform;
}

private void doParse(XmlParser parser)throws Exception{
    tmpform=new Form("");
    boolean leave = false;
    do {
        ParseEvent event = parser.read ();
```

```
ParseEvent pe;
switch ( event.getType() ) {
    // For example, <title>
    case Xml.START_TAG:
        if ("title".equals(event.getName())){
            pe = parser.read();
            tmpform.setTitle(pe.getText());
        }else if ("other tags".equals(event.getName())){
            .
            .
        }
        break;
    // For example </body>
    case Xml.END_TAG:
        leave = false;
        break;
    // For example </html>
    case Xml.END_DOCUMENT:
        leave = true;
        break;
    // For example, the text between tags
    case Xml.TEXT:
        String str=event.getText();
        break;
    default:
}
} while( !leave );
}
```

---

## Control

Questo è un Thread che rappresenta il *Controller* dell'applicazione, si occupa di gestire tutte le richieste fatte dalla *View*, ha il ruolo di ricercare i dispositivi Bluetooth, effettuare la ricerca dei soli dispositivi che offrono il servizio richiesto, autenticare i server trovati e di inviare la richiesta di pagine e immagini ai server trovati, a richiesta esaudita aggiorna il *Model* e avvisa la *View* che il modello è stato aggiornato e la richiesta processata.

All'avvio si mette in attesa di nuove richieste, all'arrivo di una richiesta della vista se necessario ricerca i dispositivi server ed autentica gli eventuali trovati, poi esegue la richiesta ed infine si rimette in attesa di una prossima, nelle

successive non esegue la ricerca dei dispositivi a meno che neanche uno dei dispositivi trovati precedentemente sia disponibile.

### MyRsaDec

Questa è una classe di servizio che utilizza le librerie BouncyCastle, contiene una chiave pubblica per l'algoritmo di cifratura RSA e ha dei metodi che eseguono la decifrazione di array di byte usando l'algoritmo RSA, questa classe serve a *Control* quando esegue la verifica dei server.

Di seguito sono riportati i passaggi fondamentali per la decifrazione di un array di byte, usato nell'autenticazione dei server.

---

```
RSAKeyParameters publicKey=new RSAKeyParameters(...);

public String decrypt(byte[] toDecrypt,int offset,int length)
    throws InvalidCipherTextException{
    AsymmetricBlockCipher theEngine = new RSAEngine();
    theEngine.init(false, publicKey);
    byte[] dec;
    dec=theEngine.processBlock(toDecrypt, offset, length);
    return new String(dec);
}
```

---

Interessante vedere come le librerie BouncyCastle rendono molto semplice la decifrazione con chiave pubblica e algoritmo RSA.

### Data

Questo è il *Model* dell'applicazione, contiene i dati che devono essere visualizzati, in particolare contiene il testo XML dell'ultima pagina di contenuto richiesta al server. Al termine di ogni richiesta il controllo aggiorna il contenuto del modello con il codice XML della nuova pagina e avvisa la vista del cambiamento, la quale esegue il parsing e crea il form.

### LocalizationSupport

Questa è una classe di servizio che permette di rendere multi lingua l'applicazione, a partire da dei file di stringhe in varie lingue costruisce una tabella hash contenente le stringhe del programma nella lingua richiesta. La lingua viene scelta a seconda della lingua impostata dal dispositivo, che viene specificata in una delle proprietà del sistema che viene reperita tramite la seguente istruzione:

```
System.getProperty("microedition.locale");
```

### About

Questo è un form di servizio che visualizza informazioni generali sull'applicazione (Figura 4.3(a)).

### Logger

Questo è un form che visualizza il log dell'applicazione generato da *Control* (Figura 4.3(b)), più che all'utente finale è stato utile durante lo sviluppo dato che nei dispositivi non si ha lo standard output e una console di debug.

### MainPage

Questo è il form principale, è anche il menù dell'applicazione, viene presentato all'avvio dell'applicazione dopo lo Splash, da qui si può entrare nella navigazione, o si può accedere agli altri form di servizio (Figura 4.2(a)).

### Setting

Questo è un form che visualizza e permette di modificare le impostazioni dell'applicazione, in particolare da qui si può selezionare la lingua preferita per i contenuti e si può scegliere se visualizzare o meno le immagini all'interno delle pagine, quest'ultima opzione per offrire l'opportunità di navigare più velocemente perché evita la richiesta delle immagini al server (Figura 4.2(b)).

### Splash

Questo Form funge da Splash e viene visualizzato per una manciata di secondi all'avvio dell'applicazione, è più di presentazione, non di utilità.

### WaitForm

Questo è il form che viene visualizzato quando si mette in attesa l'utente, visualizza una barra animata e un messaggio che varia a seconda dell'operazione che stiamo eseguendo. Il messaggio cambia a seconda dell'operazione svolta per non dare all'utente l'impressione che l'applicazione si sia bloccata ma per dare un'idea dello stato dell'applicazione.

## 4.3 Scelte Implementative

### 4.3.1 Design

Il design pattern scelto è *Model-View-Controller*, i cui ruoli sono stati assegnati nella precedente descrizione delle classi, come modello abbiamo scelto il codice XML della pagina richiesta, formalmente sarebbe stato più corretto fare del modello una rappresentazione dei dati da visualizzare indipendenti dalla vista, ma questo ci avrebbe portato ad uno spreco di risorse perché avremmo dovuto eseguire doppi passi per la costruzione dei form. Mettendo invece il codice XML è la vista ad occuparsi del parsing e della visualizzazione, facendo così un passo in meno, abbiamo perso un po' di formalità ma abbiamo guadagnato molto nelle prestazioni e nell'uso della memoria che è preziosa nei dispositivi mobili.

### 4.3.2 Multi-Thread

L'applicazione sviluppata è composta da 3 Thread indipendenti, questo per garantire che l'applicazione non si blocchi durante l'uso, nei dispositivi cellulari è molto importante far uso dei Thread perché hanno una potenza di calcolo limitata e una piccola operazione può causare l'apparente blocco, molto fastidioso per un utente che ne fa uso.

Quindi avremo 3 Thread principali:

**Controllo:** questo thread si occupa di eseguire le ricerche dei dispositivi e di eseguire tutte le richieste;

**Vista:** si occupa di visualizzare tutti i form sullo schermo, importante che sia un thread indipendente per avere l'applicazione sempre fluida e reattiva;

**Ascoltatore dei comandi:** si occupa di ascoltare tutti gli input dell'utente, Controllo ed Ascoltatore dei comandi in un MVC classico coinciderebbero, sono stati separati per garantire la reattività dell'applicazione e per lasciare libera la visualizzazione dei form, dato che nei dispositivi mobili è molto importante la suddivisione del lavoro in Thread distinti.

### 4.3.3 Sicurezza

Per la sicurezza dei server utilizzati abbiamo usato un metodo di autenticazione basato sul principio di chiave asimmetrica e di non modifica del codice in esecuzione. Il client è in possesso di una chiave pubblica resa lecita dal presupposto che il codice non sia stato modificato, questo può essere garantito

con la firma del codice generato con un certificato a pagamento rilasciato da una autorità certificante. Quando un client vuole verificare un server gli invia un token random calcolato al momento, il server per autenticarsi dovrà cifrare questo token con la propria chiave privata e rispedirlo al client, il client prende il pacchetto ricevuto, lo decifra e se il risultato è il token inviato precedentemente allora il server è autentico e può essere utilizzato per le successive richieste. L'algoritmo di cifratura utilizzato è RSA, un algoritmo a chiave asimmetrica che garantisce una sicurezza computazionale elevata.

- **Client**  $\rightarrow$  **Server** : TokenRandom
- **Server**  $\rightarrow$  **Client** : Encrypt( PR(Server) , TokenRandom )
- **Client** : result := Decrypt( PU(Server), Encrypt( PR(Server) , TokenRandom ) )
- **Client** : *If* result  $\equiv$  TokenRandom *then* !ServerOK!

## 4.4 Funzionamento

Dopo aver installato l'applicazione, la si troverà nel menù del dispositivo, ogni casa costruttrice posiziona le icone dei nuovi programmi in modo diverso, non è decidibile la posizione.

Quando si avvia l'applicazione appare uno Splash di benvenuto al cui interno è contenuta la scritta del programma, dopo un breve timeout viene visualizzato il menù principale (Figura 4.2(a)), da qui possiamo accedere a tutte le operazioni possibili.

Accedendo alle impostazioni dell'applicazione ci viene presentato il form di figura 4.2(b), da qui è possibile selezionare la lingua preferita, si può decidere se caricare o meno le immagini durante la navigazione e poi possiamo salvare le nuove impostazioni tornando al menù principale.

Accedendo alle informazioni generali viene visualizzato il form in figura 4.3(a), qui si possono leggere informazioni relative all'applicazione.

Attraverso il menù dei comandi possiamo accedere al log, figura 4.3(b), qui vengono stampati i messaggi relativi alle operazioni svolte e agli eventuali errori o eccezioni.

Accedendo alla navigazione viene caricata la home page dei contenuti, un esempio è la figura 4.4(b). Se è la prima volta che accediamo alla navigazione dopo l'avvio dovremo attendere un po' perché come viene visualizzato nel messaggio di attesa (Figura 4.4(a)), verrà effettuata la ricerca dei server che offrono il servizio da noi richiesto, la ricerca è composta da due parti standard relative

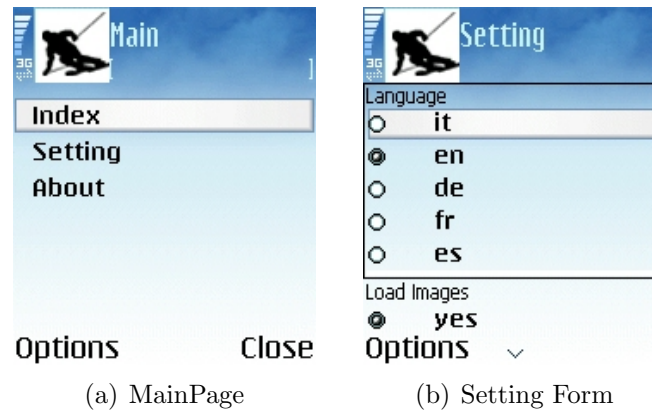


Figura 4.2: Snapshot della MIDlet - 1



Figura 4.3: Snapshot della MIDlet - 2

al protocollo Bluetooth più una fase di verifica per l'autenticazione dei server trovati.

Conclusa l'operazione di ricerca, e ammesso che sia stato trovato almeno un server lecito verrà inviata la richiesta al server, dal quale riceveremo come risposta il codice XML relativo alla pagina richiesta, a questo punto verrà eseguito il parsing del codice ricevuto, a partire dal quale andremo a costruire il form rappresentante che poi visualizzeremo, se nella pagina compaiono immagini queste vengono caricate in modo asincrono in un thread apposito concorrentemente all'operazione di parsing e di visualizzazione.



Figura 4.4: Snapshot della MIDlet - 3

Se l'operazione di richiesta ad uno dei server fallisce viene tentata in tutti i server individuati, se nessuno di questi esaudisce la richiesta allora si andrà alla ricerca di nuovi server in grado di processare la richiesta, questo per garantire il miglior funzionamento, la maggiore disponibilità delle informazioni e per garantire il funzionamento quando l'utente si sposta in aree coperte da server diversi.

Di seguito, nella figura 4.5 possiamo vedere il sequence diagram di richiesta di una pagina ad un server Bluetooth dopo aver già compiuto la ricerca e verifica dei server.

Se il codice da cui si installa l'applicazione non è stato firmato con un certificato valido di un'autorità certificante riconosciuta dal dispositivo, al momento del primo tentativo di accesso al canale Bluetooth verrà chiesto all'utente se vuole consentire o meno all'applicazione di accedere al canale Bluetooth (Figura 4.6).

Questo non è stato imposto da noi ma bensì dalle impostazioni di sicurezza di Java 2 Micro Edition, per le applicazioni non firmate viene chiesta autorizzazione all'utente per accedere al filesystem, al Bluetooth e così via, questo per

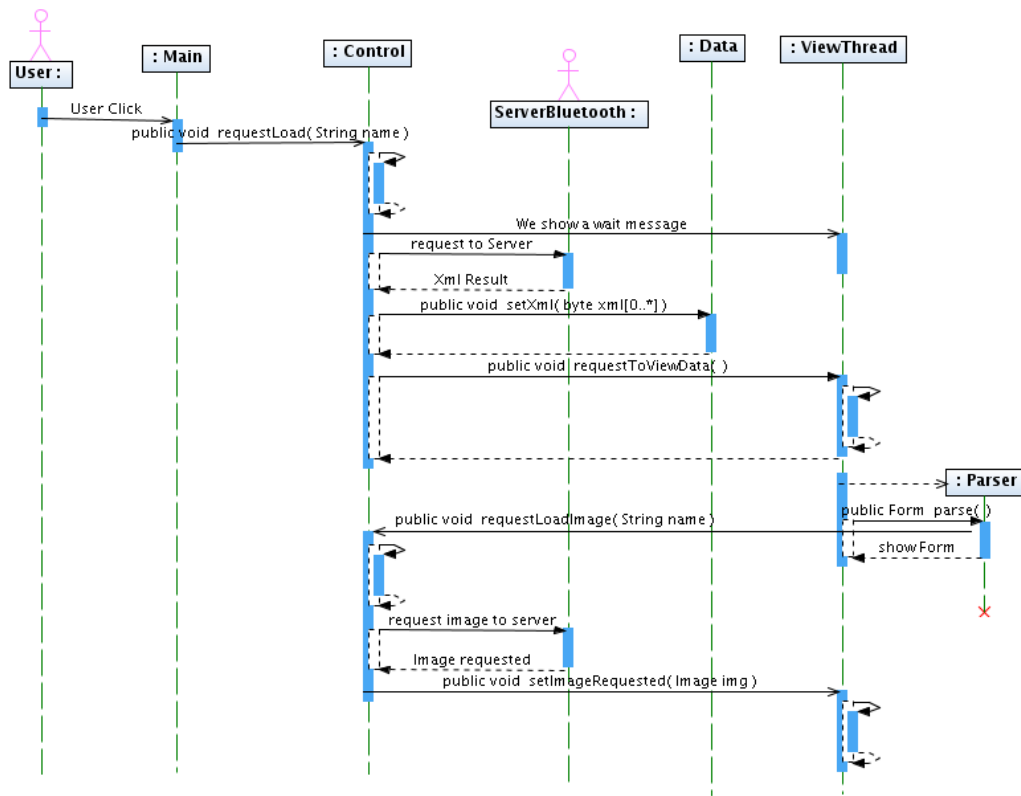


Figura 4.5: Sequence Diagram di inoltro di una richiesta



Figura 4.6: Richiesta di autorizzazione di accesso al canale Bluetooth

non consentire ad applicazioni malevole di danneggiare i nostri dati e quelli degli altri dispositivi.

# Capitolo 5

## Server

La parte server è composta da più parti, oltre all'applicazione che dovrà fungere da server dei contenuti e gestire le richieste, è composta anche dal database dei contenuti e dall'interfaccia necessaria per l'inserimento dei dati nel database. Di seguito descriveremo le varie parti senza entrare troppo nei dettagli per non creare confusione nella generica comprensione del progetto.

### 5.1 Server

#### 5.1.1 Requisiti

Il requisito principale è la realizzazione di un'applicazione che funga da server dei contenuti, dovrà avviarsi e rimanere in attesa di richieste, al loro arrivo dovrà poi servirle secondo il tipo di richiesta ricevuta.

L'applicazione dovrà essere efficiente e dovrà gestire al meglio tante richieste che potranno arrivare molto ravvicinate, sarà importante la buona gestione del canale Bluetooth e sarà necessario implementare una coda per gestire le richieste.

#### 5.1.2 Classi Realizzate

Di seguito riporteremo una descrizione di tutte le classi realizzate, mentre nella figura 5.1 possiamo vedere il diagramma UML delle principali classi realizzate con i principali metodi, sono stati esclusi i metodi di servizio per motivi di grafica e per rendere più semplice la lettura.

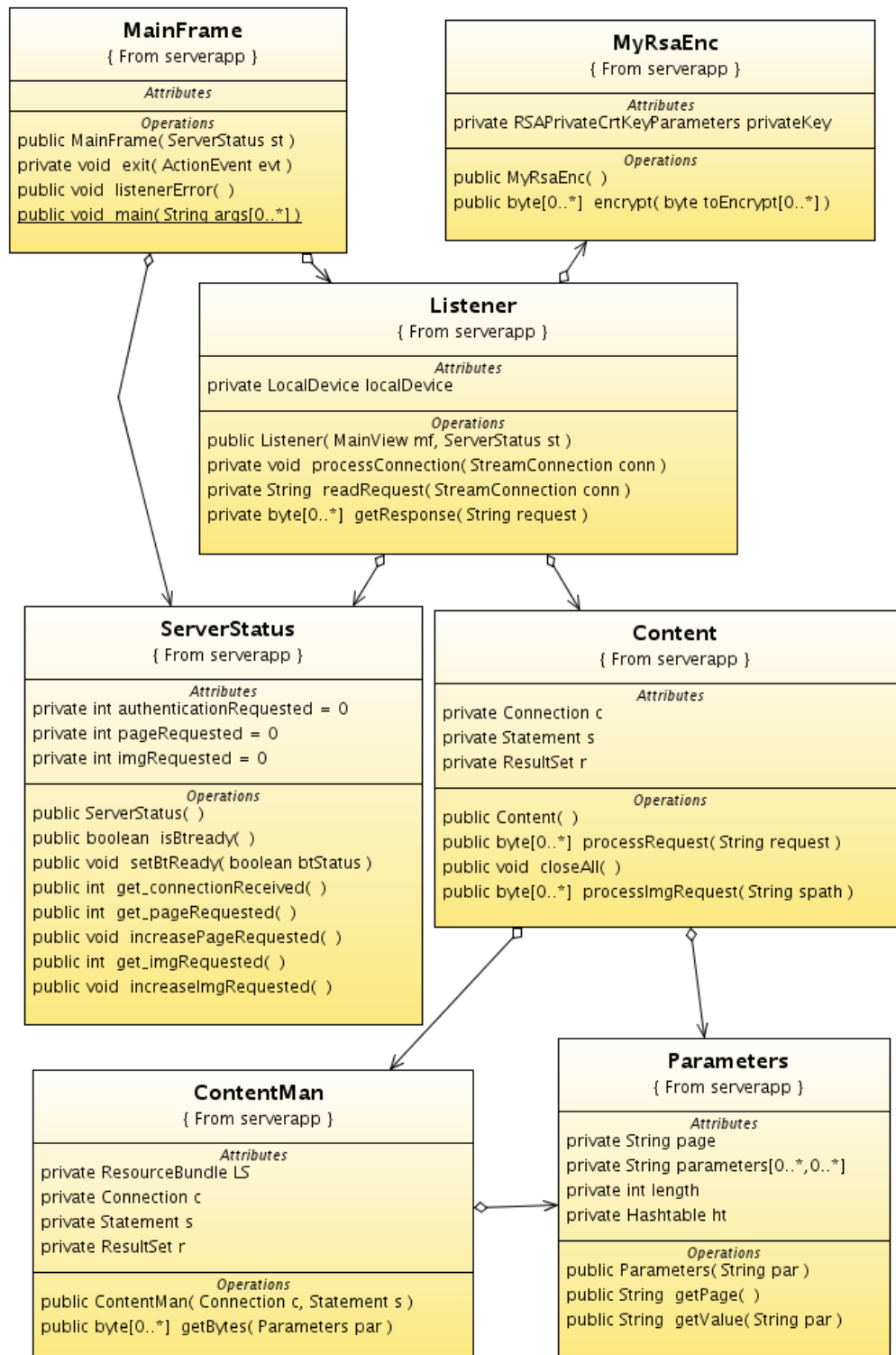


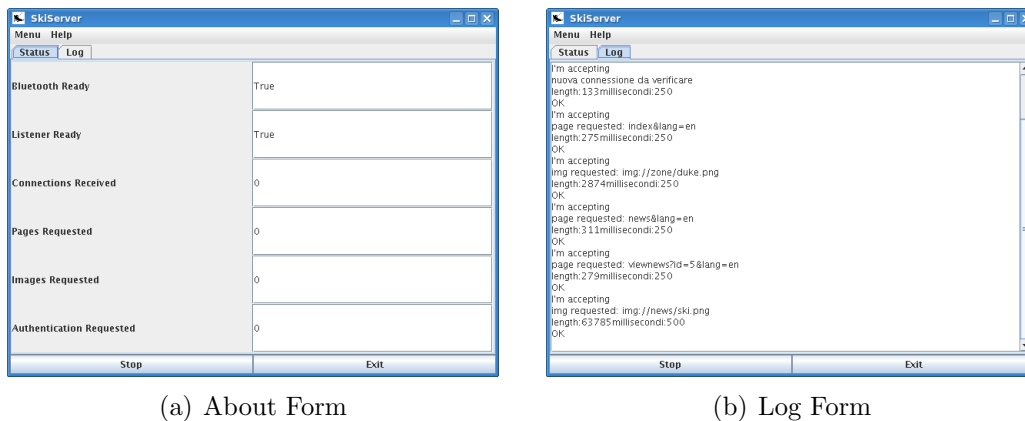
Figura 5.1: Diagramma delle classi principali del server

## MainView

È un'interfaccia che rappresenta la *View* e i metodi che bisogna implementare. Per costruire una vista bisogna implementare questa interfaccia e i relativi metodi.

## MainFrame

Questa classe implementa una *View* per il server con GUI, si occupa di avviare tutte le classi necessarie e apre la finestra che permette di fermare e riavviare il server e di visualizzare lo stato (Figura: 5.2(a)) e il log (Figura: 5.2(b)) relativi al server.



(a) About Form

(b) Log Form

Figura 5.2: Snapshot di MainFrame

## MainConsole

Questa classe è un'altra implementazione della vista, che è utile per l'avvio da console del servizio perché non ha interfaccia grafica, semplicemente avvia il server e stampa in output il log nello standard output.

## ServerStatus

Questo è il *Model* dell'applicazione, rappresenta lo stato del server con un insieme di variabili, viene aggiornato dal *Controller*, quando lo stato viene modificato viene informata la vista ed eventuali altri osservatori dei cambiamenti.

## Listener

Questo è un Thread che si occupa di aprire il servizio Bluetooth, di mettersi in attesa delle richieste e di processarle una ad una. Implementa una coda per contenere i tentativi di richiesta che poi uno ad uno serve spedendo le informazioni richieste. Questo è il *Controller* dell'applicazione perché si occupa di tenere aggiornato lo stato del server richiesta dopo richiesta.

## MyRsaEnc

Questa è una classe di servizio che utilizza le librerie BouncyCastle, permette di cifrare con l'algoritmo RSA e la chiave privata del server degli array di byte, che nel caso del server servono per l'autenticazione descritta precedentemente. La cifratura con chiave privata viene effettuata nel seguente modo:

---

```
private RSAPrivateCrtKeyParameters privateKey;

public byte[] encrypt(byte[] toEncrypt,int offset,int length)
    throws InvalidCipherTextException{
    privateKey=new RSAPrivateCrtKeyParameters(...);
    AsymmetricBlockCipher theEngine = new RSAEngine();
    theEngine.init(false, privateKey);
    return theEngine.processBlock(toEncrypt, offset, length);
}
```

---

## Content

Questa è una classe intermedia per la gestione dei contenuti, nel caso venga richiesta una pagina crea la connessione al database MySql e delega la gestione della richiesta alla classe *ContentMan*, mentre nel caso venga richiesta un'immagine viene processata direttamente leggendo il file richiesto e restituendolo al *Listener*.

## Parameters

Questa è una classe che permette di gestire le stringhe di richiesta dei client, ne prende in input una e mette a disposizione dei metodi per ottenere da essa la pagina richiesta e i vari parametri che vengono specificati come in HTTP con metodo GET.

## ContentMan

Questa classe si occupa di gestire i contenuti a partire da un'istanza di *Parameters*, viene restituito il codice XML a seconda della pagina e dei parametri specificati. Modificando questa classe è possibile gestire e offrire qualsiasi tipo di contenuto, per esempio si può specializzare l'applicazione per altri contesti.

### 5.1.3 Scelte Implementative

Il design pattern utilizzato è lo stesso usato per il client, è il *Model-View-Controller*, è la scelta migliore per questa applicazione perché i tre ruoli sono ben distinti ed è quello che basta per offrire buone potenzialità. Come controllo non potevamo che scegliere la parte che si occupa di stare in ascolto delle richieste e di servirle che quindi è il *Listener*, come modello ovviamente lo stato dell'applicazione che si è rivelato nella classe *ServerStatus* e infine la vista rappresentata da un'interfaccia, *MainView*, in modo che possa essere implementata in più varianti, ne abbiamo viste due, una con GUI e una a linea di comando, interessante in sviluppi futuri è l'aggiunta di nuove viste.

Java 2 Standard Edition non implementa l'accesso al canale Bluetooth, per questo siamo dovuti ricorrere a delle librerie, citate in 2.2.1, siamo ricorsi a queste perché hanno un prezzo contenuto ed offrono un buon servizio, a parte lo sviluppo che è ancora in corso dato il bug che è stato rilevato (se ne parla sempre in 2.2.1).

Per la cifratura abbiamo usato le stesse librerie usate dal client, le Bouncy-Castle (vedi: 2.2.2), perché si sono rivelate una buona scelta, data la loro leggerezza e l'offerta di servizio.

Ovviamente in esecuzione ci sono 2 Thread, uno per la vista e uno per la gestione delle richieste, altro non si poteva fare, perché le richieste vanno processate una alla volta.

### 5.1.4 Funzionamento

All'avvio viene creata la vista, il modello relativo allo stato e avviato il Thread del Listener che apre un servizio Bluetooth con l'UUID conosciuto da client e server, il quale si mette in attesa di ricevere connessioni, alla richiesta di connessione da parte di un dispositivo lo si mette in una coda da processare. Quando si processa una richiesta si apre la connessione con il dispositivo interessato, si legge la stringa di richiesta, la si esamina e a seconda di questa si elabora la risposta che viene inviata al client.

Se è una richiesta di autenticazione si cifra il token con RSA tramite la classe *MyRsaEnc*, altrimenti viene passata la gestione della richiesta a *Content* che

in caso venga richiesta una pagina istanzia *ContentMan* per ottenere il risultato, altrimenti in caso di una immagine la restituisce lui stesso. In figura 5.3 possiamo vedere il sequence diagram di gestione di richiesta di una pagina di contenuto.

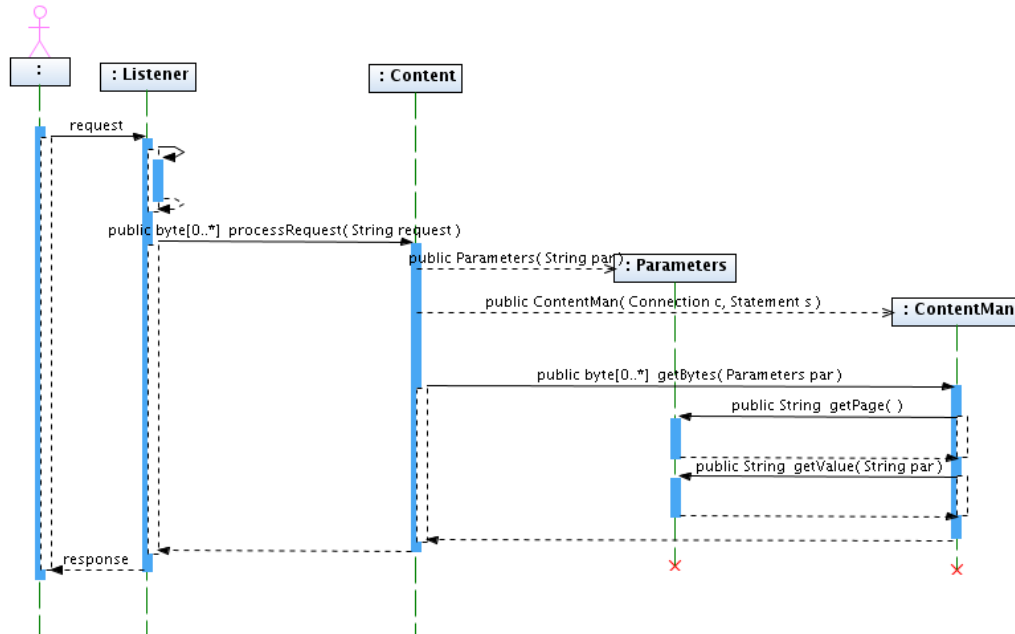


Figura 5.3: Sequence Diagram di gestione di una richiesta

## 5.2 Database dei contenuti

### 5.2.1 Requisiti

Bisogna realizzare un database per i contenuti, dovrà contenere le seguenti informazioni riguardanti possessori di comprensori sciistici:

- informazioni generali riguardanti ogni comprensorio sciistico, meglio definito come zona sciistica, ogni zona dovrà avere un nome, una descrizione ed eventualmente un'immagine che la rappresenti;
- informazioni generali riguardanti le località appartenenti alle zone, chiamate subzone, ogni subzona dovrà avere un nome, una propria descrizione, un'eventuale immagine rappresentante ed un'immagine che sia la mappa delle piste da sci;

- informazioni relative alle piste da sci delle varie subzone, ogni pista avrà un nome, una descrizione e un livello di difficoltà, scelto tra: blu, rosso, nero;
- informazioni sui rifugi presenti in una subzona, ogni rifugio avrà un nome, un'immagine e una descrizione;
- le news riguardanti le zone, ogni news avrà un titolo, una data di inserzione, un testo ed un'eventuale immagine;

Le informazioni sopracitate dovranno essere memorizzate in una o più lingue a scelta, in modo da offrire un servizio più ampio.

### 5.2.2 Analisi e progettazione

Analizzando i requisiti siamo giunti allo schema ER in figura 5.4. Nel passaggio da modello ER al modello relazionale viene aggiunta una sola entità oltre a quelle specificate, la relazione molti a molti tra le entità *text* e *lang* nel passaggio al modello reale diventa una nuova entità con chiave primaria data dalle due chiavi esterne delle tabelle partecipanti, unico altro attributo è quello specificato nella relazione.

È stata creata un' entità per ogni contenuto che dobbiamo gestire (zone, subzone, rifugi, piste e news), dato che ogni contenuto dovrà essere multi lingua e ogni contenuto possiede una descrizione, questo deve possedere una o più descrizioni nelle varie lingue.

Abbiamo voluto centralizzare la gestione di testi multi lingua, così abbiamo creato una tabella di identificatori che sono univoci all'interno dell'intero insieme di tutti i contenuti, ogni identificatore individua il codice di un testo per il quale esisteranno uno o più testi in varie lingue.

Le lingue sono mantenute in un' entità apposita in modo da rendere il più facile possibile l'aggiunta di una nuova lingua, che richiede semplicemente l'inserimento di una nuova tupla e l'inserimento del testo in nuova lingua per i contenuti interessati.

Il database risulta essere poco leggibile guardando singolarmente le tabelle, a questo si può ovviare creando delle viste in modo da rendere più facile la consultazione dal lato amministratore.

### 5.2.3 Implementazione

Il database è stato implementato sul DBMS MySQL, descritto in 2.2.5, è stato scelto per la sua leggerezza e velocità. Il server ha accesso ad esso tramite il driver JDBC messo a disposizione dal produttore.

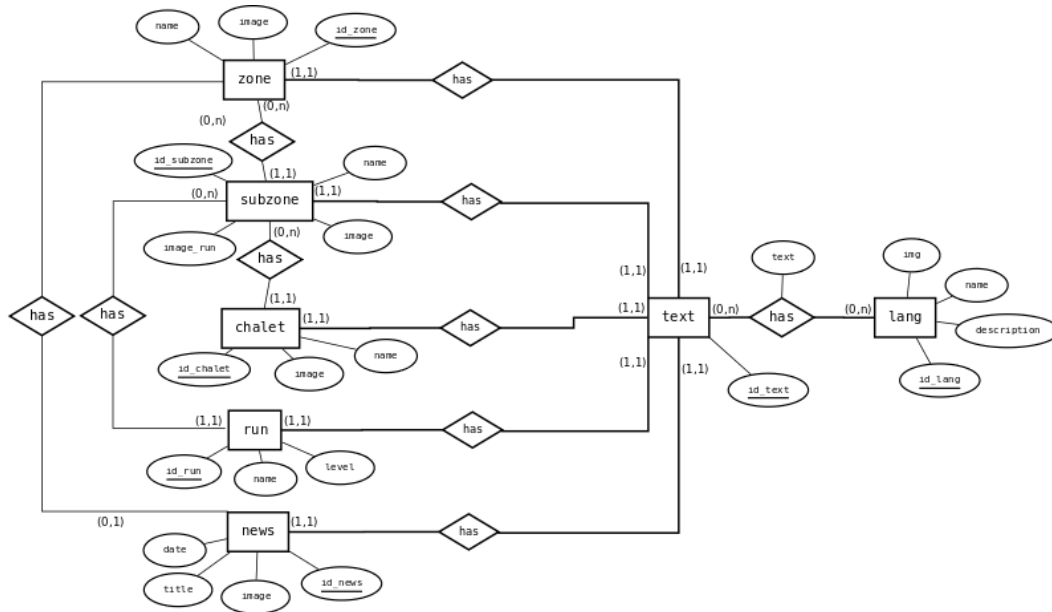


Figura 5.4: Schema ER del database dei contenuti

## 5.3 Interfaccia inserimento contenuti

### 5.3.1 Requisiti

Il requisito principale è la realizzazione di un'interfaccia web per l'inserimento dei dati nel database dei contenuti (vedi 5.2), l'interfaccia dovrà essere accessibile solo a certi utenti in possesso di un nome utente e una password e dovrà essere possibile scegliere per ogni utente le operazioni a cui lo si vuole abilitare. Consigliabile è rendere l'interfaccia multi tema e multi lingua in modo da rendere maggiormente personalizzabile e più facile l'inserimento da parte degli operatori.

### 5.3.2 Implementazione

L'interfaccia è stata sviluppata in PHP (vedi 2.1.6), un linguaggio di scripting lato server per il Web, è stato scelto per la sua semplicità e duttilità.

L'accesso all'interfaccia è protetto con un sistema di login che richiede un nome utente e una password, dopo aver eseguito l'accesso si accede al pannello di amministrazione dal quale si può accedere alle varie parti.

L'interfaccia è divisa in 4 parti principali:

- la testata;

- il menù;
- la parte principale;
- il bottom.

L'interfaccia è multi lingua, sono state sviluppate la lingua italiana e la lingua inglese, lo sviluppo di altre lingue consiste nella sola creazione di un file contenente le stringhe per la nuova localizzazione.

L'interfaccia è anche multi tema, ce ne sono di vari colori, in modo che ogni operatore addetto all'uso possa scegliere la tonalità più rilassante.

Gli utenti sono divisi in due fasce principali, i super user e gli utenti normali, i primi hanno il permesso di aggiungere nuovi utenti, di modificare i permessi e hanno il pieno controllo su ogni singola parte del pannello di amministrazione. Gli utenti normali invece sono abilitati alle sole operazioni a loro concesse, per ogni categoria di inserimento possono venire dati i permessi di inserimento, di modifica e di cancellazione, un esempio è la figura 5.5. Le informazioni riguar-

Operazioni Concesse			
	Inserire	Modificare	Eliminare
In <b>zone</b> puoi:	X	X	X
In <b>subzone</b> puoi:	X	X	X
In <b>chalet</b> puoi:	X	X	X
In <b>run</b> puoi:	X	X	X
In <b>news</b> puoi:	X	X	X
In <b>lingue</b> puoi:	X	X	X

Figura 5.5: Riassunto dei permessi

danti gli utenti e quelle necessarie all'interfaccia di inserimento sono contenute in apposite tabelle all'interno del database dei contenuti.

### La testata

La testata, figura 5.6, contiene il banner dell'applicazione, tra cui il suo logo, il suo titolo e due loghi delle tecnologie utilizzate.

Nel riquadro in basso a sinistra è possibile selezionare la lingua preferita per l'interfaccia tra quelle disponibili.

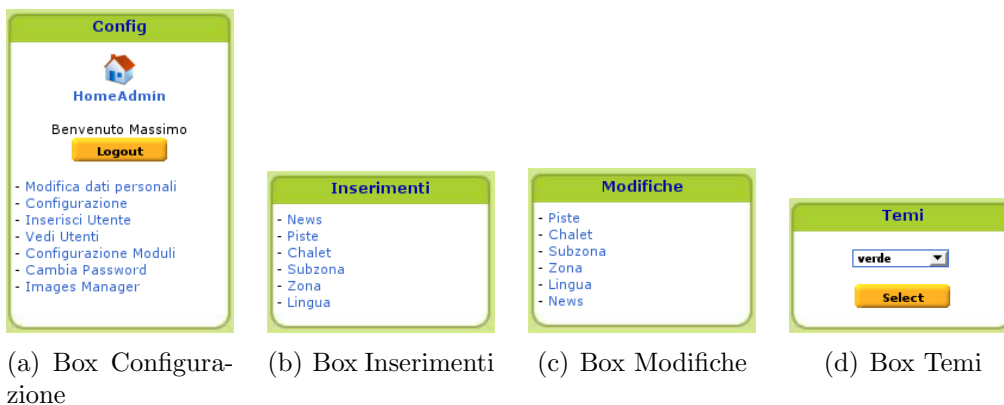
### Il menù

Il menù è diviso in quattro parti:



Figura 5.6: Testata dell'interfaccia

1. Il box delle configurazioni, figura 5.7(a), permette di accedere alle informazioni generali e di modificarle, di cambiare la password di accesso e di accedere al file manager per le immagini. Agli utenti super user permette anche di accedere all'inserimento di nuovi utenti, alla modifica dei permessi dei vari utenti e ad alcune impostazioni di configurazione. Permette anche di eseguire il logout a tutti gli utenti;
2. Il box degli inserimenti, in figura 5.7(b), permette di accedere alle pagine d'inserimento relative ad ogni categoria presente in lista, ammesso di avere i permessi di inserimento;
3. Il box delle modifiche e dell'esplorazione. Da qui è possibile andare ad esplorare tutte le categorie elencate e in seguito modificarne gli elementi, sempre se si è in possesso dei permessi necessari, questo box lo possiamo vedere in figura 5.7(c);
4. Il box dei temi, da qui è possibile selezionare il tema scelto da un menù a tendina e cliccando il pulsante di selezione, lo possiamo vedere in figura 5.7(d).



(a) Box Configurazione

(b) Box Inserimenti

(c) Box Modifiche

(d) Box Temi

Figura 5.7: Menù dell'interfaccia

## Il bottom

Il bottom è la parte bassa della pagina in cui appaiono le icone delle tecnologie e degli strumenti utilizzati, lo possiamo vedere nella figura 5.8.



Figura 5.8: Bottom - Parte bassa dell'interfaccia

## Parte principale

Ultima parte e di maggior importanza è la parte centrale e più grande della schermata in cui andranno ad inserirsi tutte le pagine che permettono l'inserimento, la modifica e l'esplorazione degli elementi delle varie categorie.

Utenti							
User	Nome	Cognome	Mail	Theme	Super User	Edit	Priv Del
admin	admin	admin	admin@admin.it	verde	y		
demo	demo	demo	demo@demo.it	orange	n		
massimo	Massimo	Paladin	massimo85@gmail.com	verde	y		

Figura 5.9: Lista degli utenti

Selezione Zona			
Tutte le zone			
Cortina	Plan de Coronas	Tre Valli	Civetta

News			
Data	Titolo	Zona	Operazioni
22/04/2007	prova news tre valli	Tre Valli	
21/04/2007	Prova news generale	Tutte	
20/04/2007	prova news civetta	Civetta	
19/04/2007	news cortina	Cortina	
18/04/2007	Prova news generale 2	Tutte	
18/04/2007	prova news plan	Plan de Coronas	

1

Figura 5.10: Lista delle news

Un esempio di parte principale che visualizza le operazioni a cui si è abilitati è quella riportata in figura 5.5. Un altro esempio è quello riportato in figura 5.9 che visualizza la lista degli utenti e permette di accedere alla loro modifica e alla loro eliminazione. Ultimo esempio riportato è quello in figura 5.10, che riporta la lista delle news di tutte le zone e permette di selezionare la zona interessata in modo da restringere il campo di ricerca.

# Capitolo 6

## Conclusione e futuri sviluppi

### 6.1 Conclusioni

Il problema principale è stato l'approccio ad una tecnologia abbastanza nuova e senza uno standard come il Bluetooth, questo ha richiesto un continuo test durante la progettazione in modo da garantire il funzionamento su dispositivi reali. Il semplice test sull'emulatore non sarebbe bastato perché differisce dalle implementazioni che vengono fatte sui dispositivi reali, certe cose che su emulatore funzionavano sul cellulare creavano problemi.

Nello sviluppo non si sono verificati grossi problemi, a parte qualche piccolo inconveniente nella compatibilità tra vari dispositivi nell'uso del Bluetooth, tutti i problemi comunque sono stati risolti, purtroppo non è ancora uno standard e le implementazioni delle API differiscono causando qualche piccola difficoltà, che comunque è risultata risolvibile.

Interessante è stata anche l'attenzione a sviluppare un'applicazione il più leggera possibile, tendenzialmente nel programmare per un calcolatore si tende a fare meno attenzione, ma nei dispositivi mobili bisogna porre molta cura all'utilizzo delle risorse perché con poche mosse sbagliate si può compromettere la velocità dell'applicazione. Per esempio in Java 2 Micro Edition il garbage collector non è efficiente come nella Standard Edition quindi bisogna stare attenti e cercare di mettere sempre a null le variabili non usate in modo che il garbage collector vada a liberare la memoria.

La piattaforma sviluppata si è comportata bene in campo di test, a parte il bug rilevato nelle librerie utilizzate (vedi pag. 12 per approfondimenti) per l'utilizzo del Bluetooth in J2SE, il problema comunque è risolvibile in vari modi: o con un tempo di delay come descritto a pagina 12, oppure usando le librerie Avetana in Windows oppure provando altre librerie simili.

## 6.2 Sviluppi futuri

Di seguito vengono esposti alcuni ampliamenti interessanti per il progetto realizzato.

### 6.2.1 Piattaforma per la distribuzione dell'applicazione

Per l'utilizzo reale di questa applicazione sarebbe buona cosa avere una piattaforma di distribuzione del file jar del Client in modo da fornire a tutti i dispositivi nelle vicinanze del server l'applicazione per l'esplorazione dei contenuti.

### 6.2.2 Arricchimento dei contenuti

Le pagine sono scritte in linguaggio HTML, ad ora vengono riconosciuti i tag html basilari, che garantiscono la possibilità di rappresentare una buona pagina di contenuto, anche perché la grafica nelle MIDlet è limitata. Interessante comunque sarebbe l'ampliamento dei tag HTML riconosciuti in modo da arricchire graficamente il contenuto delle pagine sempre tenendo conto delle limitazioni delle MIDlet e della potenza di calcolo dei dispositivi.

### 6.2.3 Realizzazione Servlet di controllo

Interessante è lo sviluppo di una Servlet che funga da View (di MVC) della parte server, in modo da poter amministrare il server anche da remoto, magari con una successiva integrazione con l'interfaccia web per l'inserimento dei contenuti.

### 6.2.4 Ampliamento per Musei

Un ampliamento interessante sarebbe lo sviluppo della parte dei contenuti per la gestione dei contenuti di un museo, per come è stata costruita l'applicazione è sufficiente realizzare:

- una nuova classe per il server che gestisca le stringhe di richiesta, sulla base di ContentMan
- realizzazione del database necessario per la gestione dei contenuti per un museo e implementazione con qualsiasi DBMS, basta cambiare il driver ODBC nella classe Content

- realizzazione dell'interfaccia Web per l'inserimento dei contenuti nel database con altri linguaggi di scripting Web, per esempio usando Java Server Pages, che affiancato ad una Servlet di controllo (6.2.3), permetterebbe la migrazione ad un unico sistema server basato su Java.



# Bibliografia

- [1] Andrew S. Tanenbaum, *Computer Networks*. Prentice Hall (2003) Cap. 4.
- [2] Kjell Jorgen Hole, *Bluetooth*. Personal Slides (2006).
- [3] *PHP* - Wikipedia (<http://it.wikipedia.org/wiki/Php>). Ultima visita in data 26 maggio 2007
- [4] *kXML* - Sito kXML (<http://kxml.sourceforge.net/about.shtml>). Ultima visita in data 26 maggio 2007
- [5] *MySql* - Wikipedia (<http://it.wikipedia.org/wiki/MySQL>). Ultima visita in data 25 maggio 2007
- [6] *Netbeans* - Sito Netbeans (<http://www.netbeans.org/products/index.html>). Ultima visita in data 25 maggio 2007