Indice

Introduzione

Questa tesi di laurea è stata sviluppata per conto della ditta Software Products Italia di Sesto Fiorentino (Fi). Si tratta della realizzazione di un tool di sincronizzazione tra un database consolidato (server) e un database remoto per palmari, per il controllo del traffico autostradale in ambiente .Net.

La trattazione proseguirà nel seguente modo: dopo una breve introduzione sul progetto e sulle specifiche imposte dal cliente, seguirà una panoramica sull'ambiente di sviluppo che è stato scelto per la realizzazione del progetto (Capitolo 1). Dal secondo capitolo in poi, inizia la descrizione dell'applicativo che è stato realizzato in maniera dettagliata, a partire dall'interfaccia grafica per l'utente. Il terzo capitolo tratta la parte centrale del progetto, ovvero la sincronizzazione tra il palmare e il server; in esso sono anche incluse le descrizioni dei due database (quello sul server e quello sul palmare. In ordine di importanza, gli argomenti successivi alla sincronizzazione col server, sono: la descrizione delle scelte effettuate a fronte dei problemi riscontrati e la gestione degli errori. Per facilitare la comprensione di termini specifici e poco utilizzati nel linguaggio comune è stato realizzato un breve glossario.

I. Inquadramento del progetto

Il progetto in questione consiste nello sviluppo di un applicativo per palmare in grado di mostrare lo stato di una determinata tratta autostradale selezionata dall'utente, visualizzando la cartina in questione e modificandola a seconda delle informazioni ricevute dalla sincronizzazione col server. Quest'ultima operazione è la parte centrale del progetto; inizia quando l'utente seleziona dal menù la voce "Sincronizza" e così facendo aggiorna il database remoto presente sul palmare. L'applicativo che è stato realizzato lavorerà su diversi palmari; il palmare fornito dall'azienda è un Dell Axim X30, dotato di un dispaly touch-screen e un tastierino numerico con alcuni pulsanti, su cui è installato il sistema operativo Microsoft Windows Mobile 2003. Per simulare la sincronizzazione ho inserito alcuni dati in un database ASA residente sul mio computer portatile. Ogni volta che l'utente sincronizza il palmare con tale server, le tabelle di questo database vengono copiate in un database Ultralite rendendo i dati immediatamente disponibili all'applicativo che li elaborerà per visualizzare lo stato della tratta autostradale in questione (la frequenza degli aggiornamenti non è stata resa automatica mediante la funzione Thread, per semplificare la simulazione).

II. Specifiche

Chi ha commissionato il lavoro, ha precisato il linguaggio di programmazione che è stato utilizzato per realizzare il tool di sincronizzazione, per poterne analizzare pro e contro ed effettuare dei confronti in futuro con altri linguaggi (ad esempio Java).
Il committente ha chiesto che:

- venisse impiegato un database remoto e uno consolidato utilizzando gli strumenti forniti da SQL Anywhere 9
- l'applicazione visualizzasse una cartina autostradale e che la modificasse a seconda dello stato autostradale, utilizzando i dati del database remoto
- la sincronizzazione tra server e database remoto potesse essere effettuata in qualunque momento

Le specifiche sopra indicate non sono particolareggiate, ma indicano soltanto le linee guida che sono state seguite durante la realizzazione del progetto.

1. Panoramica sull’ambiente di sviluppo

Questo capitolo si propone di fornire qualche informazione di base sull’ambiente di sviluppo che è stato utilizzato per portare a termine la realizzazione di questo progetto. Le informazioni fornite di seguito sono puramente descrittive; per nozioni più approfondite si rimanda ai rispettivi manuali.

La scelta dell’ambiente di sviluppo è stata condizionata sia da richieste specifiche del committente (valutare pro e contro dell’ambiente di sviluppo, nel realizzare un applicativo per palmari che sincronizzi tramite MobiLink il database consolidato ASA con il database remoto UltraLite), sia da potenzialità che questo possiede. L’ambiente di sviluppo che è stato scelto è Visual Studio .NET 2003, il linguaggio di programmazione con cui è stato sviluppato il software è Visual C# .NET 2003.

Per l’accesso e la gestione dei dati, l’applicativo utilizza UltraLite collegandosi ad un database implementato con Sybase Adaptive Server Anywhere.La tecnologia MobiLink (inclusa in SQL Anywhere Studio) dà la possibilità di sincronizzare dati da DataBase consolidati (come Sybase Adaptive Server Anywhere e Adaptive Server Enterprise, Microsoft SQL Server, Oracle e DB2) con DataBase ASA remoti e applicazioni UltraLite.

Visual Studio .NET è lo strumento Microsoft di seconda generazione per la creazione e la distribuzione di software Microsoft .NET. Visual Studio .NET 2003 include una versione avanzata di Windows .NET Framework, basato sulle precedenti versioni; esso è dotato di nuove funzioni e include miglioramenti sia del software che della documentazione. Grazie al supporto integrato per .NET Compact Framework, Visual Studio .NET 2003 inserisce nell'ambiente .NET i dispositivi portatili e incorporati come Pocket PC, così come altri dispositivi dotati del sistema operativo Microsoft Windows CE .NET. Visual Studio .NET può essere utilizzato per:

- Creazione di applicazioni per Windows.
- Creazione di applicazioni Pocket PC.
- Creazione di applicazioni Web.
- Creazione di applicazioni Web portatili in grado di riconoscere il dispositivo di destinazione.
- Utilizzo dei Web service in tutti i contesti applicativi sopra indicati.
- Eliminazione dei conflitti noti come "inferno delle DLL".
- Eliminazione dei costosi problemi di distribuzione e di manutenzione delle applicazioni.

1.1 Windows .NET Framework

Il C# è un linguaggio che non opera direttamente nell'ambiente standard in cui i programmi sono compilati in codice macchina, ma, come il Java, ha bisogno di un ambiente di esecuzione virtuale. Nel caso di Java, c'è bisogno della famosa Java Virtual Machine che consente ai programmi (che hanno estensione **.**class) di poter girare. Il C# ha invece bisogno dell'istallazione del cosiddetto **.NET Framework** scaricabile dal sito della Microsoft. Questa è la sua definizione tecnica: .NET Framework è una nuova piattaforma per computer che semplifica lo sviluppo delle applicazioni nell'ambiente altamente distribuito di Internet. .NET Framework è progettata per ottenere gli obiettivi indicati di seguito:

- Fornire un ambiente di programmazione orientato agli oggetti coerente, sia che il codice degli oggetti sia memorizzato ed eseguito localmente, eseguito localmente ma distribuito su Internet oppure eseguito in modalità remota.
- Fornire un ambiente di esecuzione del codice che minimizzi la distribuzione del software e i conflitti di versioni.
- Fornire un ambiente di esecuzione del codice che garantisca un'esecuzione sicura anche dei codici creati da produttori sconosciuti o semi-trusted.
- Fornire un ambiente di esecuzione del codice che elimini i problemi di prestazioni degli ambienti basati su script o interpretati.
- Rendere coerente l'esperienza dello sviluppatore attraverso tipi molto vari di applicazioni, quali applicazioni basate su Windows e applicazioni basate sul Web.
- Generare tutte le comunicazioni in base agli standard industriali per assicurare che il codice basato su .NET Framework possa integrarsi con qualsiasi altro codice.

.NET Framework presenta due componenti principali: Common Language Runtime e la libreria di classi .NET Framework. Common Language Runtime rappresenta la base di .NET Framework e può essere considerato come un agente che gestisce il codice in fase di esecuzione, fornendo servizi di base quali gestione della memoria, gestione di thread e servizi remoti, attivando al contempo una rigida indipendenza dei tipi e altre forme di accuratezza del codice che assicurano protezione ed efficienza. Il concetto di gestione del codice è infatti un principio fondamentale di runtime. Il codice destinato al runtime è definito codice gestito, mentre quello non destinato al runtime è definito codice non gestito. La libreria di classi, l'altro componente principale di .NET Framework, è un insieme completo orientato agli oggetti di tipi riutilizzabili che possono essere impiegati nello sviluppo delle applicazioni, da quelle tradizionali della riga di comando o con interfaccia utente grafica (GUI, Graphical User Interface) a quelle basate sulle più recenti innovazioni fornite da ASP.NET, quali Web Form e servizi Web XLM.

1.1.1 Il .NET Compact Framework

Il .NET Compact Framework è studiato per i dispositivi compatti (pocket PC, Smart phone ecc); rappresenta un elemento fondamentale per lo sviluppo di applicazioni mobili, in quanto mette a disposizione funzionalità per dispositivi intelligenti come il modello di programmazione unificato con .NET Framework su desktop e su server, il supporto integrale per gli XML Web service, l'accesso ai dati mediante ADO.NET e XML, e librerie di classi che permettono di realizzare applicazioni sofisticate in tempi ridotti. Questa piattaforma è supportata da Ultralite, il database relazionale che permette di sviluppare rapidamente applicazioni che estendono la gestione e l'analisi dei dati anche ai nuovi dispositivi intelligenti. Microsoft Visual Studio .NET fornisce un insieme omogeneo di strumenti e interfacce per creare applicazioni attraverso le tecnologie mobili di Microsoft. Purtroppo, essendo questa piattaforma destinata a dispositivi portatili (quindi con poca memoria), Microsoft ha dovuto ridurre notevolmente le funzionalità del .NET Framework per arrivare a questo prodotto (che occupa poco meno di 2 MB di memoria su disco), di conseguenza lo sviluppatore ha a sua disposizione un numero decisamente ridotto di metodi e proprietà per ogni classe.

1.2 Il linguaggio C#

C# (C Sharp) è un linguaggio orientato agli oggetti che consente ai programmatori di creare rapidamente una vasta gamma di applicazioni per la nuova piattaforma Microsoft .NET. A parte qualche termine, il C# è un linguaggio adatto anche ai principianti che si affacciano solo ora nel mondo della programmazione (come il sottoscritto) ed è vero che chi ha già avuto esperienze con altri linguaggi, troverà facile comprendere i costrutti del C#. C# rappresenta un'ottima scelta per la creazione di architetture per una vasta gamma di componenti, dagli oggetti business di alto livello ad applicazioni a livello di sistema. Questo linguaggio è progettato per offrire una modalità di sviluppo rapido, senza sacrificare la potenza e il controllo che hanno rappresentato le caratteristiche distintive di C e C++. Proprio grazie a questa eredità, C# presenta un livello di somiglianza molto elevato con C e C++. Il design di questo linguaggio elimina la maggior parte degli errori di programmazione più comuni in C++. Ad esempio:

- Le attività di garbage collection sollevano il programmatore dal compito di dover gestire la memoria manualmente.
- In C# le variabili vengono inizializzate automaticamente dall'ambiente.
- Le variabili sono indipendenti dai tipi.

Inoltre, integra direttamente nel linguaggio il supporto per la gestione delle versioni. Ad esempio, l'override dei metodi deve essere esplicito e non può avvenire inavvertitamente come in C++ o in Java. In questo modo si prevengono gli errori di codifica e si conserva la flessibilità nella gestione delle versioni. Una funzionalità correlata è il supporto nativo per le interfacce e l'ereditarietà delle interfacce. Queste funzionalità consentono di sviluppare e migliorare nel tempo strutture complesse. Infine, C# include il supporto nativo per le API COM (Component Object Model) e basate su Windows® e consente un utilizzo limitato dei puntatori nativi.

1.3 Adaptive Server Anywhere e UltraLite

Indipendentemente dal tipo di connessione o applicazione, SQL Anywhere Studio garantisce un accesso ininterrotto e sempre disponibile ai dati e alle applicazioni aziendali permettendo agli addetti ai lavori di essere produttivi con le informazioni di cui necessitano al momento giusto. L'operazione offline, la pietra angolare dell'accesso ininterrotto e sempre disponibile, è possibile grazie alla sincronizzazione dei dati e alla gestione dei dati locali completamente transazionale. I dipendenti possono accedere alle informazioni ed effettuare transazioni offline, riducendo i costi di comunicazione e aumentando al tempo stesso le prestazioni delle applicazioni e la durata della batteria dei dispositivi. Le tecnologie di sincronizzazione dati permettono lo scambio tempestivo di nuove informazioni e riducono la quantità di dati da inviare durante una connessione wireless o dial-up. SQL Anywhere Studio include una serie completa di tecnologie di sincronizzazione scalabile, bidirezionale delle informazioni aziendali tra i sistemi all'interno dell'impresa e quelli remoti. Le tecnologie di sincronizzazione di SQL Anywhere Studio sono ottimizzate sia per ambienti connessi occasionalmente che per quelli praticamente in tempo reale, compresa la sincronizzazione sul server per trasmettere avvertimenti tempestivi a dispositivi remoti. Queste tecnologie permettono la comunicazione sicura di informazioni per utenti remoti e mobile grazie a un'ampia varietà di protocolli sincroni, asincroni, wireless, dial-up e per Internet.

Caratteristiche Generali
Prestazione e scalabilità: Adaptive Server Anywhere è stato ideato per offrire un'ottima prestazione out-of-the-box. Un query optimizer di grandi prestazioni e capace di self-tuning determina il modo più efficace per accedere alle informazioni, migliorando le prestazioni ed eliminando la necessità di far intervenire un esperto. Il supporto al multiprocessore simmetrico (SMP) e innovativi algoritmi di elaborazione delle query velocizzano le prestazioni per query più complesse, e per un maggior numero di utenti connessi, ottimizzando inoltre l'uso di SQL Anywhere Studio negli ambienti aziendali. L'Index Consultant offre agli amministratori e agli sviluppatori un modo facile per ottimizzare le prestazioni guidandoli nella selezione degli indici più adatti.

Supporto multipiattaforma
SQL Anywhere Studio supporta un'ampia gamma di sistemi operativi, inclusi Windows (32 o 64 bit), Mac OS X, Netware, diverse varianti di UNIX e Linux (32 o 64 bit) e piattaforme portatili molto diffuse, come Microsoft Windows CE e Palm Computing Platform per offrire uno sviluppo estremamente flessibile. I file del database Adaptive Server Anywhere sono compatibili a livello binario, permettendo così agli sviluppatori di copiare semplicemente il database nei molteplici sistemi operativi. Supporto alle Open API e agli strumenti di sviluppo: Adaptive Server Anywhere supporta molti standard per un accesso ai dati facile e di alta qualità compresi ADO.NET, SOAP, XML, OLE DB, ODBC, JDBC , Sybase OpenClient ed Embedded SQL. Con la tecnologia UltraLite , gli sviluppatori possono utilizzare i linguaggi Visual Basic, Java, .NET o JScript per l'accesso ai dati su dispositivi palmari e dispositivi intelligenti. Grazie al fatto che supporta i principali strumenti di sviluppo e ambienti, inclusi Java, Sybase PowerBuilder®, Microsoft Visual Basic,

Visual C++ e Visual Studio .NET, AppForge MobileVB, Symantec Visual Cafe, Borland Delphi e MetrowerksCodeWarrior, SQL Anywhere Studio permette di potenziare gli investimenti esistenti in capacità e strumenti di sviluppo.

Ottimizzato per dispositivi palmari e apparecchi intelligenti
Il database UltraLite di SQL Anywhere Studio offre una soluzione ideale per gli ambienti a memoria limitata, come i dispositivi palmari. UltraLite fornisce capacità aziendale con integrità referenziale ed elaborazione di transazioni. Assicura inoltre l'integrazione con strumenti di sviluppo molto diffusi, come Microsoft Visual Basic, Visual Studio .NET e Borland JBuilder. UltraLite Dynamic SQL facilita lo sviluppo ai programmatori SQL che vi fanno affidamento per creare applicazioni dati dinamiche. Inoltre, UltraLite permette la diffusione di applicazioni Java su qualsiasi piattaforma che supporti Java JDK 1.1.8 e versioni successive o VM compatibile JEODE Personal Java 1.2 su Windows CE.

Sincronizzazione dei dati aziendali
La sincronizzazione di SQL Anywhere Studio è stata ideata in modo che sia facile da utilizzare sia per gli amministratori del database che per gli utenti finali, permettendo così di essere sviluppata e gestita facilmente da un gran numero di utenti. L'amministrazione è notevolmente semplificata visto che tutte le funzioni amministrative vengono eseguite da un database consolidato. Assicura affidabilità grazie all'invio sicuro dei messaggi. Se si perde la connessione alla rete oppure un messaggio viene corrotto durante la trasmissione, i dati rimangono transazionalmente intatti e la sincronizzazione permette di riprendere da dove si è verificata l'interruzione.

Sincronizzazione eterogenea
La tecnologia di sincronizzazione MobiLink di SQL Anywhere Studio assicura uno scambio di informazioni sicuro e bidirezionale tra Adaptive Server Anywhere remoto o il database UltraLite e numerose fonti di dati aziendali incluse Adaptive Server Anywhere, Sybase Adaptive Server Enterprise, Oracle, Microsoft SQL Server e IBM DB2. I sistemi remoti si connettono con protocolli Internet standard al server di sincronizzazione MobiLink, che comunica con il database back-end. è possibile utilizzare la sicura tecnologia opzionale di crittografia a 128-bit per garantire dati totalmente protetti durante la trasmissione.

Protocolli specializzati, incluso il wireless
La tecnologia di sincronizzazione presente in SQL Anywhere Studio supporta la sincronizzazione wireless sulla rete LAN, nonché su quella pubblica garantendo così un accesso immediato alle informazioni aziendali. Per quanto riguarda i palmari,la sincronizzazione MobiLink supporta protocolli specializzati, tra i quali Palm Computing HotSync e Microsoft ActiveSync. MobiLink offre anche la possibilità di trasferire importanti dati a un dispositivo remoto mediante messaggi SMS o UDP.

2. L’interfaccia grafica

L’interfaccia grafica si presenta semplice e di facile utilizzo. Il programma presenta soltanto quattro form (ognuno con diverse casistiche): un form principale, presente all’avvio e quando viene visualizzata la mappa, un form per la sincronizzazione, uno per selezionare da un elenco le cartine autostradali e infine, un form che mostra la legenda.

2.1 Il Form principale

Considerando le forti limitazioni del .NET Compact Framework (dovute a problemi di memoria, come meglio specificato nel paragrafo 1.1.1), si può affermare che le scelte, per la maggior parte degli oggetti C# e dei metodi utilizzati dal tool, siano state obbligate. Il form principale è costituito da un menù e da due scrollbar (orizzontale e verticale) per scorrere le cartine autostradali, una volta visualizzate all’interno di una picturebox. In figura 1, è mostrata la schermata iniziale, dove sono visibili menù e scrollbar, mentre la picturebox non si vede perché è soltanto una sorta di “contenitore” per le immagini. Nel menù abbiamo varie voci che costituiscono l’elenco delle funzionalità dell’applicativo. Ognuna di queste consente di accedere ai form secondari, a cui è stato accennato in precedenza. Da questo menù è infatti possibile aprire un form secondario (“show dialog”) sia per la sincronizzazione, sia per la selezione di una determinata mappa, che per la visualizzazione della legenda.
Il codice C# è disponibile nell’ Appendice A.

Figura 1 – Schermata iniziale.

2.2 Il Form di sincronizzazione

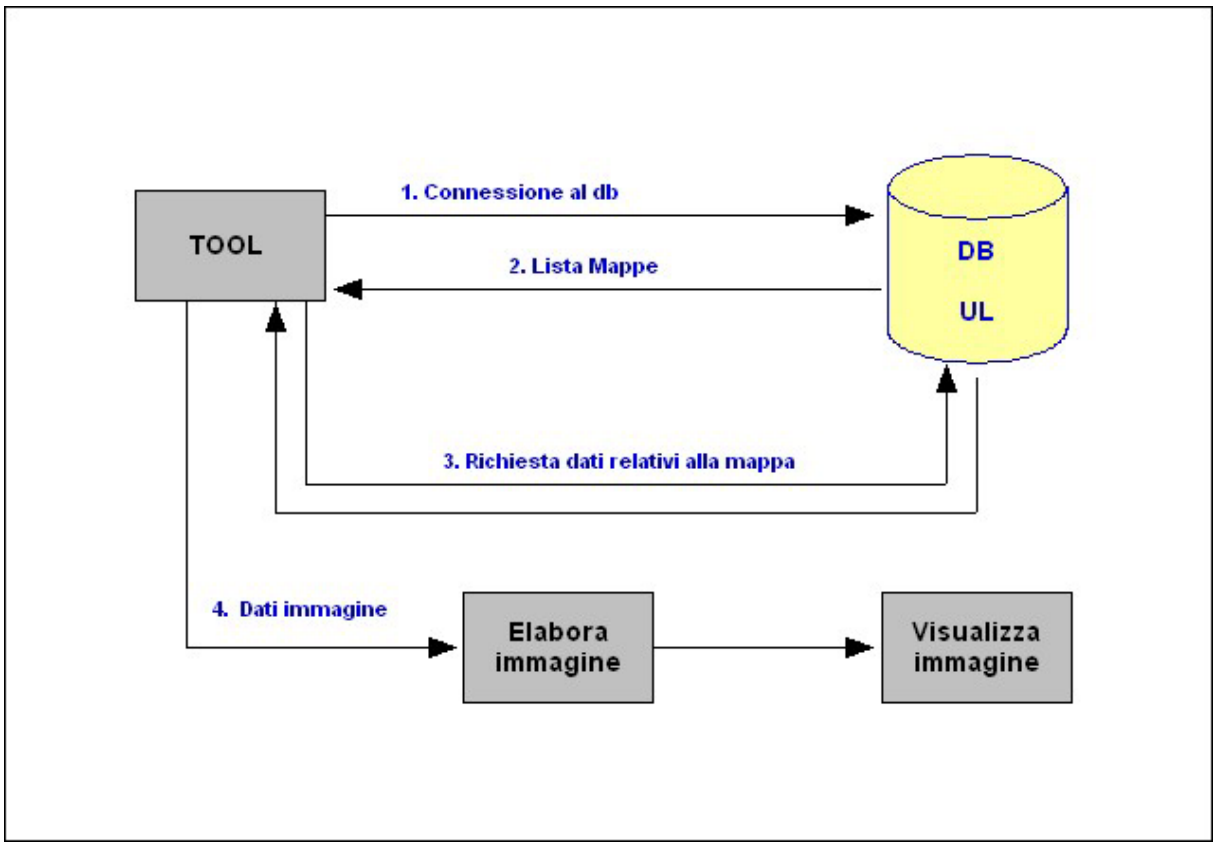
Ogni volta che viene selezionata dal menù la voce “Sincronizza”, il programma apre una show dialog cioè un secondo form, chiamato “DotNetSynchronization” (figura 2). Anche questo form è molto semplice, infatti è costituito solamente da due textbox e un pulsante. La prima textbox consente di impostare l’indirizzo IP del dispositivo che ospita il database consolidato, con il quale si vuole effettuare la sincronizzazione (per questa simulazione il mio portatile). Il pulsante consente di avviare il processo di sincronizzazione; la seconda textbox invece, mostra le fasi di quest’ultima e segnala se il database è stato sincronizzato o meno. La textbox mostra infatti il messaggio “Il database è vuoto, sincronizzare” se il programma ha rilevato che le tabelle del database non contengono nulla, diversamente si potrà leggere il messaggio: “Il database è sincronizzato”. Vedere l’ Appendice B, per visionare il codice.

Figura 2 – Form di sincronizzazione

Figura 3 – Form di selezione delle Mappe

2.3 Il Form di selezione delle mappe

La figura 3 in alto a destra, mostra la seconda show dialog che il programma apre se si sceglie dal menù la voce “Seleziona Mappa”. Si nota subito una listbox, la cui funzione è quella di elencare la lista delle mappe disponibili nel database remoto del palmare, che è stato appena sincronizzato con quello consolidato. Selezionata una mappa della quale si vuole conoscere lo situazione autostradale e premuto il tasto “Ok”, si consente al tool di elaborare l’immagine di questa cartina mediante i dati contenuti nel database. Queste informazioni identificano un particolare evento che si è manifestato lungo una determinata tratta, e il programma li interpreta colorando quel tratto di un preciso colore, il cui significato è illustrato nella legenda. La mappa una volta elaborata, viene visualizzata nella picturebox del form principale (figura 4). Ovviamente nella realtà, dovrà esistere un servizio che aggiorni questi dati nel database consolidato e il programma dovrà sincronizzarsi (grazie ai thread) più frequentemente e in modo automatico. Il codice di questo form è disponibile nell’ Appendice C; il codice che consente al tool, di reperire le informazioni per visualizzare ed elaborare l’immagine connettendosi al database remoto, fa parte invece, della classe “ConnessioneDatabase.cs”, consultabile nell’ Appendice E.



2.4 Il Form della legenda

Questo form (figura 5) ha la sola funzione di mostrare un'immagine che rappresenta la legenda, pertanto è stato molto semplice realizzarlo. Ogni colore identifica un particolare evento o una particolare situazione meteorologica che si può verificare sulle singole tratte autostradali. Il relativo codice è contenuto nell' Appendice D.

Figura 4 - Mappa elaborata

Figura 5 - Form della legenda

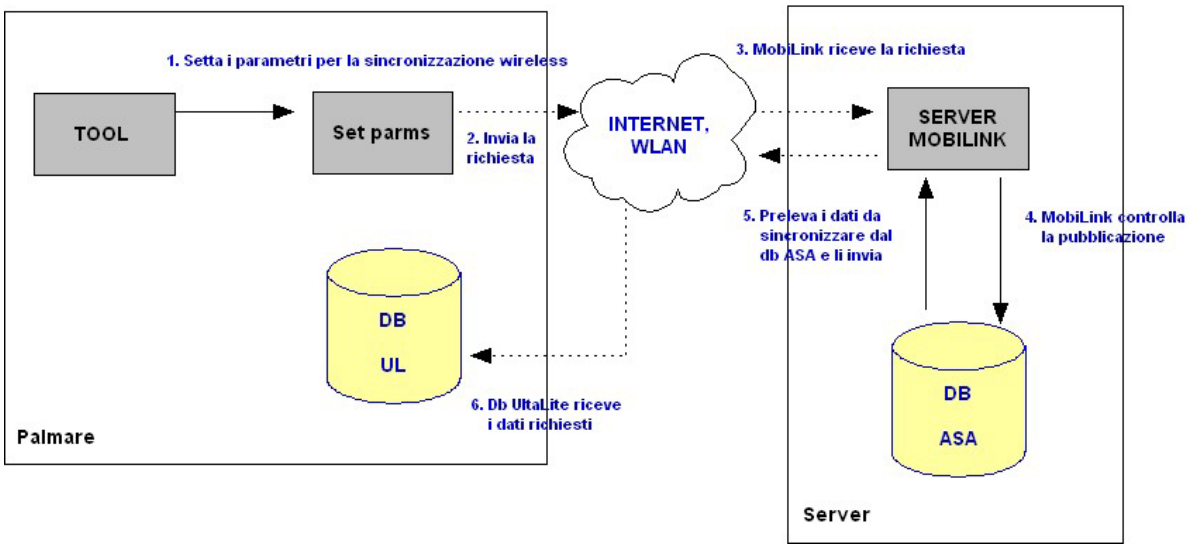
3. La sincronizzazione tra il palmare e il server

La sincronizzazione tra il palmare ed il server è la parte cruciale di questo progetto. La simulazione è stata effettuata facendo interagire via wireless (con una connessione “ad-hoc”) il database remoto UltraLite del palmare, con il database server residente sul mio notebook. Questa sincronizzazione è per il momento unilaterale, ma in futuro verrà sviluppata una nuova funzionalità che permetterà all’utente in viaggio di segnalare, il verificarsi di particolari eventi e di inviare questi dati dal palmare al server.

3.1 La connessione tra il server e il palmare

In questa simulazione l’utente avvia la sincronizzazione, semplicemente digitando nell’apposita textbox l’indirizzo IP del dispositivo che ospita il database consolidato e premendo il tasto “Sync”; nella realtà invece, le sincronizzazioni saranno automatiche e avverranno dopo determinati intervalli di tempo per poter utilizzare dati sempre aggiornati. In entrambi i casi, le operazioni da eseguire sono le medesime, cambia soltanto la loro implementazione. Ogni volta che si richiede una sincronizzazione, l’applicativo:

- Se non esiste, crea il database remoto “.udb” utilizzando il file “.usm” che ne rappresenta lo schema
- Si connette via wireless al dispositivo contenente il database consolidato “.db”, tramite indirizzo IP
- Popola il database remoto inserendo i dati provenienti dal database consolidato



3.2 Il database server e il MobiLink synchronization server

Per realizzare questa simulazione, innanzitutto ho creato due file ".bat" (build.bat e step1.bat) eseguibili da MS-DOS per poter eseguire comodamente ogni tentativo di sincronizzazione. Questo perché la sincronizzazione necessita sempre la prima volta, dell'esecuzione di alcuni comandi di SQL Anywhere 9. Bisogna in pratica, predisporre l'ambiente in cui si sta sperimentando. Per prima cosa bisogna definire un ODBC data source, attraverso cui si possa effettuare la connessione al nostro database ASA. La riga di comando è la seguente:

```
dbdsn" -q -w SmartDeviceApplication1 -y -c "uid=DBA;pwd=SQL;
dbf=SmartDeviceApplication1.db;eng=SmartDeviceApplication1"
```

Una volta definito il data source (a cui è stato dato lo stesso nome del nostro applicativo per comodità) si può procedere alla creazione del database ASA che lo utilizza, con il comando:

```
dbinit" -q SmartDeviceApplication1.db
```

Ora si può pertanto popolare questo database con le tabelle e con qualche dato necessario alla simulazione. Ciò avviene mediante un altra riga di comando, che esegue uno script SQL (build.sql) precedentemente creato che contiene le istruzioni per:

1. la creazione di tutte le tabelle del database
2. l'inserimento di dati di prova in queste tabelle
3. la definizione di quali verranno sincronizzate mediante opportuno script di sincronizzazione.

```
dbisql" -q -c "dsn=SmartDeviceApplication1" build.sql
```

Analizziamo passo per passo i tre punti chiave dello script Sql appena elencati, riportandone il codice. Avremo così modo di capire la struttura dei database utilizzati. Inizialmente vengono definite e create le varie tabelle con l'apposito comando SQL "create table".

```
-- SQL script to create SmartDeviceApplication1 database and fill it with data.

--message '>>Creating tables.' type info to client
--go

create table Colore (codice char ( 1 )primary key,
                     valore char ( 6 ))
go

create table Mappa (codMappa char ( 2 )primary key,
                    descrizione varchar ( 15 ),
                    percorsoFile  long varchar )
go

create table Rete (codRete int primary key,
                   nomeRete varchar ( 50 ))
go

create table TipoEvento (codTipo char ( 3 ) primary key,
                         descrizione varchar ( 50 ),
                         nomeIcona varchar( 255 ))
go
```

```
create table TipoTraffico (codice int  primary key,
                           codiceColore char ( 1 ),
                           descrizione varchar( 20 ))
go

create table TrafficoRete (codTrafficoRete int  primary key,
                           codRete int ,
                           dataOra datetime ,
                           codStazE int ,
                           codStazU int ,
                           tipoTraffico int ,
                           notizie varchar ( 255 ))
go

create table Tratta (codStazE int,
                     codStazU int,
                     codMappa char ( 2 ),
                     nomeTratta varchar ( 100 ),
                     codReteE  int,
                     codReteU  int,
                     x1  int,
                     y1  int,
                     x2  int,
                     y2  int,
                     primary key (codStazE, codStazU))
go
```

Descriviamo ora brevemente, le tabelle del database appena create; la tabella “Colore” contiene un elenco dei colori utilizzati dal programma per evidenziare i vari tratti autostradali delle cartine, soggetti ad un particolare evento. La tabella “Mappa” riporta il nome della cartina autostradale e il nome del file a cui è associata la sua immagine, mentre la tabella “Rete” contiene i nomi delle autostrade. In “TipoEvento” sono inserite tutte le possibili situazioni che si possono verificare lungo una tratta autostradale, mentre “TipoTraffico” le associa ad un preciso colore. La tabella “TrafficoRete” contiene l’elenco vero e proprio dei tratti autostradali soggetti ad un evento. Infine, la tabella “Tratta” consente al programma di tracciare i tratti colorati, poichè memorizza per ogni singola tratta, le coordinate sulla cartina che la delimitano. Ora con il comando SQL “insert into” le tabelle vengono popolate con qualche dato.

```
--message '>>Adding data' type info to client
--go

-- add data to Colore table
insert into Colore ( codice, valore ) values ( 'R', 'FF0000' );
insert into Colore ( codice, valore ) values ( 'G', '00FF00' );
insert into Colore ( codice, valore ) values ( 'B', '00FFFF' );
insert into Colore ( codice, valore ) values ( 'Y', 'FF9800' );
insert into Colore ( codice, valore ) values ( 'N', '000000' );
insert into Colore ( codice, valore ) values ( 'W', 'FFFFFF' );
insert into Colore ( codice, valore ) values ( 'M', 'A36917' );
insert into Colore ( codice, valore ) values ( 'O', 'D58C24' );
insert into Colore ( codice, valore ) values ( 'A', '24D5D3' );
commit
go


-- add data to Mappa table
insert into Mappa ( codMappa, descrizione, percorsoFile ) values ( 1, 'firenze', 'dt4.png' );
insert into Mappa ( codMappa, descrizione, percorsoFile ) values ( 2, 'bologna', 'dt3.png' );
commit
go
```

```
-- add data to Rete table
insert into Rete  ( codRete, nomeRete ) values ( 1, 'a1' );
insert into Rete  ( codRete, nomeRete ) values ( 2, 'a11' );
insert into Rete  ( codRete, nomeRete ) values ( 3, 'a13' );
insert into Rete  ( codRete, nomeRete ) values ( 4, 'a14' );
commit
go

-- add data to TipoEvento table
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'COD', 'Coda', null );
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'TRA', 'Traffico intenso', null );
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'NEV', 'Neve', null );
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'CHI', 'Tratto chiuso', null );
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'NEB', 'Nebbia', null );
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'CAN', 'Cantiere', null );
insert into TipoEvento ( codTipo, descrizione, nomeIcona ) values ( 'GHI', 'Ghiaccio', null );
commit
go

-- add data to TipoTraffico table
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 0, 'B','Non definito' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 1, 'G','Regolare' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 2, 'Y','Traffico intenso' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 3, 'R','Coda' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 4, 'W','Neve' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 5, 'A','Ghiaccio' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 6, 'M','Nebbia' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 7, 'N','Tratto chiuso' );
insert into TipoTraffico ( codice, codiceColore, descrizione ) values ( 8, 'O','Cantiere' );
commit
go


-- add data to TrafficoRete table
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 1, 2,
null, 336, 337, 2, 'intenso', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 2, 2,
null, 337, 336, 3, 'code', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 3, 2,
null, 338, 337, 8, 'cantiere', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 4, 2,
null, 334, 335, 4, 'neve', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 5, 1,
null, 10,   11, 2, 'intenso', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 6, 1,
null, 11,   10, 3, 'code', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 7, 2,
null, 335, 336, 5, 'ghiaccio', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 8, 2,
null, 339, 338, 5, 'ghiaccio', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 9, 2,
null, 336, 335, 0, 'non disponibile', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 10,2,
null, 337, 338, 7, 'chiuso', );
insert into TrafficoRete( codTrafficoRete, codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie ) values ( 11,2,
null, 338, 339, 6, 'nebbia', );
commit
go

-- add data to Tratta table
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values (  10,
11, 2, 'modena nord - modena sud', 1, 1, 129, 228, 155, 271 );
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values (  11,
10, 2, 'modena sud - modena nord', 1, 1, 159, 269, 133, 224 );
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 334,
335, 1, 'prato est - prato ovest',  2, 2, 276, 309, 244, 290 );
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 335,
334, 1, 'prato ovest - prato est',  2, 2, 243, 295, 273, 314 );
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 335,
336, 1, 'prato ovest - pistoia' ,   2, 2, 244, 290, 201, 283);
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 336,
335, 1, 'pistoia - prato ovest',    2, 2, 201, 288, 242, 295 );
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 336,
337, 1, 'pistoia - montecatini',    2, 2, 166, 293, 201, 283 );
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 337,
336, 1, 'montecatini - pistoia',    2, 2, 167, 298, 200, 288 );
```

```
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 337,
338, 1, 'montecatini - chiesina uzzanese',2,2,146,316,164,296);
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 338,
337, 1, 'chiesina uzzanese - montecatini',2,2,149,321,167,301);
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 338,
339, 1, 'chiesina uzzanese - altopascio', 2,2,134,324,145,314);
insert into Tratta ( codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2) values ( 339,
338, 1, 'altopascio - chiesina uzzanese', 2,2,137,328,149,319);
commit
go
```

Ora il database è stato popolato; restano da definire le tabelle di cui si sincronizzeranno i dati. Il seguente codice utilizza la stored procedure “ml_add_table_script” per aggiungere degli Sql table scripts al database consolidato. In particolare viene associato il “download_cursor” event, che serve per specificare su quali righe della tabella deve essere effettuato il download nel database remoto.

```
------------------------------------------------
-- Create synchronization scripts
------------------------------------------------

-- Snapshot synchronization -- download all rows


call ml_add_table_script( 'SmartDeviceApplication1', 'Colore', 'download_cursor',
   'SELECT codice, valore FROM Colore' )
go

call ml_add_table_script( 'SmartDeviceApplication1', 'Tratta', 'download_cursor',
   'SELECT  codStazE, codStazU, codMappa, nomeTratta, codReteE, codReteU, x1, y1, x2, y2  FROM Tratta' )
go

call ml_add_table_script( 'SmartDeviceApplication1', 'Mappa', 'download_cursor',
   'SELECT codMappa, descrizione, percorsoFile  FROM Mappa' )
go

call ml_add_table_script( 'SmartDeviceApplication1', 'Rete', 'download_cursor',
   'SELECT codRete, nomeRete FROM Rete' )
go

call ml_add_table_script( 'SmartDeviceApplication1', 'TipoEvento', 'download_cursor',
   'SELECT codTipo, descrizione, nomeIcona   FROM TipoEvento' )
go

call ml_add_table_script( 'SmartDeviceApplication1', 'TipoTraffico', 'download_cursor',
   'SELECT codice , codiceColore,  descrizione  FROM TipoTraffico' )
go

call ml_add_table_script( 'SmartDeviceApplication1', 'TrafficoRete', 'download_cursor',
   'SELECT codTrafficoRete , codRete, dataOra, codStazE, codStazU, tipoTraffico, notizie FROM TrafficoRete' )
go
```

Infine viene creato un MobiLink user, grazie ad un altro comando che stabilisce quali utenti si possono connettere e sincronizzare al database consolidato. Il tutto digitando ed eseguendo:

```
dbmluser" -c "dsn=SmartDeviceApplication1" -u SmartDeviceApplication1 -p
SmartDeviceApplication1
```

Con Step1.bat viene attivato il MobiLink synchronization server sul database appena creato, che a sua volta deve essere in esecuzione. A questo server, il palmare si collegherà tramite indirizzo IP.

```
dbmlsrv9 -c "dsn=SmartDeviceApplication1" -o mlserver.mls -v+ -dl -za -zu+
```

Si precisa che la sincronizzazione, in upload o in download, è definita dal codice C# dell'applicativo e che non è necessario attivare in questo caso, un MobiLink synchronization client. Al termine dell'esecuzione dei due file ".bat" si avrà come risultato la schermata di figura 6.

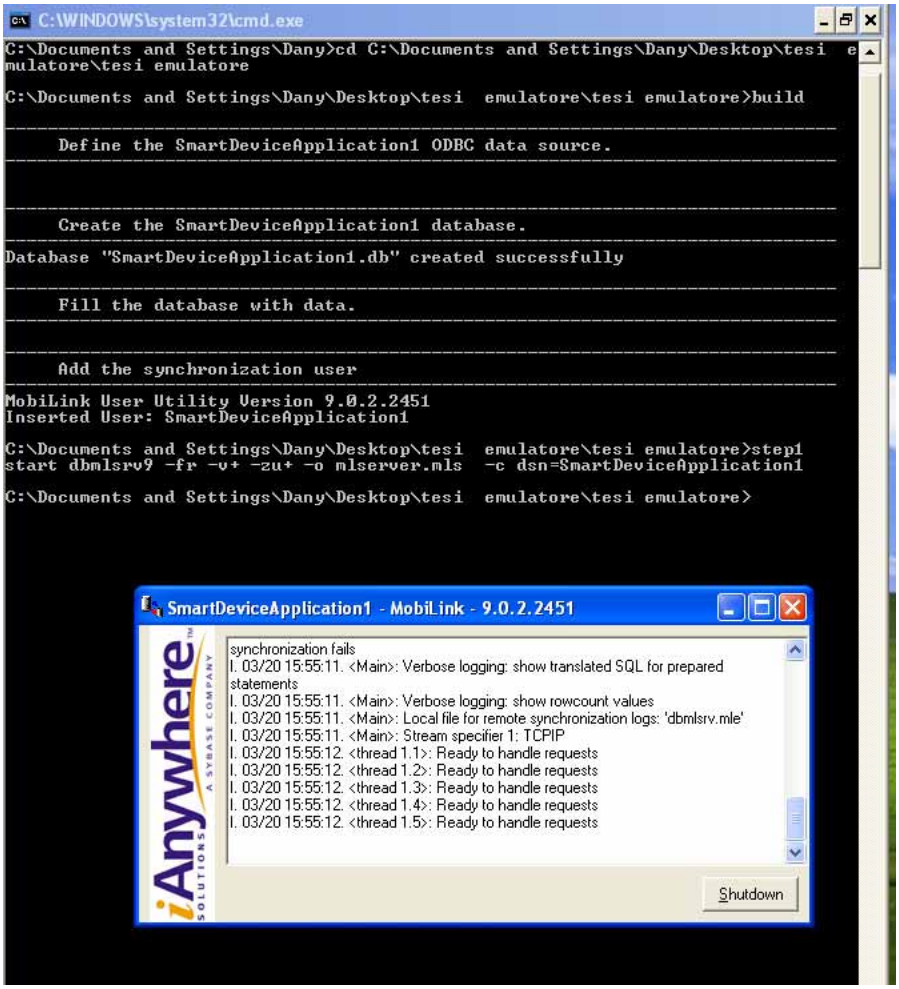


Figura 6 – Build.bat e step1.bat

3.3 Il database UltraLite sul palmare e la sincronizzazione

Nel pacchetto di file che costituisce l'applicativo e che viene trasferito durante il deploy sul palmare, figura anche un file ".usm" . Rappresenta lo schema del database, cioè la sua descrizione e definisce tabelle, indici, chiavi e pubblicazioni presenti al suo interno. Uno schema del database può essere creato utilizzando l' UltraLite Schema Painter o mediante l'utility "ulinit". Quest'ultima crea lo schema sfruttando un Adaptive Server Anywhere database già esistente, nel nostro caso il database consolidato precedentemente definito. Ovviamente avremo un database remoto costituito dalle stesse tabelle di quello consolidato anche se, sempre per motivi di spazio, non sarà completo quanto un database normale (ad esempio non ci sono viste, ruoli ecc), ma tutte le funzioni di base sono implementate.

```
ulinit -c "uid=dba;pwd=sql" -f SmartDeviceApplication1.usm -n TestPublication
```

Quando si cerca di sincronizzare il database remoto UltraLite del palmare con quello consolidato, il programma controlla l'esistenza del primo; in caso negativo procede con la creazione dello stesso, utilizzando appunto il file ".usm". Analizzando infatti la parte più significativa del codice C#, possiamo trovare la definizione di un oggetto della libreria di UltraLite chiamato "ULconnectionParms". La sua funzione è quella di memorizzare il path del database ".udb", ma anche del file ."usm", che sappiamo essere indispensabile per generare il database nel caso in cui non esistesse.

```
ulConnectionParms1 = new iAnywhere.Data.UltraLite.ULConnectionParms();
ulConnectionParms1.DatabaseOnCE =
"\\ProgramFiles\\SmartDeviceApplication1\\SmartDeviceApplication1.udb";
ulConnectionParms1.SchemaOnCE =
"\\ProgramFiles\\SmartDeviceApplication1\\SmartDeviceApplication1.usm";
```

Il metodo "btnSyncNoDialog_Click" attivato dal pulsante "SyncNoDailog", ha la funzione di settare i vari parametri per la connessione al database consolidato. Infatti oltre all' indirizzo IP, vengono impostati anche username e password del MobiLink user autorizzato ad effettuare la sincronizzazione. Queste parametri sono memorizzati grazie all' oggetto "_syncParms". Questo parte di codice è disponibile e consultabile integralmente nell' Appendice B.

```
private void btnSyncNoDialog_Click(object sender, System.EventArgs e)
                {
                  try
                        {
                          textBox1.Text = "Starting sync...";
                          this.btnSyncNoDialog.Enabled = false;
                          _syncParms.Stream = ULStreamType.TCPIP;
                          _syncParms.UserName = "SmartDeviceApplication1";
                          _syncParms.Password = "SmartDeviceApplication1";
                          _syncParms.Version = "SmartDeviceApplication1";
                          _syncParms.StreamParms = "host=" + txtHost.Text;
                          _syncParms.KeepPartialDownload = true;
                          if( _syncResult.PartialDownloadRetained )
                                {
                                  _syncParms.ResumePartialDownload = true;
                                }
                          else
                                {
                                  _syncParms.ResumePartialDownload = false;
                                }

                                _sp = new SyncProgressDialog( _conn, this );
                                _sp.Sync();

                                CheckRows();
                        }
                   catch ( System.Exception t )
                        {
                          MessageBox.Show( "Exception: " + t.Message);
                        }
                   finally
                        {
                          btnSyncNoDialog.Enabled = true;
                        }
                }
```

Il seguente metodo, ”DotNetSynchronization_Load”, si connette al database del palmare e se non esiste lo crea. Infatti l’oggetto ULconnectionParms1 riporta sia il path del database remoto, sia il suo schema.

```
                private void DotNetSynchronization_Load(object sender, System.EventArgs e)
                {
                        // Connect to the database
                        _conn = new ULConnection (ulConnectionParms1.ToString());
                        _syncResult = _conn.SyncResult;
                        _syncParms = _conn.SyncParms;
                        try
                        {
                                _conn.OpenWithCreate();
                        }
                        catch ( System.Exception ex )
                        {
                                MessageBox.Show( ex.Message );
                        }
                        _conn.DatabaseID = 1;

                        CheckRows();

                }
```

A questo punto la sincronizzazione è stata impostata correttamente e procedendo con la simulazione della stessa, si è riscontrato un esito positivo.

4. Problemi riscontrati e gestione degli errori più comuni

In questo capitolo saranno descritti alcuni problemi che sono stati riscontrati durante la realizzazione di questo applicativo e gli errori che si sono manifestati durante il suo utilizzo. Sottolineo che nonostante la versione del tool sia solo di prova, è comunque stata raggiunta affrontando la maggior parte delle problematiche. In questa sezione sarà possibile vedere com'è stata modificata l'implementazione e le scelte effettuate a fronte dei problemi riscontrati. Per quanto riguarda gli errori in cui si può incorrere, si può anticipare che i principali sono di due tipi: errori generati dall'utente ed errori di sistema. I primi sono quasi sempre da attribuirsi ad un cattivo utilizzo dell'applicativo, mentre i secondi possono essere generati in diversi modi.

4.1 Problemi dovuti al Compact Framework

Come già detto più volte, la piattaforma utilizzata ha dei limiti dovuti alle dimensioni della memoria presente sul palmare. Per questo motivo non è possibile sfruttare tutte le librerie e le potenzialità del linguaggio di programmazione ed è necessario fare uso di espedienti per risolvere alcune tipologie di errori. Infatti lo sviluppo dell'applicativo, ha richiesto più tempo del previsto; se si fosse trattato di realizzare un semplice windows form (un' applicazione per windows), le potenzialità di Visual Studio .Net lo avrebbero consentito in pochissimo tempo. Ad ogni modo, una strada per realizzare un obiettivo nella programmazione di applicazioni per palmari, esiste quasi sempre.

4.1.1 Problemi riscontrati nella scelta del Database Remoto

Il problema principale è stato quello relativo alla scelta del database remoto. Poiché la guida di SQL Anywhere 9, non presenta alcun esempio su questo tipo di sincronizzazione, è stato difficile capire quale database si dovesse installare sul palmare. Inizialmente si è cercato di sincronizzare due database ASA. Una scelta non assurda, dal momento che esiste SQL Anywhere per Windows CE, ovvero per i palmari. Infatti una volta installato SQL Anywhere sul dispositivo, il funzionamento dell'applicazione è regolare con questo tipo di database. Inoltre i comandi (AsaCommand, AsaDataReader, ecc..) per operare con il database ASA, sono simili a quelli di ADO .Net, pertanto facili da utilizzare e ampiamente illustrati. Il problema è sorto quando si è dovuta realizzare la funzione di sincronizzazione. Infatti questi database hanno bisogno del MobiLink Synchronization Client, che risiede nel file dbmlsync. Questo file viene sì, installato con SQL Anywhere for Windows CE sul palmare, ma non è possibile impostare i parametri di connessione di quest'ultimo ed eseguirlo da codice. La conclusione è stata raggiunta quando, seguendo un esempio sulla guida di Adaptive Server Anywhere, ci si è accorti che l'oggetto C# che utilizzava il file dbmlsync, non era disponibile per le applicazioni di tipo Smart Device (palmari, smartphone, ecc..). A questo punto dopo essermi documentato attraverso

le fonti più disparate, ho concluso che si doveva utilizzare un altro database: UltraLite. Ciò ha comportato una radicale ristrutturazione dell'applicativo e problemi non indifferenti, dal momento che tutti i comandi che effettuano l'accesso al database, sono piuttosto diversi e poco documentati. UltraLite è molto ridotto di dimensioni ed è adatto ai dispositivi mobili; infatti non necessita nemmeno del deploy di SQL Anywhere per Windows CE e del MobiLink Synchronization Client, ma le sue forti limitazioni hanno causato qualche altro problema. Ad esempio, quando si è trattato di eseguire sulle tabelle di questo database una banalissima query, è stato complicato capire cosa non funzionasse. La query paradossalmente funzionava con il database che si era installato precedentemente e con Interactive SQL, ma non con UltraLite. Solo dopo aver appreso dell'esistenza del "Query Optimizer" di SQL Anywhere (nascosto bene oltretutto) che riproduce le limitazioni del database, si è potuto risolvere il problema.

4.1.2 Problemi con la connessione wireless

L'applicativo nella realtà, deve poter funzionare via wireless: ad esempio mentre si sta viaggiando a bordo della propria autovettura, lungo una certa autostrada. Pertanto si è voluto simulare la sincronizzazione wireless tra palmare e notebook. Il tutto è stato realizzato instaurando una connessione denominata "ad-hoc" tra i due dispositivi. Ciò non è stato semplice per via della scarsa documentazione sia del palmare sia di windows; anche su internet non è stato trovato molto, probabilmente a causa della poca importanza dell'argomento.

4.2 Errori generati dall'utente

Trattiamo ora la parte relativa agli errori. Dal momento che l'applicazione risulta piuttosto semplice e poichè non si deve impostare praticamente nulla, l'unico errore che l'utente può causare è quello dovuto all' inserimento di un errato indirizzo IP. Comunque questo può verificarsi solo in questa simulazione; nella realtà le connessioni al server saranno automatiche o guidate.

4.2.1 Casi di errore sul palmare

Nonostante si sia cercato di prevedere tutti gli errori possibili, potrebbe essere che, compiendo determinate azioni oppure al verificarsi di determinate circostanze, venga generato un errore che non era stato contemplato né in fase di progettazione, né in fase di sviluppo. Per evitare crash dell'applicazione, la maggior parte dei metodi è stato inclusa in un blocco try-catch che cattura qualsiasi eccezione; con questo tipo d'implementazione, però, non è immediatamente identificabile di quale tipo di errore si tratti , pertanto è stato necessario includere le istruzioni che potrebbero generare errori prevedibili in blocchi try-catch specifici. Ogni volta che un'eccezione viene catturata, l'applicativo genera un messaggio di errore (soltanto se necessario) e lancia l'eccezione catturata al chiamante (qualora esista).

4.2.2 Errori di sistema

Gli errori di sistema sono meno prevedibili di quelli generati dall'utente e, soprattutto, è più difficile evitarli effettuando dei controlli simili a quelli descritti nel paragrafo precedente. Essi possono dipendere da errori del sistema operativo (crash), da errori di configurazione o da errori di rete (server spento o guasti nella rete). Anche in questo caso il palmare visualizza un breve messaggio di errore o si richiede una nuova sincronizzazione.

4.2.3 Risoluzione automatica degli errori

La risoluzione automatica degli errori è uno degli argomenti su cui sono focalizzati i prossimi sviluppi del progetto. Attualmente, infatti, non si può parlare di una vera e propria risoluzione automatica degli errori, ma piuttosto di riconoscimento dell'errore. Questo è comunque molto importante, nel senso che, riconoscendo il tipo di errore accaduto, il software può comportarsi in maniera differente e mostrare diversi tipi di messaggio all'utente, in modo che, qualora quest'ultimo non riesca a risolverlo, il responsabile tecnico della ditta committente possa individuare agilmente l'origine del problema.

5. Conclusioni

L’applicativo che è stato realizzato per lo sviluppo di questa tesi di laurea, permette agli utenti che lo utilizzeranno, di rendere più agevoli i loro spostamenti lungo le autostrade. Infatti, essi possono conoscere le informazioni legate alla situazione delle tratte autostradali, premendo un solo tasto. Inoltre questo tool permetterà in futuro, con lo sviluppo di una funzione aggiuntiva, di migliorare il servizio, permettendo allo stesso utente di segnalare un evento lungo la tratta autostradale che sta percorrendo.
Personalmente ritengo la realizzazione di questo progetto particolarmente formativa, in quanto mi ha permesso di sviluppare conoscenze con tecnologie di ultima generazione e in fase di grande sviluppo (i palmari), ma ancora poco conosciute nell’ambito della programmazione e dello scambio di dati (sincronizzazione).
Per di più ho avuto modo di estendere le mie conoscenze informatiche, apprendendo un uovo linguaggio di programmazione (C#) e l’utilizzo di un ambiente di sviluppo piuttosto diffuso presso le aziende, quale Microsoft Visual Studio .NET; Infine ritengo che, grazie allo sviluppo di questo progetto, ho potuto effettuare un’esperienza,che mi permetterà di inserirmi meglio nel mondo del lavoro.

6. Bibliografia

Per la stesura di questa tesi di laurea sono stati utilizzati soprattutto e-books e documentazione
on-line, elencati di seguito assieme ai libri consultati.

Libri _

[1] D. Beneventano, S. Bergamaschi, M. Vincini: Progetto di Basi di Dati Relazionali: lezioni ed esercizi , Pitagora editrice – Bologna

[2] Jeff Prosise: Programmare Microsoft .Net, Mondadori Informatica

e-books _

[3] SQL Anywhere Studio 9.0.1: Ultralite.Net User's guide

[4] SQL Anywhere Studio 9.0.1: Asa Sql User's guide

[5] SQL Anywhere Studio 9.0.1: Asa Sql Reference

[6] SQL Anywhere Studio 9.0.1: UltraLite Database User's guide

[7] SQL Anywhere Studio 9.0.1: Mobilink Synchronization User's guide

[8] SQL Anywhere Studio 9.0.1: Mobilink Synchronization Reference

Documentazione on-line_

[9] Welcome to the .NET Compact Framework QuickStart Tutorial; disponibile su: http://samples.gotdotnet.com/quickstart/CompactFramework/

[10] Microsoft .NET Compact Framework Technical Articles; disponibile su: http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnnetcomp/html/CompactfxTechArt.asp

[11] Developing and Deploying Pocket PC Setup Applications; disponibile su: http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnnetcomp/html/netcfdeployment.asp

[12] Remote ActiveSync FAQ; disponibile su: http://www.cewindows.net/wce/20/ras.htm

[13] Database programming using ADO .NET with C#; disponibile su: http://www.c-sharpcorner.com/Database.asp

[14] Free source code, disponibile su: http://www.codeproject.com/

[15]Connessione wireless ad Hoc, disponibile su:http://www.hwupgrade.it/forum/printthread.php?t=945276

[16] iAnywhere Solutions – Downloads, disponibile su: http://www.ianywhere.com/downloads/index.html

[17] sybase.public.sqlanywhere.ultralite forum, disponibile su: http://www.dbtalk.net/sybase-public-sqlanywhere-ultralite/

[18] .net TUTORIAL, disponibile su http://www.deathlord.it/pro/sl/ricerca/tutorial.net/contents/3_CSharp/14-thread.htm

[19] Mappe autostradali, disponibili su: http://www.autostrade.it/

[20] Creazione di linee e forme con GDI+, disponibile su: http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/vbcon/html/vbtskdrawinglinesshapeswithgdi.asp

[21] Scrolling Form Contents, disponibile su: http://samples.gotdotnet.com/quickstart/CompactFramework/doc/scrolling.aspx

[22] Oggetti Pen, Brush e Color, disponibile su: http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/vbcon/html/vbconpensbrushes.asp

[23] Pocket PC articles and samples, disponibile su: http://www.c-sharpcorner.com/Article/frmDisplayArticles.aspx?SectionID=1&SubSectionID=158

[24] Mobile Web Forms e compatibilità tra Device, disponibile su: http://www.dotnethell.it/articles/MobileWebForms.aspx

[25] SQL Anywhere Studio, disponbile su: http://www.unmondodisoluzioni.it/soluzioni/sqlanywherestudio.htm

7. Glossario

Di seguito sarà fornita una breve definizione per i termini specifici e tecnici utilizzati, in modo di facilitare al lettore la lettura di questo trattato.

ADO.NET: è la nuova tecnologia per l'accesso ai dati introdotta da Microsoft con l'avvento di Visual Studio .NET. Essa è basata completamente sulla tecnologia XML, in modo che i dati possano essere facilmente consultati da qualsiasi piattaforma.

Blocco try-catch: si tratta di un costrutto che incapsula un blocco di istruzioni che possono generare errori (eccezioni). Con il comando catch le eccezioni lanciate non generano errori di sistema, ma possono essere gestite dallo sviluppatore.

Display touch-screen: si tratta di un display con il quale è possibile interagire toccando lo schermo e non soltanto da tastiera (o altra periferica di input).

Eccezione: è un oggetto che viene generato da una istruzione quando incorre un errore; un'eccezione contiene tutte le informazioni dell'errore che l'ha generata (spesso visualizzabili tramite un messaggio).

Form: è una finestra di dialogo in cui l'utente può modificare alcuni parametri (ad esempio compilare caselle di testo).

MobiLink: E' la tecnologia che viene utilizzata per la sincronizzazione dei database. E' possibile effettuare una sincronizzazione bidirezionale fra database ASA od UltraLite e differenti tipologie di database come Sybase SQL Server, Oracle, Microsoft SQL Server e IBM DB2. La connessione avviene tramite protocolli internet standard come TCP/IP, HTTP o HTTPS

Namespace: è un insieme di classi, strutture e delegazioni che hanno lo stesso ambito o svolgono funzioni simili.

Palmare: è un computer di dimensioni ridotte (sia fisiche che a livello di memoria e prestazioni) utilizzato per gli scopi più diversi: agenda, organizzatore di appunti, applicazioni industriali (come in questo caso) ecc.

Reti wireless: sono generalmente intese come reti LAN informatiche che, a differenza delle tradizionali connessioni tramite cavi in rame (UTP/FTP/STP), utilizzano vettori radio.

Sincronizzazione: nell'ambito di questo progetto esprime il processo che viene avviato dall'utente per aggiornare il database sul palmare e quello remoto sul server, al fine di poter lavorare off-line.

Server: computer che fornisce un insieme di servizi ad altri computer (client) ad esso collegati.

SQL (Structured Query Language): linguaggio standard di interrogazione dei database, che permette la consultazione (e la modifica) dei dati, indipendentemente dall'ambiente di creazione.

Thread: è un processo che, spesso, lavora in parallelo con l'applicazione principale;

UltraLite: Si tratta di un database che richiede pochissime risorse (fino a 150Kb di memoria) ma fornisce importanti caratteristiche come integrità referenziale, transazione e criptazione.

Appendice A: Form1.cs

```
using System;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using iAnywhere.Data.AsaClient;
using iAnywhere.QAnywhere.Client;
using System.Reflection;
using System.Threading;
using iAnywhere.Data.UltraLite;
using iAnywhere.UltraLite;



namespace SmartDeviceApplication1
{

	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class SmartDeviceApplication1 : System.Windows.Forms.Form
	{
		private System.Windows.Forms.MainMenu mainMenu1;
		private System.Windows.Forms.MenuItem menuItem1;
		private System.Windows.Forms.MenuItem menuItem2;
		private System.Windows.Forms.MenuItem menuItem3;
		private System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.MenuItem menuItem4;
		private System.Windows.Forms.MenuItem menuItem5;
		private System.Windows.Forms.VScrollBar vScrollBar2;
		private System.Windows.Forms.HScrollBar hScrollBar2;
		string legend="";
		string percorso="";
		string cartina="";
		public ConnessioneDatabase Dbclass;
		DatabaseManager  DbMgr = new DatabaseManager();



		public SmartDeviceApplication1()
		{
			//
			// Required for Windows Form Designer support
			//
			Dbclass = new ConnessioneDatabase();
			InitializeComponent();


			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			base.Dispose( disposing );
		}
		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
```

```
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.mainMenu1 = new System.Windows.Forms.MainMenu();
            this.menuItem1 = new System.Windows.Forms.MenuItem();
            this.menuItem2 = new System.Windows.Forms.MenuItem();
            this.menuItem3 = new System.Windows.Forms.MenuItem();
            this.menuItem5 = new System.Windows.Forms.MenuItem();
            this.menuItem4 = new System.Windows.Forms.MenuItem();
            this.hScrollBar2 = new System.Windows.Forms.HScrollBar();
            this.vScrollBar2 = new System.Windows.Forms.VScrollBar();
            //
            // pictureBox1
            //
            this.pictureBox1.Size = new System.Drawing.Size(248, 300);
            //
            // mainMenu1
            //
            this.mainMenu1.MenuItems.Add(this.menuItem1);
            //
            // menuItem1
            //
            this.menuItem1.MenuItems.Add(this.menuItem2);
            this.menuItem1.MenuItems.Add(this.menuItem5);
            this.menuItem1.MenuItems.Add(this.menuItem3);
            this.menuItem1.MenuItems.Add(this.menuItem4);
            this.menuItem1.Text = "Menu";
            //
            // menuItem2
            //
            this.menuItem2.Text = "Sincronizza";
            this.menuItem2.Click += new System.EventHandler(this.menuItem2_Click_1);
            //
            // menuItem3
            //
            this.menuItem3.Text = "Legenda Mappa";
            this.menuItem3.Click += new System.EventHandler(this.menuItem3_Click);
            //
            // menuItem5
            //
            this.menuItem5.Text = "Seleziona Mappa";
            this.menuItem5.Click += new System.EventHandler(this.menuItem5_Click);
            //
            // menuItem4
            //
            this.menuItem4.Text = "Exit";
            this.menuItem4.Click += new System.EventHandler(this.menuItem4_Click);
            //
            // hScrollBar2
            //
            this.hScrollBar2.Location = new System.Drawing.Point(0, 255);
            this.hScrollBar2.Maximum = 600;
            this.hScrollBar2.Size = new System.Drawing.Size(244, 13);
            this.hScrollBar2.SmallChange = 3;
            this.hScrollBar2.ValueChanged += new
            System.EventHandler(this.hScrollBar2_ValueChanged);
            //
            // vScrollBar2
            //
            this.vScrollBar2.Location = new System.Drawing.Point(229, 0);
            this.vScrollBar2.Maximum = 800;
            this.vScrollBar2.Size = new System.Drawing.Size(13, 255);
            this.vScrollBar2.SmallChange = 3;
            this.vScrollBar2.ValueChanged += new
            System.EventHandler(this.vScrollBar2_ValueChanged);
            //
            // SmartDeviceApplication1
            //
            this.ClientSize = new System.Drawing.Size(242, 245);
            this.Controls.Add(this.vScrollBar2);
            this.Controls.Add(this.hScrollBar2);
            this.Controls.Add(this.pictureBox1);
            this.Menu = this.mainMenu1;
            this.Text = "SmartDeviceApplication1";

        }
        #endregion

        /// <summary>
```

```
/// The main entry point for the application.
/// </summary>

static void Main()
{
        Application.Run(new SmartDeviceApplication1());

}


private void menuItem2_Click_1(object sender, System.EventArgs e)
{
        DotNetSynchronization DNS = new DotNetSynchronization();
         //se viene premuto il tasto OK della show dialog
        if(DNS.ShowDialog()==DialogResult.OK)

        Invalidate();
        DNS.Dispose();


}

private void menuItem3_Click(object sender, System.EventArgs e)
{

                Legenda dlg3 = new Legenda();
                string inizPer=dlg3.GetAppPath();
                legend=inizPer+"\\LegendaMappa.png";
                dlg3.LoadLegenda(legend);

                // se viene premuto il tasto OK della show dialog
                if(dlg3.ShowDialog()==DialogResult.OK)

                Invalidate();
                dlg3.Dispose();

}

private void menuItem4_Click(object sender, System.EventArgs e)
{
        Application.Exit();
}

public void LoadMappa(Bitmap mappa)
{        //ottengo la dimensione dell'immagine da caricare
        int altPicture = mappa.Height-200;
        int larPicture = mappa.Width -200;


        //adatto la picturebox alle dimensioni dell'immagine
        //pictureBox1.Height=  altPicture;
        //pictureBox1.Width =  larPicture;
        pictureBox1.Size = new Size(mappa.Size.Height, mappa.Size.Height);


        //allungamento delle scrollbar adattato all'immagine
        hScrollBar2.Maximum =  larPicture  ;
        vScrollBar2.Maximum=   altPicture ;

        //la pictureBox1 carica l'immagine scelta con le dimensioni originali
        pictureBox1.Image=mappa;

        // oggetto graphics da associare alla mappa per colorare le tratte
        Graphics g = Graphics.FromImage(mappa);
        Connection ConnTemp = Dbclass.Connetti(DbMgr);
        PreparedStatement prepStmt2;

        prepStmt2 = ConnTemp.PrepareStatement("select
        Tratta.x1,Tratta.y1,Tratta.x2,Tratta.y2, TipoTraffico.codiceColore from
        Tratta join TrafficoRete on (Tratta.codStazE=TrafficoRete.codStazE and
        Tratta.codStazU=TrafficoRete.codStazU) join TipoTraffico
        on(TrafficoRete.TipoTraffico = TipoTraffico.Codice) join Mappa
        on(Tratta.codMappa = Mappa.codMappa) where Mappa.descrizione='" +
        cartina+ "'");
        ResultSet coordinateCartina = prepStmt2.ExecuteQuery();
        coordinateCartina.MoveBeforeFirst();
```

```
        while( coordinateCartina.MoveNext() )
        {

                int x_1 = coordinateCartina.GetInt(1);
                int y_1 = coordinateCartina.GetInt(2);
                int x_2 = coordinateCartina.GetInt(3);
                int y_2 = coordinateCartina.GetInt(4);
                string colore=coordinateCartina.GetString(5);

                switch(colore)
                {
                        case "G":
                                SolidBrush brushGreen = new
                                SolidBrush(Color.FromArgb(0,255,0));
                                DrawPolygonPoint(g,brushGreen, x_1, y_1, x_2,
                                y_2);
                                break;
                        case "R":
                                SolidBrush brushRed = new SolidBrush(Color.Red);
                                DrawPolygonPoint(g,brushRed, x_1, y_1, x_2, y_2);
                                break;
                        case "Y":
                                SolidBrush brushYellow= new
                                SolidBrush(Color.Yellow);
                                DrawPolygonPoint(g,brushYellow, x_1, y_1, x_2,
                                y_2);
                                break;
                        case "B":
                                SolidBrush brushBlue = new SolidBrush(Color.Blue);
                                DrawPolygonPoint(g,brushBlue, x_1, y_1, x_2, y_2);
                                break;
                        case "W":
                                SolidBrush brushWhite = new
                                SolidBrush(Color.White);
                                DrawPolygonPoint(g,brushWhite, x_1, y_1, x_2,
                                y_2);
                                break;
                        case "N":
                                SolidBrush brushBlack = new
                                SolidBrush(Color.Black);
                                DrawPolygonPoint(g,brushBlack, x_1, y_1, x_2,
                                y_2);
                                break;
                        case "M":
                                SolidBrush brushBrown = new
                                SolidBrush(Color.Chocolate);
                                DrawPolygonPoint(g,brushBrown, x_1, y_1, x_2,
                                y_2);
                                break;
                        case "O":
                                SolidBrush brushOrange = new
                                SolidBrush(Color.Orange);
                                DrawPolygonPoint(g,brushOrange, x_1, y_1, x_2,
                                y_2);
                                break;
                        case "A":
                                SolidBrush brushAqua= new SolidBrush(Color.Aqua);
                                DrawPolygonPoint(g,brushAqua, x_1, y_1, x_2, y_2);
                                break;
                        default:
                                Console.WriteLine("Invalid selection");
                                break;
                }
        }

        coordinateCartina.Close();
        ConnTemp.Close();
}


public void DrawPolygonPoint(Graphics e,SolidBrush brush,int x_1, int y_1, int
x_2, int y_2)
{
```

```
// le coordinate dei punti vanno prese negli spigoli in alto sia per il tratto di
andata che per quello di ritorno
// Creo punti che definiscono il poligono che evidenzierà il tratto autostradale.


                int x1= x_1;
                int y1= y_1;
                int x2= x_2;
                int y2= y_2;



                if ((Math.Abs(x1-x2))>(Math.Abs(y1-y2)))
                 //inclinazione da 0 a 44 per entrambe le pendenze
                {
                        int X1=x2;
                        int X2=x2;
                        int X3=x1;
                        int X4=x1;

                        int Y1=y2+4;
                        int Y2=y2-1;
                        int Y3=y1-1;
                        int Y4=y1+4;


                        Point point1 = new Point(X1,  Y1);
                        Point point2 = new Point(X2,  Y2);
                        Point point3 = new Point(X3,  Y3);
                        Point point4 = new Point(X4,  Y4);

                        Point[] curvePoints ={point1,point2,point3,point4};

                        // Disegno il poligono sullo schermo.
                        e.FillPolygon(brush, curvePoints);
                }

                        //angolo da 45 a 90 per entrambe le pendenze
                else
                {
                        int X1=x2+2;
                        int X2=x2-3;
                        int X3=x1-3;
                        int X4=x1+2;

                        int Y1=y2;
                        int Y2=y2;
                        int Y3=y1;
                        int Y4=y1;

                        Point point1 = new Point(X1,  Y1);
                        Point point2 = new Point(X2,  Y2);
                        Point point3 = new Point(X3,  Y3);
                        Point point4 = new Point(X4,  Y4);

                        Point[] curvePoints ={point1,point2,point3,point4};

                        // Disegno il poligono sul display.
                        e.FillPolygon(brush, curvePoints);


                }

        }



        private void menuItem5_Click(object sender, System.EventArgs e)
        {
                Seleziona_Mappa dlg = new Seleziona_Mappa();
                dlg.DbManager(DbMgr);
                dlg.DbCLASS(Dbclass);
                dlg.PathImmagine();
```

```
                        // se viene premuto il tasto OK della show dialog
                        if(dlg.ShowDialog()==DialogResult.OK)
                        {


                                //se non si seleziona nessuna mappa viebne visualizzato il logo
                                dell'azienda \\images\\logospi.png
                                if (percorso.Equals(""))
                                {
                                        String inizioPer= dlg.GetAppPath();
                                        percorso=inizioPer+"\\italiabn.png";

                                }

                                //creo una variabile che contiene la cartina selezionata da
                                utilizzare nella query sucessiva
                                cartina= dlg.CartinaSelezionata;
                                percorso=dlg.DirettorioSelezionato;

                                //creo una variabile che contiene l'immagine il cui percorso è
                                in "percorso"
                                Bitmap map = new Bitmap(percorso);

                                //chiamo la funzione per visualizzare la mappa scelta
                                LoadMappa (map);

                                Invalidate();


                        }
                        dlg.Dispose();




                }

                private void vScrollBar2_ValueChanged(object sender, System.EventArgs e)
                {
                 this.pictureBox1.Top = -this.vScrollBar2.Value;

                // Display the current values in the title bar.
                this.pictureBox1.Text = "x = " + this.pictureBox1.Location.X + ", y = " +
                this.pictureBox1.Location.Y;

                }

                private void hScrollBar2_ValueChanged(object sender, System.EventArgs e)
                {
                 this.pictureBox1.Left = -this.hScrollBar2.Value;

                // Display the current values in the title bar.
                this.pictureBox1.Text = "x = " + this.pictureBox1.Location.X + ", y = " +
                this.pictureBox1.Location.Y;

                }


        }
}
```

Appendice B: DotNetSynchronization.cs

```
using System;
using System.IO;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Reflection;
using System.Threading;
using iAnywhere.Data.UltraLite;


namespace SmartDeviceApplication1
{
	/// <summary>
	/// Summary description for DotNetSynchronization.
	/// </summary>
	public class DotNetSynchronization : System.Windows.Forms.Form
	{
		public iAnywhere.Data.UltraLite.ULConnectionParms ulConnectionParms1;
		public ULConnection _conn;
		public ULSyncParms _syncParms ;
		public ULSyncResult _syncResult;
		public SyncProgressDialog _sp;
		public System.Windows.Forms.TextBox textBox1;
		public System.Windows.Forms.TextBox txtHost;
		public System.Windows.Forms.Label label1;
		public System.Windows.Forms.Button btnSyncNoDialog;
		public System.Windows.Forms.MainMenu mainMenu1;

		public DotNetSynchronization()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			base.Dispose( disposing );
		}
		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.mainMenu1 = new System.Windows.Forms.MainMenu();
			this.textBox1 = new System.Windows.Forms.TextBox();
			this.txtHost = new System.Windows.Forms.TextBox();
			this.label1 = new System.Windows.Forms.Label();
			this.btnSyncNoDialog = new System.Windows.Forms.Button();
			//
			// ulConnectionParms1
			//
			ulConnectionParms1 = new iAnywhere.Data.UltraLite.ULConnectionParms();
			ulConnectionParms1.DatabaseOnCE = "\\Program
			Files\\SmartDeviceApplication1\\SmartDeviceApplication1.udb";
			ulConnectionParms1.SchemaOnCE = "\\Program
			Files\\SmartDeviceApplication1\\SmartDeviceApplication1.usm";
			//
			// textBox1
			//
			this.textBox1.Location = new System.Drawing.Point(8, 160);
			this.textBox1.Multiline = true;
			this.textBox1.Size = new System.Drawing.Size(224, 96);
			this.textBox1.Text = "";
```

```
            //
            // txtHost
            //
            this.txtHost.Location = new System.Drawing.Point(8, 32);
            this.txtHost.Size = new System.Drawing.Size(88, 20);
            this.txtHost.Text = "tslee-xp";
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(8, 16);
            this.label1.Size = new System.Drawing.Size(72, 16);
            this.label1.Text = "Host Name:";
            //
            // btnSyncNoDialog
            //
            this.btnSyncNoDialog.Location = new System.Drawing.Point(8, 64);
            this.btnSyncNoDialog.Size = new System.Drawing.Size(112, 20);
            this.btnSyncNoDialog.Text = "Sync (No Dialog)";
            this.btnSyncNoDialog.Click += new
            System.EventHandler(this.btnSyncNoDialog_Click);
            //
            // DotNetSynchronization
            //
            this.Controls.Add(this.btnSyncNoDialog);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.txtHost);
            this.Controls.Add(this.textBox1);
            this.Menu = this.mainMenu1;
            this.Text = "DotNetSynchronization";
            this.Load += new System.EventHandler(this.DotNetSynchronization_Load);

    }
    #endregion

    /// <summary>
    /// The main entry point for the application.
    /// </summary>



    private void btnSyncNoDialog_Click(object sender, System.EventArgs e)
    {
            try
            {
                    textBox1.Text = "Starting sync...";
                    this.btnSyncNoDialog.Enabled = false;
                    _syncParms.Stream = ULStreamType.TCPIP;
                    _syncParms.UserName = "SmartDeviceApplication1";
                    _syncParms.Password = "SmartDeviceApplication1";
                    _syncParms.Version = "SmartDeviceApplication1";
                    _syncParms.StreamParms = "host=" + txtHost.Text;
                    _syncParms.KeepPartialDownload = true;
                    if( _syncResult.PartialDownloadRetained )
                    {
                            _syncParms.ResumePartialDownload = true;
                    }
                    else
                    {
                            _syncParms.ResumePartialDownload = false;
                    }


                    _sp = new SyncProgressDialog( _conn, this );
                    _sp.Sync();

                    CheckRows();

            }
            catch ( System.Exception t )
            {
                    MessageBox.Show( "Exception: " + t.Message);
            }
            finally
            {
```

```
                btnSyncNoDialog.Enabled = true;
            }
        }


        private void DotNetSynchronization_Load(object sender, System.EventArgs e)
        {
            // Connect to the database
            _conn = new ULConnection( ulConnectionParms1.ToString() );
            _syncResult = _conn.SyncResult;
            _syncParms = _conn.SyncParms;
            try
            {
                _conn.OpenWithCreate();
            }
            catch ( System.Exception ex )
            {
                MessageBox.Show( ex.Message );
            }
            _conn.DatabaseID = 1;

            CheckRows();

        }



        private void CheckRows()
        {
            try
            {
                // Check the number of rows
                ULCommand Command = _conn.CreateCommand();
                Command.CommandText = "SELECT COUNT(*) FROM Colore";
                ULDataReader ResultSet = Command.ExecuteReader();
                ResultSet.MoveFirst();
                int RowCount = ResultSet.GetInt32(0);
                if (RowCount!=0)
                {textBox1.Text = "Il database è sincronizzato";}
                else
                {textBox1.Text = "Il database è vuoto, sincronizzare";}

            }
            catch( System.Exception ex)
            {
                textBox1.Text = ex.Message;
            }


        }



        private void btnCount_Click(object sender, System.EventArgs e)
        {
            CheckRows();
        }




        public void DisplayText( String s)
        {
            textBox1.Text = s;
        }


    }
}
```

Appendice C: Seleziona_Mappa.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using iAnywhere.UltraLite;
using iAnywhere.Data.AsaClient;
using iAnywhere.Data.UltraLite;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;

namespace SmartDeviceApplication1
{
	/// <summary>
	/// Summary description for Seleziona_Mappa.
	/// </summary>
	class Seleziona_Mappa : Form
	{
		ListBox listBox1;
		Button OKButton;
		Button NotOKButton;
		Label  Mappa;

		public ConnessioneDatabase dbclass1 = null;
		public DatabaseManager  dbMgr1 = null;



		string direttorio="";
		string city="";


		public  void DbManager ( DatabaseManager DbMgr)
		{

			dbMgr1 = DbMgr;

		}


		public void DbCLASS (ConnessioneDatabase Dbclass)
		{
			dbclass1 = Dbclass ;

		}


		public void PathImmagine()
		{

			Connection ConnTemp = dbclass1.Connetti(dbMgr1);
			PreparedStatement prepStmt;
			prepStmt = ConnTemp.PrepareStatement("select descrizione from mappa
			order by descrizione");
			ResultSet cityNames1= prepStmt.ExecuteQuery();
			cityNames1.MoveBeforeFirst();
			while( cityNames1.MoveNext() )
			{

				listBox1.Items.Add( cityNames1.GetString( 1 ) );

			}
			listBox1.Show();
			ConnTemp.Close();

		}
```

```
public void listBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{


        Connection ConnTemp = dbclass1.Connetti(dbMgr1);
        PreparedStatement prepStmt;
        string città = listBox1.SelectedItem.ToString();
        prepStmt = ConnTemp.PrepareStatement("select percorsoFile from mappa
        where descrizione = '" + città + "'");
        city = listBox1.SelectedItem.ToString();
        string finePath = "";

        ResultSet cityPath = prepStmt.ExecuteQuery();
        cityPath.MoveBeforeFirst();
        while( cityPath.MoveNext() )
        {

                finePath =  cityPath.GetString( 1 ) ;

        }

        cityPath.Close();
        ConnTemp.Close();



        // compongo il percorso totale dell'immagine e lo restituisco mediante
        la get della funzione PathImmagine
        string inizioPath=GetAppPath();
        direttorio= inizioPath+finePath;
}



public string CartinaSelezionata
{
        get {return city;}
}

public string DirettorioSelezionato
{
        get {return direttorio;}
}


public string GetAppPath()
{
        System.Reflection.Module[] modules =
        System.Reflection.Assembly.GetExecutingAssembly().GetModules();

        string aPath = System.IO.Path.GetDirectoryName
        (modules[0].FullyQualifiedName);

        if ((aPath != "") && (aPath[aPath.Length-1] != '\\'))

                aPath += '\\';

        return aPath;

}

public Seleziona_Mappa()
{
        // Initialize the dialog's visual properties

        FormBorderStyle = FormBorderStyle.FixedDialog;
        MaximizeBox = false;
        MinimizeBox = false;
        Text = "Seleziona Mappa";


        listBox1 = new ListBox();
        listBox1.Location = new Point(12, 51);
        listBox1.Size= new Size(210,165);
        listBox1.SelectedIndexChanged += new
        System.EventHandler(listBox1_SelectedIndexChanged);
```

```
                        Mappa = new Label ();
                        Mappa.Location = new Point (12, 10);
                        Mappa.Size = new Size (130, 40);
                        Mappa.Text = "Seleziona una mappa:";


                        OKButton = new Button ();
                        OKButton.Location = new Point (12, 250);
                        OKButton.Text = "Ok";
                        OKButton.DialogResult = DialogResult.OK;

                        NotOKButton = new Button ();
                        NotOKButton.Location = new Point (152, 250);
                        NotOKButton.Text = "Cancel";
                        NotOKButton.DialogResult = DialogResult.Cancel;

                        // Add the controls to the dialog

                        Controls.Add (NotOKButton);
                        Controls.Add(OKButton);
                        Controls.Add(listBox1);
                        Controls.Add (Mappa);
                }

        }

}
```

Appendice D: Legenda.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace SmartDeviceApplication1
{
	/// <summary>
	/// Summary description for Legenda.
	/// </summary>
	class Legenda : Form
	{


		PictureBox Pb;

		public string GetAppPath()
		{

			System.Reflection.Module[] modules =
			System.Reflection.Assembly.GetExecutingAssembly().GetModules();

			string aPath = System.IO.Path.GetDirectoryName
			(modules[0].FullyQualifiedName);

			if ((aPath != "") && (aPath[aPath.Length-1] != '\\'))

				aPath += '\\';

			return aPath;

		}

		public void LoadLegenda(string leg)

		{
			Bitmap legenda = new Bitmap (leg);
			Pb.Image=legenda;


		}


		public Legenda()
		{
			// Initialize the dialog's visual properties

			FormBorderStyle = FormBorderStyle.FixedDialog;
			MaximizeBox = false;
			MinimizeBox = false;
			Text = "Legenda";


			Pb = new PictureBox();
			Pb.Location = new Point (15, 40);
			Pb.Size= new Size (210,212);
			Pb.SizeMode = PictureBoxSizeMode.CenterImage;

			Controls.Add(Pb);
		}


	}

}
```

Appendice E: ConnessioneDatabase.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using iAnywhere.UltraLite;
using iAnywhere.Data.AsaClient;

namespace SmartDeviceApplication1
{
	/// <summary>
	/// Summary description for ConnessioneDatabase.
	/// </summary>
	public class ConnessioneDatabase
	{


		public Connection Connetti(DatabaseManager dbm)
		{

			CreateParms parms = new CreateParms();
			parms.DatabaseOnCE ="\\Program
			Files\\SmartDeviceApplication1\\SmartDeviceApplication1.udb";
			parms.Schema.SchemaOnCE ="\\Program
			Files\\SmartDeviceApplication1\\SmartDeviceApplication1.usm";

			Connection Conn = null;
			try
			{
				try
				{
					Conn = dbm.OpenConnection( parms );
				//MessageBox.Show( "Connected to an existing database." );
				}
				catch( SQLException err )
				{
					if( err.ErrorCode ==
					SQLCode.SQLE_ULTRALITE_DATABASE_NOT_FOUND )
					{
						Conn = dbm.CreateDatabase( parms );
						MessageBox.Show( "Connected to a new database.");
					}
					else
					{
						throw err;
					}
				}


			}

			catch( SQLException err )
			{
			MessageBox.Show("Exception: " + err.Message +" sqlcode=" +
			err.ErrorCode);
			}
			catch ( System.Exception t )
			{
				MessageBox.Show( "Exception: " + t.Message);
			}

			return Conn;
		}



	}
}
```

Appendice F: SyncProgressDialog.cs

```
using System;
using System.IO;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using iAnywhere.Data.AsaClient;
using System.Reflection;
using System.Threading;
using iAnywhere.UltraLite;
using iAnywhere.Data.UltraLite;


namespace SmartDeviceApplication1
{

	public class SyncProgressDialog :ULSyncProgressListener
	{
		public ULConnection _conn;
		public bool _syncCancelled;
		public bool _inSync;
		public DotNetSynchronization _form;

		public SyncProgressDialog(ULConnection conn, DotNetSynchronization form )
		{
			try
			{
				_conn = conn;
				_inSync = true;
				_syncCancelled = false;
				_form = form;
			}
			catch(System.Exception ex )
			{
				MessageBox.Show(ex.Message);
			}
		}

		public void Sync()
		{
			try
			{
				_conn.Synchronize( this );
				_inSync = false;
			}
			catch(System.Exception ex )
			{
				MessageBox.Show(ex.Message);
			}
		}
		public virtual bool SyncProgressed( ULSyncProgressData data )
		{
			System.String msg = null;
			msg = data.State.ToString();
			_form.DisplayText( msg );
			Application.DoEvents();
			return _syncCancelled;
		}

		public void CancelSync()
		{
			if( _inSync )
			{
				if( !_syncCancelled )
				{
					_syncCancelled = true;
				}
			}
		}

	}
}
```

A conclusione di questo lavoro, desidero ringraziare la prof.ssa Sonia Bergamaschi, che ha dimostrato di essere disponibile in qualunque momento e mi ha offerto l’opportunità di fare questa esperienza.

Ringrazio i miei genitori.

Ringrazio anche tutti i miei amici.