Università degli studi
di Pisa

Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea Triennale in Informatica

UNIVERSITÀ DI PISA

Progettazione e realizzazione di un sistema per la geolocalizzazione di indirizzi

Francesco Margeri

Relazione di tirocinio

Relatore:
Prof. Francesco Romani

Anno Accademico 2011/2012

Ringraziamenti

Questo è un altro importante passo
nel cammino della vita.
Ringrazio tutti coloro che
mi hanno permesso di compierlo.
Un ringraziamento speciale alla mia
compagna Federica.

INDICE

Introduzione

Presso l'azienda Thesis Contents SRL di Firenze ho svolto il tirocinio della durata di 450 ore, il cui obbiettivo è stato lo sviluppo di un sistema che a partire da uno stream di testo, effettua l'individuazione, il riconoscimento, la geolocalizzazione e la catalogazione su database di indirizzi stradali. La Geolocalizzazione è intesa come l'identificazione della posizione geografica nel mondo reale di un dato oggetto, come può essere un edificio o un luogo.

I problemi nell'identificare all'interno di un testo un indirizzo corretto, possono essere molteplici, soprattuto nei casi in cui l'indirizzo viene scritto omettendo alcuni dei suoi campi, dandoli per scontati o perché non se ne conosce l'entità. La corretta scrittura dell'indirizzo contribuisce a garantire un' individuazione più rapida per il riconoscimento ma soprattuto permette di avere una geolocalizzazione più precisa e affidabile.

L'istituto nazionale di statistica (ISTAT) è un ente di ricerca pubblico italiano che definisce, tra le sue attività, anche le informazioni territoriali ed amministrative sul territorio nazionale. L'istat pone in evidenza il numero complessivo delle province italiane a 110 e il numero dei comuni pari a 8092 unità amministrative, tali informazioni risalgono al 30 giugno 2011 periodo in cui è stato svolto il tirocinio, quindi i dati elaborati si basano su queste informazioni.

Nello scenario così descritto è inserito questo progetto di ricerca, i cui obiettivi principali sono l'individuazione di un indirizzo all'interno di un testo, il riconoscimento della sua struttura analizzando tutti i suoi campi dal nome del comune e della provincia, fino ad arrivare al codice di avviamento postale (CAP), oltre all'informazioni presenti sulla nomenclatura della strada e i suoi corrispettivi identificativi, come possono essere: via, piazza, corso e numero civico.

Lo sviluppo del progetto è stato reso possibile mediante l'utilizzo di tecniche sul riconoscimento di pattern, gestione di database relazionali e l'uso di API per la geolocalizzazione. L'implementazione è stata sviluppata in modo da essere portabile su un architettura con paradigma client/server, l'ambiente di sviluppo con il quale ho dovuto prendere confidenza è stato Microsoft Visual Studio con dati su Sql Server.

Scendendo nei particolari, per analizzare e valutare questi processi di riconoscimento, è stato eseguito uno studio su un campione di circa 13000 stringhe contenenti al loro interno un indirizzo, tali stringhe venivano prelevate dal web, quindi scritte da utenti sparsi per l'Italia che potevano segnalare degli eventi, all'interno del loro blog, o da pagine web varie.

La validazione dei risultati ottenuti è stata quindi sostenuta dalla validità dei dati in input, operanti su scenari pressoché identici alla realtà.

La dettagliata analisi sperimentale dei dati ottenuti nel riconoscimento permette di decretare un riconoscimento pari all'90% sul campione utilizzato.

L'obiettivo prefissato è stato raggiunto con successo, attraverso il lavoro svolto in autonomia, rispettando le scelte e le decisioni imposte dal tutore aziendale grazie anche ai vari confronti avvenuti sugli argomenti trattati. In particolare per le tecnologie utilizzate, non avendone familiarità, mi sono adattato all'ambiente di sviluppo trovato e nel periodo di tirocinio, ho dovuto svolgere degli aggiornamenti al riguardo attraverso lo studio su libri specifici, lezioni e consigli dati direttamente dal datore di lavoro. Inoltre per verificare un corretto apprendimento, sono susseguiti esercizi e test, in modo tale da poter proseguire in maniera autonoma allo sviluppo del progetto attraverso, appunto, le nuove metodologie acquisite. La documentazione relativa alle metodologie usate si trovano nel capitolo 2 per la prima parte del progetto, e nel capitolo 6 per la seconda parte.

La prima fase del processo consiste nella canonizzazione dell'indirizzo, in altre parole, nel rendere l'indirizzo standard e completo riordinandolo e, laddove possibile, aggiungendo informazioni mancanti come per esempio la provincia o il cap.

In seconda fase, per attuare la geolocalizzazione è stato creato un programma con parametri di chiamata per le informazioni necessarie all'individuazione dell'indirizzo, mentre per geolocalizzare gli indirizzi canonizzati sono state usate le api di Google Geocoding.

Una volta che l'indirizzo è stato canonizzato e georeferenziato, le sue coordinate sono memorizzate in un database. Un possibile utilizzo del database è di riuscire a sostituire Google, là dove in un futuro non molto prossimo si riesca a immagazzinare il maggior numero di geolocalizzazioni. Questo permetterebbe quindi di abbandonare la necessità di utilizzare Google e ottenere tali informazioni utilizzando la propria banca dati di indirizzi.

Compiuta la geolocalizzazione è stato creato un programma che visualizza su una mappa gli indirizzi georeferenziati con la possibilità di aggiungerne dei nuovi, modificarne nome e coordinate, e calcolarne il percorso.

La trattazione dei suddetti argomenti in questo documento è suddivisa in due parti: la prima riporta il caso di studio per l'individuazione e il riconoscimento dei componenti dell'indirizzo, mentre la seconda riporta lo studio e l'implementazione sulla geolocalizzazione di tali indirizzi, a sua volta seguita dalle conclusioni e dagli sviluppi futuri.

Il lavoro nel dettaglio è stato così suddiviso:

- Presentazione delle tematiche della prima parte del progetto, con conseguimento degli obbiettivi raggiunti.
- Descrizione delle tecnologie utilizzate per l'acquisizione dei dati e per lo sviluppo del progetto.
- Panoramica dei campi che compongono un indirizzo, con i relativi problemi riscontrati per il riconoscimento di ogni singolo campo.
- Progettazione secondo le specifiche.
- Seconda parte del progetto, quella sulla geolocalizzazione dell'indirizzo.
- Descrizione delle tecnologie utilizzate per la geolocalizzazione e la visualizzazione di eventi su una mappa online.
- Discussione sull'implementazione di tale realizzazione.

Partiamo dunque con la relazione vera e propria del progetto.

PARTE I

Riconoscimento dei componenti dell'indirizzo

Capitolo 1

Analisi del progetto

Attraverso una banca dati sono stati selezionati tutti gli eventi sul territorio nazionale, ad ogni evento è associato un indirizzo che, essendo reperito da fonti disomogenee, è scritto nei modi più vari possibili.

Per **indirizzo** s'intende una serie d'informazioni presentate in un formato codificato, necessario per definire la localizzazione di un edificio o di una struttura. Generalmente è costituito dal nome assegnato all'infrastruttura su cui sorge l'edificio (via, viale, piazza, corso, ecc..) con un suo eventuale numero civico, e dal nome della città e della nazione. A queste informazioni solitamente è aggiunto anche il codice postale.

A questo punto lo scopo che si vuole raggiungere è, da uno stream di testo che rappresenta un indirizzo, compiere l'individuazione e il riconoscimento dei componenti. Per questo risulta necessaria la presenza di un database (vedi paragrafo 1.1) contenente informazioni su tutti i comuni d'Italia, province e frazioni, correlate da cap, codice istat ed altre voci. Da ricordare che il codice istat altro non è che un sistema di codici che corrisponde ad un codice già utilizzato per il codice fiscale personale. Questo codice è composto da una lettera e tre numeri, e viene assegnato in ordine alfabetico crescente sull'elenco di tutti i comuni di Italia, senza tener conto della Provincia.

1.1 Descrizione del Database

In informatica il termine database, banca dati o base di dati, indica un insieme di archivi collegati secondo un particolare modello logico (relazionale, gerarchico, reticolare o a oggetti). La sua funzione è quella di consentire la gestione dei dati stessi (inserimento, ricerca, cancellazione ed aggiornamento) da parte di particolari applicazioni software dedicate.

Nei database più moderni, ovvero quelli basati sul modello relazionale, i dati sono suddivisi per argomenti (in tabelle) che vengono suddivisi per categorie (campi) con tutte le possibili operazioni sopra elencate. Tale suddivisione e funzionalità, rende i database notevolmente più efficienti rispetto ad un archivio di dati creato ad esempio tramite file system di un sistema operativo su un computer almeno per la gestione di dati complessi.

Informalmente e impropriamente, la parola "database" viene spesso usata come abbreviazione dell'espressione database management system (DBMS), che invece si riferisce a una vasta categoria di sistemi software che consentono la creazione e la manipolazione (gestione) efficiente dei dati di un database.

Le tabelle che sono state usate per lo svolgimento del progetto risiedono in un database, ed è stato possibile reperire alle informazioni ad esso contenute tramite delle interrogazioni sql.

```
Q1:
SELECT dbo_province.sigla_prov AS sigla_provincia FROM (dbo_grab_pagine INNER
JOIN (dbo_province ON dbo_comuni.sigla_prov))WHERE
dbo_grab_pagine.longitudine Is Not Null ORDER BY dbo_province_sigla_prov
```

Tabella 1.1: Esempio di Query

E' il gestore delle interrogazioni che si occupa di elaborare le richieste eseguite dal programma, di solito espresse in SQL, quindi in un linguaggio di tipo dichiarativo (in cui si descrivono i dati che si vogliono ottenere), e di tradurle in un insieme di operazioni, che saranno poi effettivamente eseguite. Di solito vi sono più modi diversi di tradurre un'interrogazione e la funzione principale del gestore delle interrogazioni è di scegliere fra le varie possibilità quella migliore, quella cioè che richiede un minor tempo di elaborazione ed una minore occupazione di memoria. Ad esempio, un'ottimizzazione consiste nell'anticipare sempre le operazioni di selezione, in modo da diminuire fin dall'inizio il numero di record da elaborare, con ovvi miglioramenti nell'occupazione di memoria e nella velocità. Altre ottimizzazioni sono fatte basandosi su criteri di tipo statistico: la grandezza di una tabella, come le tabelle sono fisicamente memorizzate, ecc. Alla fine dell'elaborazione il gestore delle interrogazioni darà delle direttive al gestore dei metodi di accesso per trovare le tuple.

Sotto riportiamo una schema delle tabelle usate per lo scopo del progetto, suddivise in argomenti e categorie dove gli argomenti corrispondo ai titoli delle tabelle e le categorie corrispondono ai campi che si trovano all'interno di ogni tabella.

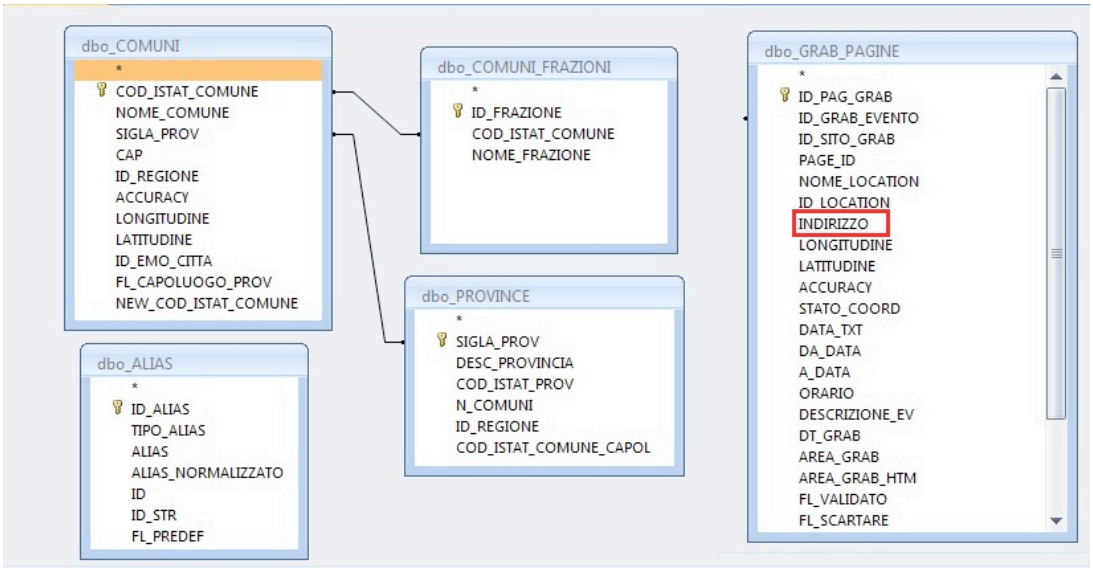

Figura 1.1. Screenshot delle tabelle usate per l'analisi dell'indirizzo

L'indirizzo che è passato in input per essere analizzato dal programma è un campo che si trova, nel nostro caso, all'interno di una tabella chiamata dbo_GRAB_PAGINE. Tale campo è stato evidenziato nell'immagine con un rettangolo rosso, e contiene la lista di tutti gli indirizzi da elaborare.

Nella tabella chiamata dbo_COMUNI_FRAZIONI sono state inserite tutte le frazioni d'Italia, in dbo_PROVINCE le province, in dbo_COMUNI i comuni e in dbo_ALIAS sono stati inseriti tutti i comuni nazionali con l'aggiunta dei suoi alias. La descrizione di quest'ultima tabella avviene in dettaglio nel paragrafo 3.1.

1.2 Standardizzare un indirizzo

Nei migliori dei casi la stringa ottimale si presenta in questo modo:

- `Via Fermi n. 21 Vinci Empoli FI 50053`

dove sono presenti i componenti:
via – frazione – comune – provincia – cap
posizionati seguendo l'ordine che si usa nella maggior parte dei casi quando si scrive un indirizzo.

Il nostro scopo sarà ricondurre tutti gli indirizzi non conformi al nostro caso ottimale. I casi, infatti, non sempre sono della tipologia dell'esempio qui sopra citato, possiamo anche incontrare stringhe di questo tipo:

- `Empoli Via Fermi, 21`

uno dei tanti possibili casi che non rispecchia la nostra stringa ottimale. Come notiamo non è rispettato l'ordine suddetto, poiché l'elemento *Comune* è anteposto a quello della *Via*, ed inoltre vi sono dei componenti mancanti, quali *Cap* e *Provincia*.

Sarà quindi nostro obiettivo principale creare un programma che sia in grado di riordinare i componenti e completare la stringa con quelli mancanti deducendoli dalle informazioni già presenti.

1.3 Obiettivi raggiunti

Nella seguente immagine è possibile osservare il risultato raggiunto. Per facilitarne la comprensione, sono stati delimitati all'interno di un rettangolo rosso la lista degli indirizzi stringa, passati in input per essere analizzati attraverso il programma e, dentro un rettangolo verde, i risultati ottenuti dal riconoscimento.

dbo_GRAB_PAGINE

INDIRIZZO	x_indirizzo	x_frazione	x_cap	x_id_comune	x_comune	x_provincia
Spilamberto (MO)			41057	036045	Spilamberto	MO
Via Circonvallazione, 7 Lunano 61026 PU	Via Circonvallazione n. 7		61026	041022	Lunano	PU
Via Giudecca, 2 Gallipoli 73014 LE	Via Giudecca n. 2		73014	075031	Gallipoli	LE
Cusano Milanino (MI)			20095	015098	Cusano Milanino	MI
Via Ronchi Fosdondo, 11 Correggio 42015 RE	Via Ronchi Fosdondo n. 11		42015	035020	Correggio	RE
Via Galvani, 24 Roma 00100 RM	Via Galvani n. 24		00100	058091	Roma	RM
Perugia (PG)			06100	054039	Perugia	PG
Via Sant'Uguzzone, 26 Milano 20126 MI	Via Sant'Uguzzone n. 26		20126	015146	Milano	MI
Via Cao Prà, 82 Lugagnano 37060 VR	Via Cao Prà n. 82	Lugagnano	37060	023083	Sona	VR
Via Ronchi Fosdondo, 11 Correggio 42015 RE	Via Ronchi Fosdondo n. 11		42015	035020	Correggio	RE
Spilamberto (MO)			41057	036045	Spilamberto	MO
Via Cao Prà, 82 Lugagnano 37060 VR	Via Cao Prà n. 82	Lugagnano	37060	023083	Sona	VR
Via G. Di Vittorio, 6 c/o MediolanumForum Assago 20090 MI	Via G. Di Vittorio n. 6 c/o MediolanumForum		20090	015011	Assago	MI
Via Don Minzoni, 96/b Taneto di Gattatico 42043 RE	Via Don Minzoni n. 96/b	Taneto	42043	035022	Gattatico	RE
Via Mazzini angolo Via Doria Cirié 10073 TO	Via Mazzini angolo Via Doria		10073	001086	Cirié	TO
Via Maestri del Lavoro, 23 Legnano 20025 MI	Via Maestri del Lavoro n. 23		20025	015118	Legnano	MI
Quarrata (PT)			51039	047017	Quarrata	PT
Via E.Curiel, 39 Mezzago 20050 MB	Via E.Curiel n. 39		20050	015145	Mezzago	MB
Viale Martiri 28 Giugno Riva del Garda 38066 TN	Viale Martiri 28 Giugno		38066	022153	Riva del Garda	TN
Via Pistoiese ang. Via Lombardia Firenze 50145 FI	Via Pistoiese ang. Via Lombardia		50145	048017	Firenze	FI
S.Vito (TA)			09040	092064	San Vito	CA
Via E.Curiel, 39 Mezzago 20050 MB	Via E.Curiel n. 39		20050	015145	Mezzago	MB
S.S. Romea, 32 S.Giuseppe di Comacchio 44020 FE	S.S. Romea n. 32 Giuseppe		44020	038006	Comacchio	FE
Corso Vigevano, 33/u Torino 10100 TO	Corso Vigevano n. 33/u		10100	001272	Torino	TO
Viale Carlo Felice, 69b Roma 00100 RM	Viale Carlo Felice n. 69b		00100	058091	Roma	RM
Via Sammartini, 30 Milano 20125 MI	Via Sammartini n. 30		20125	015146	Milano	MI
Spilamberto (MO)			41057	036045	Spilamberto	MO
Via Pallavicino, 35 Torino 10100 TO	Via Pallavicino n. 35		10100	001272	Torino	TO
Catania (CT)			95100	087015	Catania	CT
Via E.Curiel, 39 Mezzago 20050 MB	Via E.Curiel n. 39		20050	015145	Mezzago	MB
Via Camillo Ugoni, 18 Milano 20100 MI	Via Camillo Ugoni n. 18		20100	015146	Milano	MI

Record: 1 | Non filtrato | Cerca | BLOC NUM

Figura 1.2. Screenshot dei risultati ottenuti con la scomposizione di un indirizzo

Si nota quindi come il programma, da una stringa INDIRIZZO, riesca a decomporla nelle sue componenti principali associandole ai campi sopra raffigurati e, là dove possibile, riempie quelli che sarebbero stati vuoti se ricavati direttamente da una stringa che non presentava tutte le componenti richieste.

Alla fine dell'elaborazione possiamo così ricomporre una nuova stringa indirizzo con i campi trovati, per poi posizionarli nell'ordine desiderato.

Ad Esempio, ricomponendo i campi trovati da una stringa presa a caso, come può essere quella sotto indicata:

- Lugagnano Via Cao Prà, 82

rispettando il seguente ordine di ricomposizione:
x_indirizzo + x_frazione + x_comune + x_provincia + x_cap

otteniamo il seguente risultato:

- Via Cao Prà n.82 Lugagnano Sona VR 37060

(Sottolineati i campi aggiunti dal sistema).

Siamo quindi riusciti a raggiungere l'obbiettivo prefissatoci cioè, canonizzare l'indirizzo seguendo l'ordine che si usa nella maggior parte dei casi, riconducendoci così al nostro caso ottimale, precedentemente analizzato (vedi paragrafo 1.2).

Possiamo inoltre notare, sempre nello stesso esempio, come il programma è riuscito a ricavarsi tutte le componenti necessarie per avere un' informazione completa dell'indirizzo, infatti il nome del comune *Sona* che mancava, è stato ricavato dall'informazione del nome di frazione *Lugagnano*, lo stesso vale per il Cap *37060* e la sigla della provincia *VR*. Questo è stato possibile attraverso un'implementazione specifica, descritta in dettaglio nel capitolo 4.

Capitolo 2

Le tecnologie utilizzate

Come già anticipato nell'introduzione, lo sviluppo di questo progetto ha richiesto l'utilizzo di alcune tecnologie che verranno dettagliate nei paragrafi successivi.

Nel paragrafo 2.1 è descritto Microsoft Access il gestore di database; nel paragrafo 2.2 è riportato lo studio fatto per la scrittura del codice in Visual Basic; nel paragrafo 2.3 viene riportato l'utilizzo fatto di Sql Server all'interno del progetto, con particolare riferimento all'utilizzo di creazione di nuove tabelle.

2.1 Microsoft Access

Microsoft Access è un relational database management system che serve in parte a salvare dati realizzati da Microsoft, incluso nel pacchetto Microsoft Office Professional ed unisce il motore relazionale Microsoft Jet Database Engine con una interfaccia grafica.

Può utilizzare dati immagazzinati in formato Access/Jet, SQL Server, Oracle o qualsiasi database in formato compatibile ODBC. La struttura di salvataggio segue il modello tabella relazionale: ossia è possibile immagazzinare i dati da gestire in tabelle composte da un numero elevato di record, ed ogni record contiene i dati distinti per campi. Se una tabella non fosse sufficiente per immagazzinare i dati necessari e fosse necessario utilizzarne altre, è possibile a questo punto collegare le varie tabelle tra di loro con una relazione. Questo consente l'esame dei dati contenuti nel database utilizzando diverse tabelle e quindi giungere ad una pluralità di dati anche complessa.

A differenza di altri ambienti di sviluppo, in Access un unico file comprende tutti gli elementi utilizzabili per lo sviluppo di applicazioni complete: tabelle, query, maschere, report, macro, pagine e moduli. È comunque possibile, con tutte le versioni, progettare applicazioni nelle quali si mantenga la separazione fisica tra tabelle di dati (Back-End o BE) ed i restanti elementi (Front-End o FE). Queste soluzioni permettono di migliorare la distribuzione e la manutenzione di applicazioni condivise tra più utenti.

Le tabelle sono i contenitori dove vengono memorizzati i dati; è disponibile una interfaccia grafica elementare per la definizione o la modifica delle proprietà dei campi, inclusa la definizione degli indici e della chiave primaria (che può essere basata su più campi). Come nel caso di database professionali, il controllo della sintassi esercitato da Access può consistere in soli messaggi di avvertimento nei casi in cui la modifica dei campi può comportare perdita irreparabile dei dati (ad esempio, la riduzione della dimensione di un campo nel quale sono già presenti dati di lunghezza maggiore): questo non è necessariamente visto come un difetto

di Access, per quanto la destinazione potenziale del prodotto, ad utenti non esperti, possa rendere questi casi più frequenti.

Le query sono gli strumenti idonei all'interrogazione ed alla manipolazione dei dati. Access dispone sin dall'origine di un ambiente grafico per la definizione delle query (detto Query By Example o QBE) che permette anche ad utenti poco esperti la loro costruzione, con un minimo di controllo della correttezza sintattica; questa facilità, per contro, può comportare situazioni di blocco del sistema come conseguenza di errori concettuali che comportino ricorsioni. Il linguaggio utilizzato nella definizione delle query è una versione leggermente semplificata di T-SQL; in alternativa all'ambiente QBE è possibile utilizzare direttamente questo, anche per ottenere query non altrimenti costruibili con QBE.

Le maschere (o form) consistono negli elementi grafici utili alla interazione da parte degli utenti con i dati delle tabelle o delle query. Le maschere possono contenere gli elementi standard di Access ed elementi aggiuntivi (ad esempio, controlli OCX sviluppati a parte). Le maschere possono includere codice VBA destinato all'automazione degli elementi contenuti; l'area di visibilità delle routine è locale.

I report consentono la visualizzazione, destinata alla stampa, dei risultati basati sui dati, tabelle e query. L'ambiente grafico destinato alla costruzione della struttura dei report ricalca quello delle maschere, pur conservando le differenze dovute alla diversa destinazione; sono disponibili funzioni di base, quali aggregazione dei dati e totali parziali. Anche in questo caso è possibile l'inserimento di codice VBA (area di visibilità locale) per un livello maggiore di automazione.

Le macro possono contenere semplici sequenze di istruzioni, tipicamente tutto ciò che è possibile ottenere attraverso i menu di Access. Si tratta di elementi che permettono scarsa interazione con l'utente, per contro la loro costruzione è semplice.

Le pagine (ovvero pagine di accesso ai dati) permettono la pubblicazione dei dati attraverso un server web. Sono state introdotte a partire dalla versione 2000 di Access.

I moduli possono contenere codice VBA (moduli di codice e classi) che si intende rendere globali (salvo specifica dichiarazione), ovvero richiamabili da uno qualsiasi degli altri elementi dell'applicazione.

Una funzionalità presente in tutte le versioni di Access consente di accedere a dati residenti in file di database esterni, sotto forma di tabelle collegate. Database strutturati in questo modo facilitano la distribuzione e la manutenzione della medesima applicazione a più utenti, fermo restando le limitazioni del motore Access/Jet circa il numero massimo di accessi simultanei. Occorre sottolineare che queste soluzioni non possono essere definite "client/server" in quanto il carico di lavoro per la elaborazione dati è sempre locale; una alternativa praticabile con le versioni dalla 2000 in poi è rappresentata dal progetto di database (estensione del file .adp) dove di fatto si realizza solo la parte di presentazione grafica, in

appoggio a motori professionali già esistenti, ai quali è demandato il lavoro di elaborazione.

Le tabelle collegate sono utilizzabili allo stesso modo delle tabelle residenti, con l'unica limitazione data dalla non modificabilità della loro struttura se non nel database nel quale risiedono fisicamente. Le tabelle possono essere collegate attraverso il motore di database di Access se risiedono fisicamente in altri database Access, o in alcuni formati di file di database "standard", oppure via ODBC. In questo caso l'accesso a database eterogenei può richiedere la installazione di driver specifici.

Le versioni più recenti di Access dispongono di procedure guidate per la separazione in file distinti dei dati e dei restanti elementi a partire da applicazioni inizialmente costruite in un singolo file Access.

Per esigenze di sviluppo professionali è disponibile nel prodotto il linguaggio di programmazione Microsoft Visual Basic (per gli applicativi Office definito VBA - Visual Basic for Applications). Il linguaggio che è usato nel corrente progetto. (paragrafo 2.2).

Sebbene il prodotto supporti tecniche di programmazione object-oriented (OO), tuttavia non costituisce un ambiente di sviluppo interamente orientato agli oggetti.

2.2 Visual Basic

Il Visual Basic (formalmente abbreviato VB) è un linguaggio di programmazione event driven, la cui sintassi deriva dal BASIC. Le caratteristiche principali di tale linguaggio sono: facilità d'uso in quanto non utilizza formalità di punteggiatura tipica di quasi tutti gli altri linguaggi e il pratico accesso alle basi dati, caratteristica fondamentale per lo svolgimento di tale progetto.

2.3 Sql Server

Microsoft SQL Server è un DBMS relazionale, meglio noto come Relational Database Management System (RDBMS), usa una variante del linguaggio SQL standard (lo standard ISO certificato nel 1992) chiamata T-SQL Transact-SQL. Sia Microsoft SQL Server che Sybase Adaptive Server Enterprise comunicano sulla rete utilizzando un protocollo a livello di applicazione chiamato "Tabular Data Stream" (TDS). SQL Server supporta anche "Open Database Connectivity" (ODBC). Il servizio di SQL Server risponde per default sulla porta 1433. Il progetto ne ha richiesto l'uso in quanto si è lavorato con un database lato server.

Capitolo 3

I campi dell'indirizzo

Studiando un elevato campione d'indirizzi reperiti da varie fonti, arriviamo ad un punto di analisi essenziale per il progetto, quello cioè di analizzare la stringa suddividendola in token di parole. In informatica token è un blocco di testo categorizzato, normalmente costituito da caratteri indivisibili chiamati lessemi. Un analizzatore lessicale inizialmente legge i lessemi e li suddivide in categorie a seconda della loro funzione, dando loro un significato. Questa assegnazione di significato è chiamata tokenizzazione. Nel nostro caso analizziamo gli elementi partendo da quelli posizionati alla fine, dove cioè sono presenti informazioni di facile riconoscimento, come il cap e la sigla della provincia, per poi proseguire da destra verso sinistra nella ricerca del nome, del comune e della frazione.

via	frazione	comune	provincia	cap
via Bardini, 28	Corniola	Empoli	FI	50100

(Analisi a ritroso dell'indirizzo)

In questo modo, quello che si è cercato di ottenere, è di avere come ultima informazione la più difficile da riconoscere, ovvero la descrizione dello stabilimento che può variare a seconda si tratti di una `Via` con o senza numero civico o altri elementi di identificazione, come:

- Angolo, Borgo, Contrada, Corso, Corte, Frazione, Lido, Località, Piazza, Piazzale, Ponte, Sobborgo, Stazione, Strada, Viale, Vicolo.

O di indicazioni su come arrivarci, come per esempio specificando il nome di una super strada S.S e i km da percorrere. Per i motivi sopra elencati, sarebbe riduttivo usare `Via` per indicare la descrizione di uno stabilimento, quindi da ora in poi la indicheremo con il nome campo `Indirizzo` che contiene al suo interno gli elementi di identificazione.

Si è scelto di suddividere le informazioni dell'intero indirizzo nei campi più importanti, ricavati attraverso il riconoscimento sintattico della stringa, con l'aiuto di interrogazioni a database che contengono la lista di tutti i comuni, frazioni, cap e provincie d'Italia.

- Il campo **`Cap`** indica il codice postale (esempio `50100`)
- Il campo **`Provincia`** indica solo la sigla della provincia (esempio `FI`)

- Il campo **Comune** indica il nome del comune (esempio Empoli)
- Il campo **Frazione** indica la frazione del comune (esempio Corniola)
- Il campo **Indirizzo** indica la via e/o informazioni sul posto (esempio Via Emilio Bardini, 48 angolo Pisana).

3.1 Problemi riscontrati

Il funzionamento per il riconoscimento delle componenti dell'indirizzo consiste nel rimuovere il campo ogni qual volta viene riconosciuto, per poi continuare l'analisi della stringa sulle informazioni rimaste.

L'Indirizzo, il Cap e la Provincia sono stati i campi nei quali si sono verificate problematiche più semplici da risolvere.

La Frazione è stata introdotta successivamente per facilitare l'individuazione del campo Indirizzo.

Senza dubbio, il campo che ha presentato più problemi è stato il Comune, ne vedremo in seguito il perché.

Cominciamo dunque l'analisi delle problematiche dal principio.

Il **campo Indirizzo**, ha presentato problematiche legate alla difficoltà di trovare modalità di standardizzazioni univoche. All'inizio si era pensato di risolvere il problema creando una lista completa di tutte le parole che potevano indicare il nome dell'indirizzo (via, corso, piazza, vicolo ecc…) ma analizzando un campione casuale di stringhe d'indirizzi ci siamo resi conto che un luogo veniva indicato anche con il nome del palazzo/edificio o con indicazioni stradali prive di un nome specifico. Questo rendeva la lista difficile da completare in maniera esaustiva. Per questo motivo, abbiamo deciso di optare per una soluzione più semplice e immediata relegando il campo indirizzo come ultimo campo da riconoscere, quindi, ottenendolo come residuo sottraendo dalla stringa dell'indirizzo originario via via i componenti riconosciuti. E' più facile infatti riconoscere i campi che sono stati già in precedenza racchiusi in una lista standardizzata e di facile reperimento, come può essere l'elenco dei comuni italiani o delle province.

Il **campo Cap** altro non è che una stringa numerica composta da cinque cifre, nell'intervallo 00010 – 97100.

Questa sua natura di facile identificazione ci ha permesso di riconoscerlo con estrema immediatezza, senza crearci particolari problemi. Quindi, laddove troviamo un numero composto da cinque cifre che cade all'interno dell'intervallo specificato, possiamo ragionevolmente supporre che non sia un numero civico o l'informazione di una via o parte del nome di uno stabilimento, ma che sia proprio il campo Cap che stavamo cercando.

Per semplificare il riconoscimento del **campo Provincia**, si è deciso di utilizzare la sigla della provincia anziché la sua forma estesa. Questo perché dopo un'attenta analisi delle stringhe d'indirizzo, ci siamo resi conto che l'informazione relativa alla sigla era la più diffusa. Laddove invece era presente la forma per esteso priva della sigla, il campo veniva lasciato momentaneamente vuoto e si procedeva con il riconoscimento degli altri campi. In questo modo, una volta trovato, è il campo Comune che provvede a colmare la lacuna precedente, in quanto nella tabella comuni (dbo_COMUNI) compreso nel database a nostra disposizione, c'è sempre l'identificazione della provincia di appartenenza in sigla.

Un ulteriore piccolo ostacolo è stato riscontrato nelle parentesi tonde, infatti alcune sigle di province si trovano scritte dentro parentesi e altre no. Per questo abbiamo deciso di togliere tutte le parentesi tonde prima di analizzare i dati della stringa indirizzo.

Come controllo finale, viene fatta la comparazione della sigla trovata, con quelle presenti nella tabella province (dbo_PROVINCE), in modo da essere sicuri che la sigla sia effettivamente quella di una provincia esistente.

Il **campo Comune** è una stringa che può essere formata da una o più parole. Inizialmente per il riconoscimento si è usato un database contenente la lista di tutti i comuni d'Italia all'interno della tabella (dbo_COMUNI), scritti per intero e senza abbreviazioni. Questo però ha causato delle problematiche legate al riconoscimento del campo, perché il database si è rilevato insufficiente.

Infatti, è possibile riscontrare diversi nomi per indicare lo stesso comune, così creando un'incongruenza tra il nome scritto nella stringa e quello contenuto nel database, che ne impediva dunque il riconoscimento.

Quindi, i problemi più rilevanti, sono stati riscontrati nella sintassi del nome, ovvero, nella varietà di modi in cui il nome poteva essere scritto.

Le parole abbreviate, all'interno dell'indirizzo non ci permettevano di avere un riscontro positivo nella comparazione. Ad esempio, la parola *S.Giusto* all'interno di una stringa non veniva trovata perché nel database il nome del comune era stato memorizzato per esteso ovvero *San Giusto*.

Come questo, tanti altri casi di nomi di comuni scritti in maniera non estesa hanno impedito il riconoscimento del campo, proprio a causa di una mancata comparazione di esito positivo.

Oltre alle abbreviazioni, sono stati trovati altri casi in cui avveniva il suddetto problema a causa di una nomenclatura differente, ovvero alterazioni date da crasi. Come per esempio:

- Trezzo sull'Adda si può trovare scritto anche come Trezzo d'Adda
- Colle di val d'Elsa si può trovare scritto anche come Colle val d'Elsa o Colle d'Elsa.

Per ovviare a questo e ad altri casi riscontrati, si è deciso di ampliare il database di partenza con una nuova tabella (dbo_ALIAS) contenente tutti gli alias del nome di un comune, ovvero il maggior numero di modi in cui si poteva trovare scritto. Per fare ciò, si è usato un algoritmo che genera tutte le possibili combinazioni di parole che compongono il nome di un comune, in modo da ottenere risultati positivi nella comparazione e quindi di avere maggior successo nel riconoscimento del campo. (Per la descrizione dell'algoritmo vedi paragrafo 4.4).

Un ulteriore problema è stato riscontrato nei nomi di comuni composti da nomi di altri comuni, come il comune *Villafranca d'Asti* che contiene nella sua nomenclatura il nome di un altro comune *Asti*.

Il sistema, come già detto in precedenza, analizza la stringa dalla fine all'inizio, questo vuol dire, che nel nostro caso, il sistema legge nella stringa prima il nome *Asti*, che essendo un comune potrebbe essere identificato immediatamente come tale, ma dando un risultato errato.

Abbiamo imposto perciò al sistema di procedere nell'analisi della stringa fino a trovare nuove parole, nel caso del nostro esempio, *d'* e *Villafranca*, questi nuovi elementi fanno si che il sistema riconosca il dato come nuovo comune, questa volta però identificandolo in maniera corretta.

Altro problema è quando all'interno dell'informazione della via compare il nome di un comune che va quindi etichettato come campo Indirizzo e non come campo Comune. In *via Roma* per esempio, troviamo l'elemento *via* che identifica il campo Indirizzo ma con il nome *Roma* che è presente nella tabella dei comuni del database, e che può quindi essere etichettato anche come campo Comune.

Per ovviare a questo problema, un po' come si è fatto per i nomi composti dei comuni, si è chiesto al sistema di analizzare le parole precedenti al nome trovato per evitare equivoci. Essendo *Roma* nel nostro caso anche il nome di un comune, il sistema, prima di identificarlo come tale, e quindi prima di interrogare il database, prosegue nella lettura della stringa e, qualora trovasse nella posizione immediatamente precedente, elementi che identificano il campo Indirizzo, allora il sistema riconosce il nome identificandolo come campo Indirizzo e non commette l'errore di identificarlo come campo Comune".

Per ottimizzare l'identificazione del comune, si è scelto di identificare prima la provincia e, una volta individuata, di procedere con la ricerca del comune, confrontandola solo con i nomi dei comuni associati alla provincia trovata. Questo ottimizza il sistema restringendo la ricerca ed evitando di interrogare l'intera tabella dei comuni presente nel database, che contiene più di 8000 voci.

In questo modo, si ha maggior efficienza in termini di tempo di elaborazione ed un controllo aggiuntivo sulla ricerca del comune, assegnandolo alla sua reale provincia, ed evitando equivoci di identificazione.

Un ultima problematica è stata riscontrata nei casi in cui, nella stringa di riferimento, la provincia si trovava scritta nella sua forma estesa, per esempio :

- Firenze anziché FI

Il nome *Firenze* è riscontrabile all'interno del database nella tabella dei comuni, ma per non commettere errori, prima dell'identificazione definitiva il sistema continua nella lettura della stringa per verificare la presenza di un altro nome di comune. Qualora lo trovi, il sistema riconosce che il primo nome trovato, *Firenze*, è in realtà la provincia di appartenenza del comune trovato successivamente. Per esempio:

- via Bardini, 38 Empoli Firenze

Empoli viene identificato come effettivo comune e *Firenze* come provincia anche se scritta nella sua forma estesa, che verrà poi convertita dal sistema nella sua forma a sigla *FI*.

Il **campo Frazione** all'inizio dell'analisi del progetto non era considerato, quindi non presente nei risultati finali, perché non ritenuta un' informazione altamente rilevante. Così facendo però ci siamo accorti che, proprio perché il campo Indirizzo si ottiene per sottrazione degli altri campi, non avendo assegnato un campo specifico alla frazione, questa veniva inclusa nel campo Indirizzo creando confusione e alterando la correttezza del dato ottenuto. Si è scelto così di analizzare questo ulteriore campo ottenendo così un sistema di riconoscimento più preciso.

Quindi la scelta progettuale è ricaduta nell'aggiungere un ulteriore tabella all'interno database (dbo_COMUNI_FRAZIONI) che contenesse tutte le frazioni del territorio italiano, più di 61000 voci. Ed è dentro questa tabella che viene ricercata una possibile corrispondenza per il riconoscimento del campo Frazione.

Esistono, come per i comuni, possibilità diverse per chiamare la stessa frazione, nonostante ciò non si è ritenuto necessario creare un database alias con tutte le possibili alternative al nome frazione sia per l'eccessivo numero di possibilità che per la consapevolezza che, nonostante ciò, la frazione rimane un dato di scarsa rilevanza.

Capitolo 4

Implementazione

Come già accennato in precedenza, abbiamo scelto Access in quanto un efficiente sistema di gestione database. Il visual basic è una conseguenza di questa scelta.

Il sistema è sviluppato su diciassette routine raffigurate nello schema sotto, possono essere raggruppate in due categorie, la prima rappresenta la funzionalità principale del sistema e la seconda le utility.

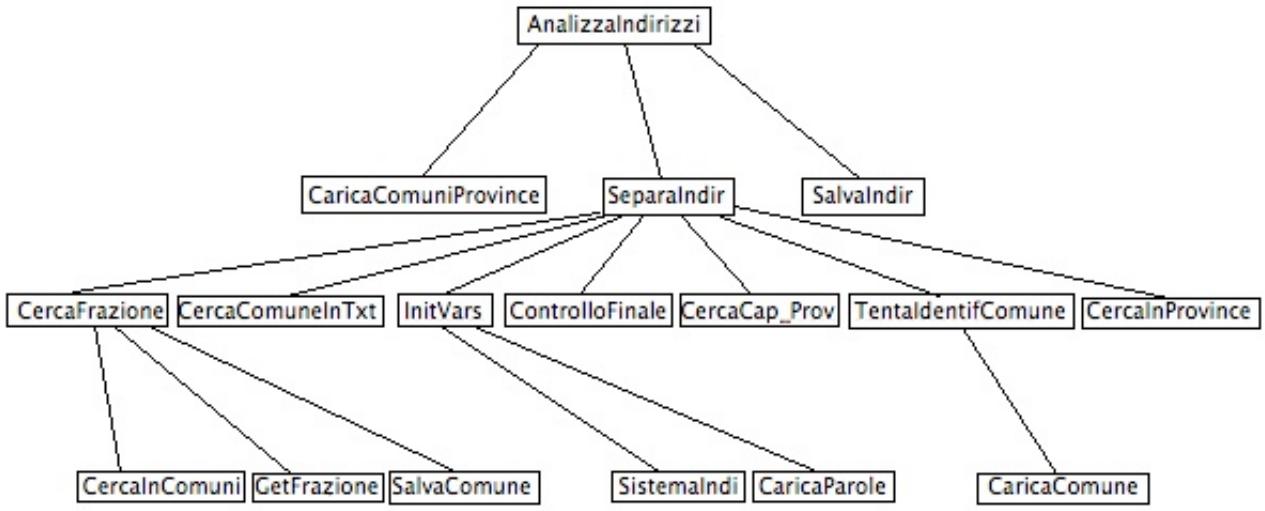

Figura 4.1. Diagramma generale di tutte le routine create, disposte nell'ordine di utilizzo.

4.1 Scelte di implementazione

La stringa indirizzo passata come input viene suddivisa in token di parole inserite all'interno di un array, questo per permettere al sistema l'analisi di ogni singolo elemento. Si è deciso di caricare in memoria locale, quindi dentro due matrici distinte, tutti i comuni e le province d'Italia, questo per evitare un eccesso di interrogazioni al database che rallenterebbero l'elaborazione del programma.

Per contenere le parole dell'indirizzo si è usato un array monodimensionale, invece per contenere tutta la lista dei comuni e delle province si sono utilizzati come già detto prima, due array multidimensionali ovvero due matrici.

La prima matrice ha dimensioni 15000x5, dove le colonne contengono tutte le informazioni riguardanti un generico comune, ovvero il cap, cod. istat, sigla provincia, nome comune.

La seconda matrice invece ha dimensioni 110x2 che contiene la lista di tutte le province suddivise in due colonne, una rappresenta la sigla l'altra la descrizione, ovvero il nome per intero.

4.2 Funzionalità principali del sistema

Le funzioni del sistema sono espresse principalmente dalle utility sotto indicate.
Si è fatto uso di una `blacklist`, ovvero una tabella contenente tutte le parole individuate all'interno della stringa che non appartengono all'informazione vera e propria dell'indirizzo ma servono a identificare il tipo di campo.
esempi:

-frazione
-località

in modo da ottenere all'interno del campo frazione, solo l'informazione "pulita" del nome della frazione.
Per ogni nome di un comune si è in grado di risalire alle informazioni mancanti nella stringa di riferimento, come il cap e la sigla della provincia, perché sono informazioni contenute e associate al nome all'interno della tabella comune.

4.3 Le Utility del sistema

La Routine CaricaComuniProvincie

Come intuibile dal nome, lo scopo di questa routine è quello di caricare in memoria tutte le province e comuni, in modo da avere i dati pronti all'uso, per fare questo vengono fatte delle interrogazioni al server in modo da farsi restituire tali dati, da questo momento in poi possiamo lavorare con i dati in locale senza andare a chiederli più al server.

La Routine CaricaParole

La funzionalità di questa routine è semplicemente quella di caricare dentro un array le parole che compongono l'indirizzo, che gli è stato passato come parametro al momento della chiamata avvenuta all'interno della routine InitVars.
Si esegue una scansione su tutti i caratteri della stringa per trovare un carattere separatore, che può essere uno spazio, una virgola o un trattino in modo da ottenere una divisione in parole.

La Routine SistemaIndi

Anche questa routine, come quella di CaricaParole, è stata chiamata dalla routine InitVars, ha come parametro l'indirizzo e il suo compito è quello di fare una prima standardizzazione su gli accenti delle lettere, sbarre e parole superflue presenti nella stringa da analizzare. Per riempire i campi dell'indirizzo con le informazioni trovate, non ci interessano informazioni superflue come ad esempio `loc.`, `frazione` e altre che vengono eliminate. Per ultimo si crea l'indicazione del numero civico standard con *`n.`*, quindi vengono sostituiti *`nr`*, *`numero`* con quello standard.

4.4 Algoritmi utilizzati

Principali algoritmi utilizzati e degni di nota.

• Divide et impera:

La tecnica detta "divide et impera" è una strategia generale per impostare algoritmi ed è stata usata all'interno del progetto per la ricerca del comune nella matrice.

Consideriamo un problema **P** e sia **n** la dimensione dei dati, la strategia consiste nel:

suddividere il problema in **k** sottoproblemi $\mathbf{P_i}$ di dimensione inferiore (ciascuno di dimensione $\mathbf{n_i}$) e successivamente riunire i risultati ottenuti dalle **k** soluzioni.

Se i **k** sottoproblemi sono "formalmente" simili al problema di partenza, si ottiene una scomposizione ricorsiva. Ci deve pertanto essere una dimensione **h** del problema che porti ad una risoluzione diretta, vale a dire che non necessiti della ricorsione.

Indichiamo con:

• **S** l'insieme dei dati

• **k** il numero dei sottoproblemi

• **h** la dimensione limite

Si può scrivere uno schema generale per la scomposizione.

algoritmo DIVETIMP (S, n)

se n < h

allora risolvere direttamente il problema P

altrimenti dividere S in k sottoinsiemi

risolvere separatamente i k sottoproblemi P_1, ..., P_k:

DIVETIMP(S_1,n_1), ... ,DIVETIMP(S_k,n_k)

riunire i risultati ottenuti.

Nel nostro caso specifico, è stato utilizzato per effettuare la ricerca del nome di un comune all'interno di una matrice che conteneva la lista di tutti i nomi comuni ordinati lessicograficamente. La procedura, a questo punto, consiste nel confrontare il nome del comune da ricercare con l'elemento allocato nella posizione centrale della matrice, Qui nel caso di mancata corrispondenza, si procede con un'ulteriore ricerca nel sottoinsieme, formato dagli elementi successivi o precedenti all'elemento centrale. Si cerca di capire cioè se, in questo caso, il nome del comune da cercare è superiore o inferiore, in senso lessicografico. Questa procedura viene ripetuta ricorsivamente fino a che non si trova il nome del comune ricercato, andando avanti ad ogni esito negativo riscontrato.

• <u>Combinazioni semplici (senza ripetizioni)</u>:

Tecnica usata per generare tutte le possibili combinazioni delle parole che compongono i nomi dei comuni, inseriti nella tabella degli "alias comuni" del database.

Si chiama combinazione semplice una presentazione di elementi di un insieme nella quale non ha importanza l'ordine dei componenti e non si può ripetere lo stesso elemento più volte. La collezione delle combinazioni di **k** elementi estratti da un insieme **S** di n oggetti distinti si può considerare ottenuta dalla collezione delle disposizioni semplici di lunghezza **k** degli elementi di **S** ripartendo tali sequenze nelle classi delle sequenze che presentano lo stesso sottoinsieme di **S** e scegliendo una sola sequenza da ciascuna di queste classi. Ciascuna delle suddette classi di sequenza di lunghezza **k** contiene **k!** sequenze, in quanto accanto a una sequenza **σ** si hanno tutte e sole quelle ottenibili permutando i componenti della **σ**. Quindi il numero delle combinazioni semplici di n elementi di lunghezza **k** si ottiene dividendo per **k!** il numero delle disposizioni semplici di n elementi di lunghezza **k**:

$$C_{n,k} = \frac{D_{n,k}}{P_k} = \frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Di solito tra le diverse disposizioni semplici di una classe si sceglie come combinazione rappresentativa la sequenza nella quale i componenti compaiono in ordine crescente (tutti gli insiemi finiti possono avere gli elementi ordinati totalmente, ovvero associati biunivocamente ai primi interi positivi).

PARTE II

Geolocalizzazione degli indirizzi

Capitolo 5

Analisi del progetto

Una volta conclusa la prima parte del progetto, concernente la standardizzazione di indirizzi, si procede alla seconda parte con la geolocalizzazione di tali indirizzi.

Lo scopo è quello di poterli visualizzare successivamente in una pagina web all'interno di una mappa, ma prima di fare ciò, si è scelto di effettuare un passaggio intermedio, ovvero quello di selezionare indirizzi in base a dei vincoli imposti, compito affidato ad un file fa noi creato. Questa selezione ci ha permesso di poter salvare, solo quelli corretti, all'interno del database degli "indirizzi salvati", ovvero quelli ritenuti sufficientemente adeguati per rappresentare un punto specifico nella mappa.

A tal proposito, sono stati scartati tutti gli indirizzi che non ricadevano all'interno del territorio italiano, e considerati non idonei tutti quelli che non rispettavano certe proprietà, come per esempio gli indirizzi incompleti, che al loro interno avevano un numero di informazioni insufficienti appunto per rappresentare con precisione un punto specifico sulla mappa. Tali indirizzi erano quelli che al loro interno avevano come informazione il solo nome di Comune, informazione ritenuta troppo generica per indicare il punto specifico, punto che per noi possa rappresentare un evento, come può essere la proiezione di un film in un determinato cinema, dove dire solo il comune in cui veniva proiettato è troppo generico. Nel nostro esempio, per essere corretto, il punto dovrebbe indicare anche la via per poter raggiungere il cinema e capire a quale cinema si sta facendo riferimento.

La scelta progettuale per la seconda parte del progetto è ricaduta sulla creazione di tre tipologie di file e, per testarne il funzionamento, ci siamo concentrati su gli indirizzi che si potessero riferire a degli eventi.

Tali file verranno discussi all'interno dei paragrafi corrispondenti, rispettando il seguente ordine di utilizzo:

- "GeorefV3.asp" descritto nel dettaglio nel paragrafo 5.1.1, dove si fa' una panoramica dei parametri utilizzati al fine di imporre i vincoli sui risultati ottenuti dalla geolocalizzazione.

- "Provincia.xml" descritto nel dettaglio nel paragrafo 5.1.2 dove si analizza la catalogazione di indirizzi geolocalizzati.

- "Mappa.html" descritto nel dettaglio nel paragrafo 5.2 che parla della visualizzazione di indirizzi geolocalizzati su una mappa.

Questa modalità di selezione e geolocalizzazione di indirizzi può essere applicata ad indirizzi generici, non solo a eventi. In base all'utilizzo che se ne vuole fare si possono quindi cambiare i vincoli di selezione modificandone i valori, rimane invece inalterata la visualizzazione su mappa, che avviene sempre nella stessa maniera a prescindere dai tipi di indirizzo che si usano.

5.1 Geolocalizzare un indirizzo mediante Google Maps

Facciamo un pò di chiarezza su due termini spesso usati per esprimere lo stesso significato.

- La **geolocalizzazione** è l'identificazione della posizione geografica nel mondo reale di un dato oggetto, come ad esempio un edificio, un telefono cellulare o un computer connesso o meno ad Internet, secondo varie possibili tecniche.

- Per **georeferenziazione** si intende invece l'attribuzione a un dato di un'informazione relativa alla sua dislocazione geografica; tale posizione è espressa in un particolare sistema geodetico di riferimento. La georeferenziazione è usata nei sistemi GIS, tanto da essere applicata sostanzialmente ad ogni elemento presente: pixel componenti un'immagine raster, elementi vettoriali come punti, linee o poligoni e persino annotazioni.

Per fare esempi più comuni e conosciuti, un sistema in cui gli elementi vengono georeferenziati è Google Maps, in cui è possibile cercare negozi o località di interesse dei quali vengono fornite non solo le tipiche informazioni che restituisce un motore di ricerca, ma viene evidenziato sulla mappa la posizione geografica ad essi riferita.

Georeferenziare una mappa significa attribuire alla mappa delle coordinate Nord/ Est, quindi, applicare al punto d'inizio dei 2 assi cartesiani, in basso a sinistra, il valore 0;0, e calcolare per ogni punto presente nella mappa le misure in metri anziché il numero di pixel.

Al momento il metodo più semplice e immediato per georeferenziare un indirizzo è quello di interrogare Google Maps (nome precedente Google Local), cioè un servizio accessibile dal relativo sito web che consente la ricerca e la visualizzazione di mappe geografiche di buona parte della Terra. Oltre a questo è possibile ricercare luoghi, trovare un possibile percorso stradale tra due punti e visualizzare foto satellitari di molte zone con diversi gradi di dettaglio. Le foto sono statiche (non in tempo reale), buona parte delle quali sono riferite alla fine degli anni novanta.

Google Maps ha una vasta gamma di API che consentono di incorporare la funzionalità e l'utilità quotidiana di Google Maps nel proprio sito web e nelle applicazioni, e sovrapporre i propri dati su di loro.

La famiglia delle API è composta da:

- **Maps API JavaScript**: Si usano quando si vuole incorporare una mappa di Google nella propria pagina web usando JavaScript. Manipolare la mappa e aggiungere contenuti attraverso molteplici servizi.

- **Maps API per Flash**: Questa API ActionScript si utilizza per incorporare una mappa di Google nelle applicazioni o pagine Web basate su Flash. Si può manipolare la mappa in tre dimensioni e aggiungere contenuti attraverso molti servizi.

- **Google Earth API**: Per incorporare un vero e proprio mondo digitale 3D nella propria pagina web. Per diventare visitatori in qualsiasi punto della Terra (anche sotto l'oceano) senza lasciare la propria pagina web.

- **Maps Image API**: Per incorporare una veloce e semplice immagine di Google Maps o Via panorama View nella tua pagina web o sito per cellulari senza la necessità di JavaScript o di caricamento delle pagine dinamiche.

- **Web Service**: Usare le richieste di URL per accedere al geocoding, direzione, altitudine, informazioni e luoghi da applicazioni client e manipolare i risultati in JSON o XML.

Per il nostro progetto si sono utilizzate le prime API (**Maps API JavaScript**), perché nell'eseguire i vari test, avevamo bisogno di visualizzare i risultati su una pagina web dinamica, in modo da reperirli e usarli più facilmente.

Tornando al nostro utilizzo, possiamo quindi ottenere le coordinate di un indirizzo usando le API di Google Maps. Sono state fatte alcune prove al riguardo, per capirne il corretto funzionamento e creati due file che ne fanno uso. Il primo, che esegue solo la geolocalizzazione dell'indirizzo e ritorna dei parametri che verranno usati per effettuare dei controlli sulla precisione del punto trovato, ovvero la possibilità di imporre dei vincoli sui risultati ottenuti, file di nome "GeorefV3.asp". Il secondo file invece di nome "Mappa.html" che, oltre la possibilità di georeferenziare un indirizzo, visualizza su una mappa gli indirizzi, e implementa altre funzionalità descritte in dettaglio nel paragrafo (5.2.1).

5.1.1 Parametri usati

Il primo file creato è stato quello per la geolocalizzazione di un indirizzo, reso possibile dalle interrogazioni verso il server di Google. Lo scopo è quello di far tornare dei risultati sotto forma di parametri.

L'implementazione finale è avvenuta all'interno di un file con estensione .asp, per ottenere una pagine web dinamica, in modo che i contenuti della pagina varino in funzione delle informazioni passate dall'utente come input.

Si deve poter utilizzare il valore del parametro che ho ottenuto con l'interrogazione a Google per effettuare una comparazione su di esso, e stabilire se va bene o no, ovvero rispetta i vincoli.

Questi parametri sono stati voluti per permettere di selezionare tra tutti, solo gli indirizzi che rispettassero certi vincoli imposti.

Nel nostro caso si è scelto di selezionare solo gli indirizzi che rispettassero tali vincoli:

- indirizzi riferiti esclusivamente a punti appartenenti al territorio italiano, in modo da poter scartare tutti gli indirizzi esteri

- indirizzi che rispettassero un certo grado di precisione, ovvero quelli che presentavano tra i campi dell'indirizzo, informazioni sufficienti a localizzare il punto in maniera esaustiva, quindi scartare gli indirizzi dove era indicato solo il nome del comune, ritenuta un' informazione troppo generica.

I risultati ottenuti dalla geolocalizzazione di un indirizzo vengono visualizzati in una pagina web statica con codice HTML, dati da un'elaborazione del codice dinamico presente sul server.

All'interno della pagina ASP, sono presenti istruzioni di scripting. Uno script è un codice di programma inserito in un file HTML che viene interpretato ed eseguito. Nel nostro caso si è fatto uso di uno script lato server, il linguaggio di programmazione usato per la creazione della pagina ASP in questione è VBScript, mentre JScript è stato utilizzato per le API di Google e ottenere le informazioni sulla geolocalizzazione dell'indirizzo, tali dati vengono poi restituiti sotto forma di pagina web HTML a chi ne ha fatto richiesta. Tutto questo per ottenere dei parametri da analizzare e applicare i vincoli sopra descritti.

I passaggi per il funzionamento del file asp sono i seguenti:

1. Il client richiede una pagina al server web (trewstudio)
2. Il server web (trewstudio) elabora il codice interno alla pagina (interrogando a sua volta il server di Google) e sostituisce il codice sorgente ASP con codice HTML
3. Il server web (trewstudio) invia la pagina web al client sotto forma di

   documento HTML
4. Il browser del client interpreta e visualizza il documento HTML ricevuto.

Nell'esempio che segue viene riportato un possibile utilizzo.

Esempio:

Inserendo nella url di un browser:
http://www.trewstudio.it/wup/francesco/georefv3-1.asp?ind=via+emilio+bardini+48+empoli

dove `http://www.trewstudio.it/wup/francesco/georefv3-1.asp?` sta ad indicare la richiesta che esegue il browser per contattare il server web (trewstudio) che elabora la pagina ASP di nome `georefv3.asp` allocata nel server, in funzione delle informazioni passate in input dalla url sotto forma di argomenti, inseriti subito dopo il carattere *?*. Tali argomenti sono separati dal segno *+* per motivi legati alla sintassi web.

Si ottengono così i seguenti parametri, visibili nell'immagine sottostante.

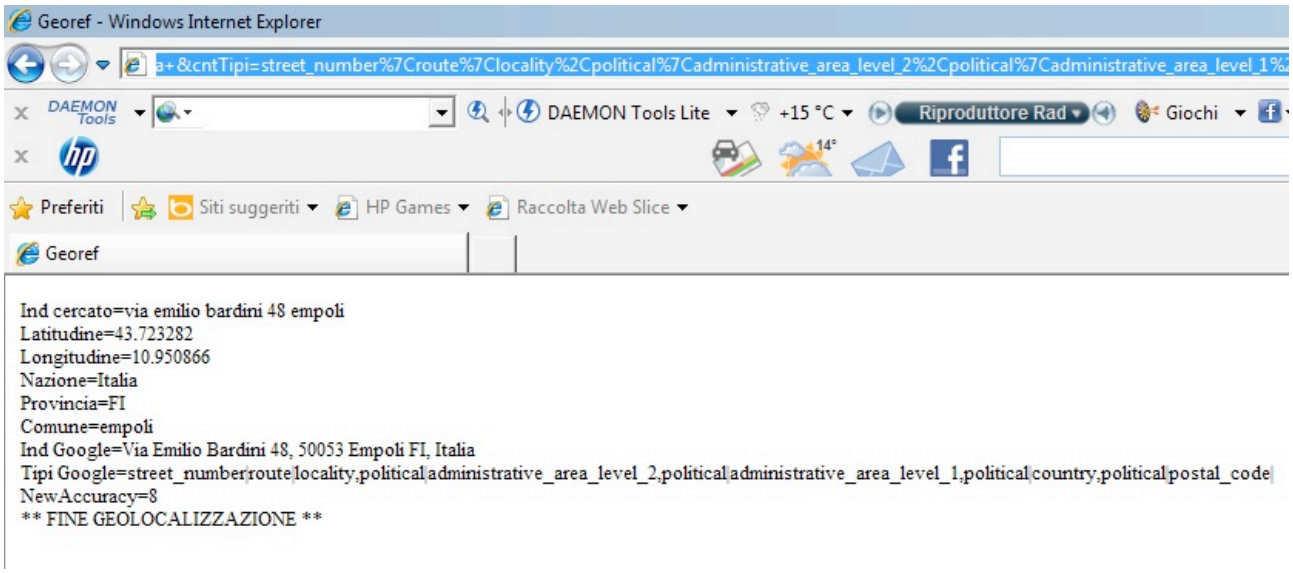

Figura 5.1: Screenshot dei risultati ottenuti con la geolocalizzazione (file .asp)

Di seguito vengono descritti i valori dei parametri restituiti dalla geolocalizzazione:

- **indirizzo cercato**: indica l'indirizzo inserito nella url come riga di comando da passare a google per la geolocalizzazione
- **latitudine**: indica la coordinata di latitudine
- **longitudine**: indica la coordinata di longitudine
- **nazione**: indica il nome della nazione

- **provincia**: indica il nome della provincia in sigla
- **comune**: indica il nome del comune
- **indirizzo di google:** indica l'indirizzo che restituisce google
- **tipi di google**: indica i tipi che restituisce l'interrogazione a google (vedi paragrafo 6.1)
- **newAccuracy**: indica l'indice di accuratezza dell'indirizzo trovato (vedi paragrafo 7.1)

Le scelte imposte per definire un indirizzo corretto, hanno fatto si che i valori accettabili dal sistema dovevano rispettare i due seguenti vincoli:

- **nazione**=Italia

- **newAccuracy** > 3

I risultati ottenuti attraverso la geolocalizzazione ed espressi sotto forma di parametri, ci hanno permesso di valutare i risultati in modo da poter scegliere gli indirizzi idonei. Sono stati selezionati così solo gli indirizzi con il parametro nazione uguale a "Italia", e quelli che con l'indice di accuratezza (newAccuracy) maggiore di 3, scartando quelli con un valore inferiore o uguale in quanto le informazioni risultavano insufficienti per poterlo memorizzare come indirizzo idoneo. Per esempio il solo indirizzo con "Firenze", dunque solo il nome del comune non bastava come informazione per stabilire un indirizzo preciso.

Così facendo è stato possibile scartare tutti gli indirizzi che non rispettavano i vincoli, con la conseguenza di non essere aggiunti nel database degli "indirizzi salvati".

Ma, dove vengono memorizzati tutti gli indirizzi risultati idonei? All'interno del database che contiene tutti gli eventi o indirizzi, più precisamente all'interno di file con estensione xml.

5.1.2 Catalogare gli indirizzi geolocalizzati

Una volta effettuata la geolocalizzazione di indirizzi e selezionati quelli idonei, possiamo memorizzare le informazioni ottenute per crearci un proprio database contenente il maggior numero di informazioni su indirizzi, che magari in un futuro, potranno sostituire la necessita di interrogare Google per ottenere dati sulla geolocalizzazione. Quindi creare la possibilità di trovare di un indirizzo ricercato, all'interno del nostro database locale. Per fare questo si sono utilizzati dei file con estensione XML in modo da immagazzinare tali informazioni ed essere reperiti quando servono. Nel nostro caso tali file hanno il compito di salvare indirizzi di eventi, suddivisi per provincia di appartenenza.

Le informazioni salvate erano, oltre ai dati ottenuti dalla geolocalizzazione quindi latitudine, longitudine e tutti i componenti di un indirizzo, anche il nome dell'evento. Tutte informazioni utilizzate per rappresentare gli eventi sulla mappa.

La scelta dunque è ricaduta su XML, un meta-linguaggio per definire la struttura di documenti e dati, ed è utilizzato come contenitore di informazioni. Analizziamo ora XML dal punto di vista logico e sintattico, con i documenti che con esso si possono creare, dando anche uno sguardo alla struttura logica. Un documento XML è intrinsecamente caratterizzato da una struttura gerarchica. Esso è composto da componenti denominati elementi. Ciascun elemento rappresenta un componente logico del documento e può contenere altri elementi (sottoelementi) o del testo.

L'organizzazione degli elementi segue un ordine gerarchico o arboreo che prevede un elemento principale, chiamato root element o semplicemente root o radice. La radice contiene l'insieme degli altri elementi del documento. Possiamo rappresentare graficamente la struttura di un documento XML tramite un albero, generalmente noto come document tree. Prendiamo in considerazione la rappresentazione del documento creato per immagazzinare i dati degli indirizzi geolocalizzati, secondo il modello XML, come mostrato in figura.

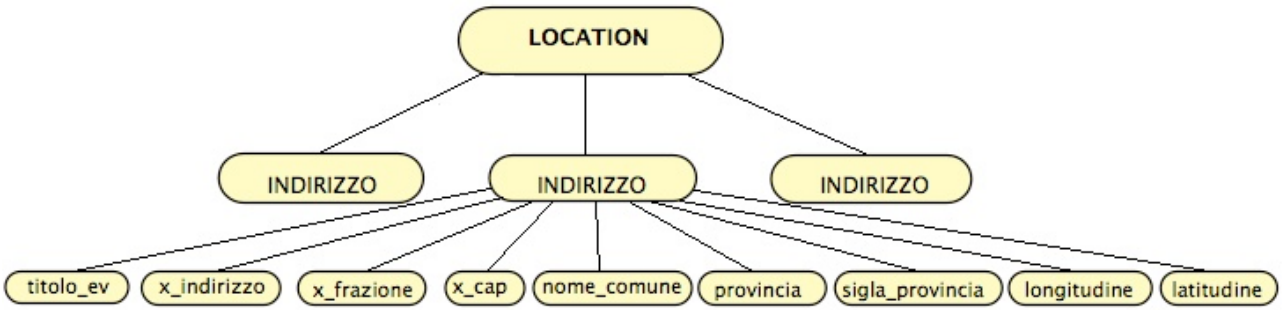

Figura 5.2. Struttura gerarchica del documento XML

Nella figura abbiamo un root element denominato LOCATION che contiene una lista di elementi che rappresentano i vari indirizzi. Ciascun indirizzo a sua volta contiene il titolo dell'evento, l'informazione indirizzo, la frazione, il cap, il nome comune, la provincia la sigla della provincia, longitudine e latitudine. La struttura logica del documento XML dipende dalle scelte progettuali. La struttura logica del documento XML è stata tradotta in una corrispondente struttura fisica composta di elementi sintattici chiamati tag (cioè sequenze di caratteri delimitate dai segni '<' e '>'). Questa struttura fisica è stata implementata tramite un file di testo, uno per ogni provincia.

La rappresentazione fisica del documento XML visto prima, è la seguente:

```
<?xml version="1.0"?>
<LOCATIONS>
  <INDIRIZZO>
    <TITOLO_EV>Corpo celeste</TITOLO_EV>
    <x_indirizzo>Piazza S. Vito 7</x_indirizzo>
    <x_frazione></x_frazione>
    <x_cap>92019</x_cap>
    <NOME_COMUNE>Sciacca</NOME_COMUNE>
    <NOME_PROVINCIA>Agrigento</NOME_PROVINCIA>
    <SIGLA_PROVINCIA>AG</SIGLA_PROVINCIA>
    <LONGITUDINE>13.0800869</LONGITUDINE>
    <LATITUDINE>37.5095244</LATITUDINE>
  </INDIRIZZO>
    .
    .
    .
</LOCATIONS>
```

Esempio di frammento del file “AG.xml” che rappresenta il file contenente tutti gli eventi della provincia di Agrigento.

Alcuni elementi possono essere vuoti, come lo è la frazione nel caso sopra indicato, cioè possono essere privi di contenuto testuale, perché tale informazione non è presente.

5.2 Visualizzare su una mappa gli indirizzi geolocalizzati

Il secondo file che è stato creato principalmente per visualizzare gli indirizzi su una mappa è un file html, le sue funzionalità sono quelle di riuscire a ricavare da un indirizzo le coordinate Latitudine e Longitudine, eseguendo anche il processo inverso ovvero quello di ricavarsi dalle coordinate l’indirizzo. Nel paragrafo successivo vengono descritte tutte le funzionalità in maniera dettagliata.

5.2.1 Funzionalità dei servizi

Adesso parliamo dei risultati ottenuti per la visualizzazione della geolocalizzazione di indirizzi. Un esempio di possibile utilizzo della geolocalizzazione di un indirizzo è quello di rappresentare degli eventi su un territorio. A questo scopo è stato creato un file di prova in html per testarne il funzionamento, implementare alcune delle funzioni di Google Maps, calcolare il percorso tra due punti, caricare

da un file xml una lista di indirizzi e visualizzarli sulla mappa con dei Marker (segnaposto).

Il file xml oltre alla lista degli indirizzi, contiene informazioni sugli eventi associati. Gli indirizzi visualizzati sulla mappa possono quindi essere caricati da un file xml, oppure inseriti manualmente come input, direttamente nella pagina web, specificando le coordinate o inserendo l'indirizzo per intero. Inoltre si è implementata la possibilità di aggiungere nuovi eventi e di calcolarne il percorso.

Di seguito vengono elencate le funzionalità sviluppate:

- Tramite le coordinate geografiche (latitudine, longitudine) trovare un punto sulla mappa

- Tramite un indirizzo trovare il punto su una mappa

- Associare ad un punto trovato il nome di un evento con la sua relativa descrizione, per esempio: cinema astro titolo film "immortale" ore 18:00 e aggiungerlo nel file xml della provincia di appartenenza del punto trovato.

- La possibilità di spostare il marker (il punto dell'indirizzo indicato sulla mappa) con differenti tipi di precisione (1,2,3) per indicare con perfezione il punto spesso non centrato perfettamente dalla geolocalizzazione di Google, o per mancanza di un numero civico, il punto va spostato a mano nel punto desiderato, come può essere all'inizio, a meta o alla fine di una strada. Altro esempio, all'interno di un campo dove non sono presenti indicazioni sulla via.

- La possibilità di calcolare il percorso fra due punti, partendo dal punto di partenza che può essere ricavato con l'inserimento manuale, o attraverso la geolocalizzazione tramite GPS se usato con un mobile. Possiamo poi selezionare l'evento scelto tra quelli presenti sulla mappa caricati in precedenza dal file xml con la provincia specificata

- Effettuare un click sulla mappa per farsi restituire le coordinate del punto indicato

- Eliminare un elemento appena creato o uno già presente, selezionandolo dal marker

- Visualizzazione di una nuvoletta una volta cliccato su un marker che rappresenta un evento, all'interno della quale compare la descrizione con l'aggiunta di informazioni riguardanti l'orario, il prezzo o il titolo dell'evento

- Pulisci mappa che cancella a video tutti gli eventi che erano comparsi specificando una provincia

http://www.trewstudio.it/wup/francesco/mappa.html

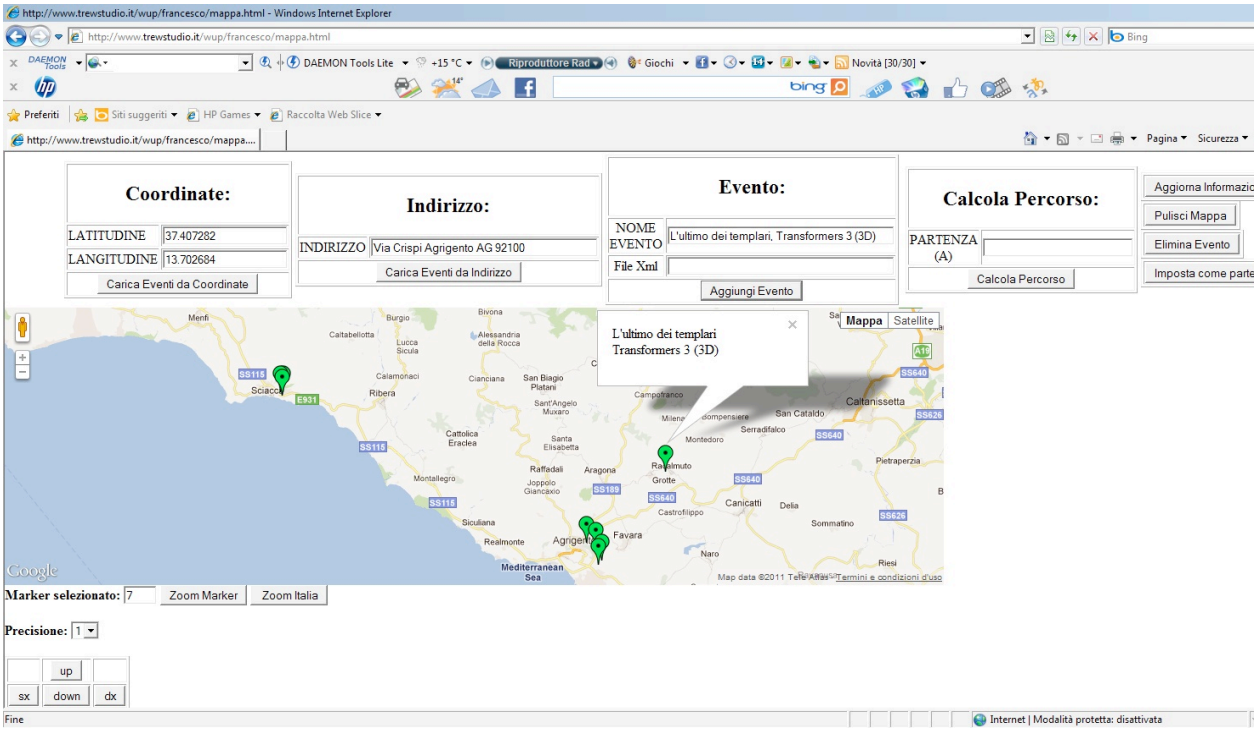

Figura 5.3. Screenshot della pagina web creata per la gestione di eventi

Attraverso questo screenshot rappresentato in figura, si possono notare i risultati ottenuti sulla visualizzazione di eventi su una mappa, eventi contrassegnati da un marker di colore verde che indica il punto preciso dell'evento. Tali punti vengono creati in base alle informazioni presenti nel file xml scelto, nel nostro caso dal file `AG.xml` che contiene gli eventi della provincia di Agrigento.

Capitolo 6

Le tecnologie utilizzate

Come già introdotto nei capitoli precedenti, lo sviluppo di questo progetto ha richiesto l'utilizzo di alcune tecnologie che andremo ad analizzare adesso nei seguenti paragrafi:

6.1 dove viene descritto l'utilizzo fatto delle Google Geocoding API all'interno del progetto;

6.2 che presenta le rappresentazioni DOM e XML per la catalogazione di indirizzi;

6.3 dove viene analizzato l'HTML, linguaggio di programmazione usato per la creazione di pagine web;

6.4 dove viene descritto javascript, il linguaggio di programmazione per gestire gli script all'interno dei file html e asp;

6.5 dove viene riportata la definizione di ASP utilizzato per la creazione di pagine web dinamiche.

6.1 Google Geocoding API

Il geocoding è il processo di conversione da indirizzo (ad esempio "Via Emilio Bardini, 48 Empoli FI 50053") a coordinate geografiche (ad esempio latitudine 43.723282 e longitudine 10.950866), le quali possono essere usate per posizionarsi in una mappa. Il Google Geocoding API fornisce servizi di geocoding tramite una richiesta HTTP; questo servizio permette anche di compiere l'operazione inversa, da coordinate ad indirizzo, conosciuta come reverse geocoding. Una descrizione sull'implementazione di queste funzioni si trova nel paragrafo (7.2).

Figura 6.1: Google Geocoding API - Web services

I risultati ottenuti con la geolocalizzazione (vedi paragrafo 5.1.1) in Asp.net, e le notevoli informazioni aggiuntive, qui successivamente presentate, hanno portato alla creazione di tecniche Map Based “intelligenti” capaci di selezionare indirizzi che rispettassero certi vincoli per essere considerati idonei.
Di seguito vengono riportate le informazioni a cui si è fatto riferimento per effettuare la selezione e implementare il nuovo indice di accuratezza. Tali informazioni vengono definite come tipi di Google, contenuti nei risultati restituiti dalla geolocalizzazione.

- **street address** indica un indirizzo preciso di una strada.
- **route** indica il nome di un percorso come route US 101 (usato negli USA).
- **intersection** indica un importante incrocio, di solito di due strade principali.
- **political** indica un’ entità politica, un’ area di un’ amministrazione civile.
- **country** indica la nazione.
- **administrative area level 1** indica il primo ordine dell’entità civile al di sotto del livello nazionale.
- **administrative area level 2** indica il secondo ordine dell’entità civile al di sotto del livello nazionale.
- **administrative area level 3** indica il terzo ordine dell’entità civile al di sotto del livello nazionale.
- **colloquial area** indica un nome alternativo per l’enitià civile.
- **locality** indica la città o la provincia.
- **sublocality** indica il primo ordine dell’entità civile al di sotto della provincia.
- **neighborhood** indica la frazione.
- **premise** indica il nome di una posizione, al di solito un quartiere o un insieme di edifici con un nome comune.
- **subpremise** indica il nome di una posizione, denomina di solito un edificio singolare all’interno di un insieme di edifici con un nome comune.

- **postal code** indica il codice postale usato dalla posta all'interno del paese.
- **natural feature** indica una caratteristica naturale.
- **airport** indica un aereoporto.
- **park** indica il nome di un parco.
- **point of interest** indica punti di interesse, punti importanti di enti locali, come per esempio la Statua della Libertà.
- **post box** indica una specifica casella postale.
- **street number** indica il numero civico di una strada.
- **floor** indica il piano di un edificio.
- **room** indica la stanza di un edificio.

Come indicato e riportato nel paragrafo 7.1, la tecnica dell'indice di accuratezza potrebbe essere interpretata in vari modi differenti. La più gettonata sarebbe quella per cui il sistema, per quanto riguarda i dati forniti da Google Geocoding API, analizza i parametri sopra indicati per valutare la precisione dell'indirizzo.

6.2 Dom e Xml

Il Document Object Model (DOM), letteralmente modello a oggetti del documento,è una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti, è indipendente dalla piattaforma e dal linguaggio.

I principali formati gestiti sono HTML, XHTML e XML. Gli aspetti di DOM, ovvero i suoi elementi "Elements", possono essere manipolati con la sintassi del linguaggio di programmazione in uso.

DOM è lo standard ufficiale del W3C per la rappresentazione di documenti strutturati in maniera da essere neutrali sia per la lingua che per la piattaforma.

DOM è inoltre la base per una vasta gamma delle interfacce di programmazione delle applicazioni; alcune di esse sono standardizzate dal W3C. Il suo utilizzo in questo progetto ha riguardato l'estrapolazione di dati derivanti dagli eventi per poi passare le coordinate a Google Geocoding API (vedi paragrafo 5.2.1).

6.3 Html

In informatica l'HyperText Markup Language (HTML) (traduzione letterale: linguaggio di descrizione per ipertesti) è il linguaggio solitamente usato per i documenti ipertestuali disponibili nel World Wide Web. In tali documenti, un tratto di testo può essere contrassegnato inserendo delle etichette, tag, che ne descrivono la funzione, il colore, il link, o altre caratteristiche. Il contenuto servito dai siti web in seguito a una richiesta dell'utente solitamente consiste di un documento HTML: un web browser scarica da uno o più web server il contenuto HTML ed eventuali documenti collegati e li elabora, ossia ne interpreta il codice, al

fine di generare la visualizzazione della pagina desiderata sullo schermo del computer.

L'HTML non è un linguaggio di programmazione (in quanto non prevede alcuna definizione di variabili, strutture dati, funzioni, strutture di controllo) ma solamente un linguaggio di markup che descrive le modalità di impaginazione, formattazione o visualizzazione grafica (layout) del contenuto, testuale e non, di una pagina web attraverso tag di formattazione. Tuttavia, l'HTML supporta l'inserimento di script come il JavaScript usato in questo progetto.

6.4 JavaScript

JavaScript è un linguaggio di programmazione interpretato e leggero, che dispone di funzionalità orientate agli oggetti. Il nucleo del linguaggio è stato incorporato all'interno di Internet Explorer e degli atri browser Web più diffusi. Inoltre è stato arricchito con funzionalità che lo hanno reso adatto all'editoria Web: sono stati aggiunti oggetti rappresentativi della finestra di navigazione e del suo contenuto. La versione di JavaScript lato client permette l'inserimento di contenuti eseguibili all'interno di pagine Web; così le pagine Web non sono più pagine HTML statiche, ma possono comprendere programmi che interagiscono con l'utente, controllando il browser e creando dinamicamente nuovi contenuti HTML.

Sintatticamente, il linguaggio di base JavaScript somiglia al C, al C++ e a Java. Dispone di costrutti di programmazione come l'istruzione if, il ciclo while e l'operatore &&. Le somiglianze, purtroppo, si fermano qui. JavaScript è un linguaggio debolmente tipizzato: le variabili non devono necessariamente essere dichiarate con un tipo. Gli oggetti in JavaScript assomigliano più agli array associativi del Perl che alle strutture del C o ali oggetti del C++ e Java. Come il Perl, anche JavaScript è un linguaggio interpretato; prende spunto dal Perl anche per altri versi come, ad esempio, l'uso di espressioni regolari e le modalità di gestione degli array. Utilizzato si nella creazione della pagina web in HTML che per il file con estensione asp.

6.5 Asp

In informatica, le Active Server Pages (Pagine Server Attive, in genere abbreviato in ASP) sono pagine web contenenti, oltre al puro codice HTML, degli script che verranno eseguiti dal server per generare runtime il codice HTML da inviare al browser dell'utente (proprio per questo vengono in genere definite pagine web dinamiche). In questo modo è possibile mostrare contenuti dinamici (ad esempio estratti da database che risiedono sul server web) e modificarne l'aspetto secondo le regole programmate negli script, il tutto senza dover inviare il codice del programma all'utente finale (al quale va inviato solo il risultato), con un notevole risparmio di tempi e banda.

I linguaggi utilizzati sono VBScript e JScript per l'ambiente ASP. Nel progetto è stato utilizzato il JScript.

Capitolo 7

Implementazione

Come già accennato in precedenza, la scelta del linguaggio di programmazione per lo sviluppo della seconda parte del progetto è ricaduta su Javascript, per motivi legati alla scrittura del file GeorefV3 con estensione asp mentre l'uso dell'html, per la visualizzazione della pagina web che gestisce eventi su una mappa. Nei paragrafi che seguono sono stati descritti le implementazioni più rilevanti. Il paragrafo 7.1, che implementa la struttura dell'indice di accuratezza definita nel file GeorefV3.asp con i suoi relativi valori e, infine, il paragrafo 7.2 che fornisce una panoramica delle funzioni che hanno fornito l'interfacciamento con Google Geocoding API.

7.1 L'indice di accuratezza

L'indice di accuratezza, è stato implementato all'interno del file "GeorefV3.asp", tale indice nella vecchia versione di google api era un numero che oscillava da 1 a 10, e serviva per indicare la precisione della localizzazione di un indirizzo in base alle informazioni che si passavano, quindi inserire un numero civico all'interno di un indirizzo permetteva una maggiore precisione nel localizzare latitudine e longitudine. Dunque, avere un risultato più accurato, corrispondeva ad avere come indice di accuratezza un numero sempre più grande e più vicino alla soglia massima di precisione, ovvero 10. Nella nuova versione di google api V3 però questo parametro è stato omesso, per questo è nata la necessità di implementarne uno nuovo all'interno del file "GeorefV3.asp", di seguito sono riportati i valori attribuiti al nuovo indice di accuratezza, con relativa descrizione. Le scelte sono state prese in base all'uso che ne fa il progetto.

Il valore cresce in maniera direttamente proporzionale alla precisione.

ACCURACY=8 "SE HO VIA CON NUMERO CIVICO O NOME EDIFICIO"

ACCURACY=6 "SE HO LA VIA SENZA NUMERO CIVICO"

ACCURACY=5 "SE HO SOLO LA FRAZIONE"

ACCURACY=4 "SE HO SOLO IL COMUNE"

ACCURACY=3 "SE HO TROVATO UN INDIRIZZO SU UN COMUNE DIVERSO"

ACCURACY=1 "NON IN ITALIA"

ACCURACY=0 "GOOGLE NON TROVA L'INDIRIZZO"

7.2 Funzionalità usate per la geolocalizzazione

Le funzionalità messe a disposizione dalle API di Google per la geolocalizzazione sono molte, di seguito verrano elencate con una breve descrizione solo quelle principali di cui se ne sono fatte uso:

Caricamento di Google Maps API

Questo script di tag è necessario per l'utilizzo delle API di Google Maps.

```
<html>
  <head>
  <script type="text/javascript" src="http://maps.googleapis.com/maps/
api/js?sensor=false&language=it"></script>
```

L'URL contenuto nel tag "`script`" è la posizione di un file JavaScript che carica tutti i simboli e le definizioni necessarie per l'utilizzo di Google Maps API.

L'oggetto Mappa

```
var map = new google.maps.Map(document.getElementById("map_empoli"),
          myOptions);
```

La classe JavaScript che rappresenta una mappa è la classe "`Map`". Oggetti di questa classe definiscono una singola mappa su una pagina. (È possibile creare più di un'istanza di questa classe - ogni oggetto definirà una mappa separata sulla pagina.) Creiamo una nuova istanza di questa classe usando l'operatore "`new`" di JavaScript. Quando si crea una nuova istanza di mappa, si specifica un elemento HTML nella pagina come un contenitore per la mappa. Nodi HTML sono figli dell'oggetto "`document`" di JavaScript, e si ottiene un riferimento a questo elemento attraverso il metodo "`document.getElementById()`".

Questo codice definisce una variabile (denominata map), e assegna la variabile a una nuovo oggetto "Map", passando anche delle opzioni definite all'interno dell'oggetto "`mapOptions`". Queste opzioni saranno utilizzati per inizializzare le proprietà della mappa.

La funzione “Map()” è nota come costruttore e la sua definizione è la seguente:

Costruttore	Descrizione
Mappa(mapDiv: Node,opts?:MapOptions)	Crea una nuova mappa all'interno del contenitore di HTML dato – che in genere è un elemento DIV – utilizzando qualsiasi (opzionale) parametri che vengono passati.

Caricamento della mappa

```
<body onload= "initialize()">
```

Durante l’esecuzione di una pagina HTML, il Document Object Model (DOM) viene costruito, e le eventuali immagini esterne e gli script vengono ricevuti e incorporati nell’oggetto “document”. Al fine di garantire che la nostra mappa viene inserita nella pagina dopo che la pagina è completamente caricata, si limita a eseguire la funzione che costruisce la mappa una volta che l'oggetto <body> elemento della pagina HTML riceve un evento “onload”. In questo modo evita comportamenti imprevedibili e ci dà un maggiore controllo su come e quando la mappa viene visualizzata.

Il corpo del tag “onload” attributo è un esempio di un gestore di eventi. La Google Maps JavaScript API fornisce anche una serie di eventi che è possibile gestire per determinare i cambiamenti di stato.

Richieste di Geocoding

Geocoding è il processo di conversione degli indirizzi (come "Via Bardini, Empoli") in coordinate geografiche (come latitudine e longitudine 43.722963", "10.950923), che è possibile utilizzare per inserire i marcatori (market) o la posizione della mappa. La reverse geocoding è invece il processo inverso di conversione, ovvero da coordinate geografiche ad un formato indirizzo leggibile.

L’accesso al servizio di geolocalizzazione è asincrono, in quanto le API di Google Maps hanno bisogno di effettuare una chiamata a un server esterno. Per questo motivo, è necessario passare un metodo callback da eseguire al termine della richiesta. Questo metodo callback elabora il risultato. Si noti che il geocoder può restituire più di un risultato.

Si accede al servizio Google Maps API geocoding all'interno del codice tramite l’oggetto “google.maps.Geocoder”. Il metodo “Geocoder.geocode()” avvia una richiesta al servizio di geocodifica, passandogli un oggetto “GeocodeRequest”

contenente le condizioni di input e il metodo callback da eseguire al momento del ricevimento della risposta.

L'oggetto "GeocodeRequest" contiene i seguenti campi:

```
{
 address :  string,
 latLng :  LatLng,
 bounds :  LatLngBounds,
 region :  string
}
```

Dove address e latLng sono campi obbligatori. Si può passare sia un indirizzo o un latLng per la ricerca. (Se si passa un latLng, il geocoder esegue ciò che è noto come reverse geocode.

Risposte di Geocoding

Nel servizio di geolocalizzazione è necessario un metodo callback da eseguire al momento del recupero dei risultati del geocoder. Questa callback deve passare due parametri per contenere i risultati e lo stato del codice, in questo ordine.

Dal momento che il Geocoder può restituire più di una voce, l'oggetto "GeocoderResults" è un array.

L'oggetto "GeocoderResults" rappresenta un singolo risultato Geocoding ed è un oggetto della forma seguente:

```
results[]: {
 types[]: string,
 formatted_address: string,
 address_components[]: {
   short_name: string,
   long_name: string,
   types[]: string
 },
 geometry: {
   location: LatLng,
   location_type: GeocoderLocationType
   viewport: LatLngBounds,
   bounds: LatLngBounds
 }
}
```

Lo stato del codice può restituire uno dei seguenti valori:

- `google.maps.GeocoderStatus.OK` indica che il geocode ha avuto successo;
- `google.maps.GeocoderStatus.ZERO_RESULTS` indica che il geocode ha avuto successo, ma non ha prodotto risultati. Questo può verificarsi se al geocode è stato passato un indirizzo inesistente o un latng in una locazione remota;
- `google.maps.GeocoderStatus.OVER_QUERY_LIMIT` indica che si è sopra la vostra quota;
- `google.maps.GeocoderStatus.REQUEST_DENIED` indica che la richiesta è stata negata per qualche motivo;
- `google.maps.GeocoderStatus.INVALID_REQUEST` indica generalmente che la query (indirizzo o latLng) è mancante.

8 Conclusioni e sviluppi futuri

Durante lo svolgimento del progetto sono stati portati a termine tutti gli obiettivi prefissati e sono state sviluppate delle funzionalità che hanno permesso l'analisi di tecniche, al fine di migliorare le applicazioni nel contesto reale, in particolare:

- sono stati studiati i flussi di dati provenienti dalle varie fonti esterne, al fine di analizzare il maggior numero di indirizzi presenti nel web per ricavarne una standardizzazione;

- il numero delle tabelle, usate per effettuare il riconoscimento di un indirizzo, è stato incrementato, con l'aggiunta nel database della tabella contenente tutte le frazioni di italia, ovvero più di 60.000 voci;

- dopo un'attenta analisi sono state individuate le migliori strategie di riconoscimento utilizzabili nel contesto reale richiesto;

- sono state studiate ed implementate nuove tecniche di geolocalizzazioni al fine di apportare miglioramenti a quelle inizialmente proposte;

- l'implementazione delle tecniche descritte è stata svolta in modo da rendere agevole la portabilità del codice su altre architetture;

- sono state confrontate le diverse tecniche al fine di individuare pregi e difetti di ognuna.

Considerando che la seconda parte del progetto è stata svolta in un' ottica di estensione e usabilità orientate agli sviluppi futuri del progetto sulla gestione di eventi geolocalizzati su una mappa, sono elencate qui alcune proposte, in particolare, per quando riguarda le tecniche sulla geolocalizzazione:

- queste si drovebbero testare su scenari molto più estesi, con dati sugli eventi di qualsiasi categoria, aggiornati e forniti da fonti affidabili, al fine di validare il tutto;

- le categorie possono variare dal cinema, teatro, concerti live, manifestazioni, sagre, fiere, mostre, mercatini, concorsi, feste, eventi culturali e privati.

Per quanto riguarda invece le possibili estensioni di utilizzo:

- si potrebbe sviluppare un'applicazione mobile sulla gestione di eventi, sfruttando le caratteristiche dei moderni smartphone, come il gps per localizzare gli eventi presenti nelle zone limitrofe del punto individuato;

Figura 1. Eventi su mappa visualizzata su smartphone

- si potrebbe anche sviluppare un sito web per condividere, aggiungere o segnalare gli eventi presenti su tutto il territorio nazionale;

- si potrebbe addirittura, implementare funzionalità come la gestione degli eventi, all'interno di un navigatore satellitare, sotto forma di punti di interesse. Tali dispositivi, dovrebbero avere la possibilità di collegarsi alla rete in tempo reale, quindi di essere online per avere a disposizione dei dati aggiornati. Dispositivi che a loro volta possono essere impiantati su veicoli circolanti negli scenari urbani delle nostre città.

Figura 2. Eventi su mappa all'interno dei gps di automobili

Bibliografia

[1] David Flanagan, *Javascript: The Definitive Guide*. Oreilly, 2011

[2] George Shepherd, *Microsoft ASP .Net 4*. Mondadori informatica, 2010

[3] Marco Ferrero, *Access*. Apogeo, 2005

[4] Michael Halvorson, *Microsoft Visual Basic 2010*. Mondadori informatica, 2011

[5] Paolo Camagni e Riccardo Nikolassy, *Programmare per il Web. HTML, CSS, JavaScript, VBScript, ASP, PHP*. Hoepli informatica, 2009.

[6] Riccardo Nikolassy, *HTML, CSS, XML*. Hoepli informatica, 2006.

[7] Document Object Model (DOM). Web sites: http://www.w3.org/DOM/ http://www.w3.org/2003/01/dom2-javadoc/index.html

[8] Georeferenzazione e geolocalizzazione, da Wikipedia. Website: http://it.wikipedia.org/wiki/Georeferenzazione

[9] The Google Geocoding API. Web site: http://code.google.com/intl/it-IT/apis/maps/documentation/geocoding/.

[10] VisualBasic. Website: http://www.html.it/guide/guida-visual-basic/