POLITECNICO DI MILANO

FACOLTA' DI INGEGNERIA

Dipartimento di Elettronica e Informazione

Generazione di mappe geografiche personalizzabili senza vincoli d'autore con tecnologie Open Source

Relatore: Prof. Giuseppe Pozzi
Correlatore: Ing. Stefano Cazzani

Elaborato di laurea di:
Riccardo Fosso – Matricola: 671632

Anno accademico 2009/2010

RINGRAZIAMENTI

Ringrazio Vertigo che mi ha dato la possibilità di realizzare questo progetto. Ringrazio il Prof. Giuseppe Pozzi per avermi seguito nella stesura della tesi. Ringrazio la mia ragazza Simona e i miei nonni. Più di tutti ringrazio i miei genitori e mia sorella, che mi hanno sempre sostenuto e incoraggiato.

Indice generale

1

INTRODUZIONE

Introduzione

Fin dall'antichità l'uomo ha sempre cercato di conoscere i territori che stavano intorno a lui ed è sempre stato attratto dai luoghi inesplorati, spingendolo ad attraversare situazioni e avventure inimmaginabili, portandolo così ad una conoscenza più ampia del mondo che lo circondava. Avventurieri, esploratori, studiosi hanno iniziato così a scoprire e a documentare le strade che il mondo poteva offrire al genere umano, immagazzinando le informazioni sempre più dettagliate in strumenti via via più sofisticati, si è passati da rudimentali disegni sulla pietra, ai papiri, alla carta fino ad arrivare a calcolatori capaci di fornire istantaneamente informazioni dettagliate su qualsiasi parte del globo terrestre.

Tutto questo ha generato al giorno d'oggi un giro d'affari enorme, con aziende pronte ad offrire nuove funzionalità e prestazioni sempre più elevate dei proprio prodotti nel tentativo di primeggiare nel mercato del settore.

Alcuni dei primi strumenti che vengono in mente al giorno d'oggi pensando alla cartografia in generale sono ad esempio Google Maps, cartine turistiche rilasciate da vari enti culturali o uno dei vari navigatori GPS attualmente in commercio. Strumenti che oramai sono entrati nella vita quotidiana di moltissime persone perché sono comodi, veloci, relativamente precisi e consentono di fare tutto con più facilità. Purtroppo però questi strumenti sono soggetti al diritto d'autore, quindi non consentono la distribuzione ne la personalizzazione di queste informazioni.

Il tema di questo documento è legato ad un progetto eseguito durante il mio tirocinio formativo presso Vertigo S.R.L., e sulla cartografia libera, ma prima di approfondire il progetto realizzato nel tirocinio, introdurrei il problema della cartografia proprietaria e dei vincoli imposti che comportano l'utilizzo di questi strumenti.

I dati geografici (geo data), in molti paesi del mondo, Italia ed Europa incluse, non sono gratuiti, ma sono soggetti a delle royalty secondo le disposizioni dei loro autori. In linea di massima l'onere della realizzazione delle mappe è delegata ad agenzie nazionali che poi le rivendono a privati o aziende e ne ricavano finanziamenti. Gli Stati Uniti

d'America sono l'unica eccezione macroscopica; qui infatti, le leggi sul copyright delle agenzie nazionali rendono questi dati di pubblico dominio.

Quindi chi abita nei paesi che hanno i diritti d'autore sulle mappe, paga le tasse perché vengano realizzate le mappe, e nuovamente paga se vuole avere una copia di quelle stesse mappe.

I dati forniti però, non sono soggetti a modifiche sostanziali in tempi rapidi, infatti le strade non si annoiano e quindi non si spostano di propria iniziativa[OpenStreet Map00]. In linea di massima, una qualsiasi mappa tracciata dopo la seconda guerra mondiale si può considerare sufficientemente dettagliata per potersi muovere agevolmente anche oggi.

Spesso le mappe che si trovano in commercio, e che potremmo pensare di "copiare", contengono errori intenzionali chiamati in gergo *Easter Eggs*. Si tratta solitamente di strade inesistenti o mancanti, oppure indicazione di edifici che in realtà non esistono. Ovviamente se si tentasse di realizzare una mappa utilizzando proprio questi dati protetti da diritto d'autore, le ditte od enti che le hanno realizzate potranno tranquillamente riconoscere l'utilizzo improprio o la modifica di una loro mappa, semplicemente controllando se sono presenti i loro errori.

Le mappe possono essere imprecise anche perché, come detto precedentemente sono state realizzate anni prima ed alcune strade potrebbero essere diventate parchi ecc. Se si ammettessero queste difficoltà, risulta chiaro che di quei dati si possono al massimo fare delle fotocopie, ed anche questo in gran parte del mondo potrebbe essere una violazione della legge. Non si può correggere il nome di una strada od aggiungere l'indicazione di un pub lungo la strada o anche usare i dati con un computer senza pagare i relativi diritti. Mandare una mappa ad un amico, inserirla su di un forum o metterla come indicazione in un invito sono cose meno legali di quanto si sia portati a pensare.

Il progresso tecnologico e la presenza sul mercato di ricevitori GPS economici hanno reso possibile creare le proprie mappe, in collaborazione con altri o meno, senza alcuna delle limitazioni di cui si è appena letto. Questa possibilità ci restituisce una parte della comunità in cui viviamo: se non puoi realizzarne una mappa non puoi descriverla[OpenStreet Map00].

Dopo aver affrontato il tema della cartografia in generale e dei vincoli che possiede, è ora di introdurre il tema principale della tesi, ovvero la realizzazione del sistema produttivo per la generazione di cartine tematiche tramite cartografia libera e strumenti open source, per questo è opportuno avere una visione generale del progetto.

L'idea del progetto e quindi anche di questa testi, sono nati da Vertigo, azienda operante nel campo dell'informatica da vent'anni. Il concetto di base è molto semplice, infatti il progetto deve poter fornire una suddivisione in cartine di una generica nazione, fornendo informazioni stradali, geografiche e amministrative mirate, escludendo così tutte le informazioni reputate superflue in modo da poter assicurare delle guide stradali semplici ed essenziali.

Lo scopo ultimo però non è legato solo alla produzione di tali mappe geografiche, ma devono essere presenti su di esse anche informazioni per raggiungere aziende di un

determinato settore commerciale, garantendo indicazioni dettagliate sfruttando le informazioni di un indice. L'intero progetto, come si è intuito fin dalle prime pagine, è soggetto ad una limitazione di grande rilievo, ovvero le cartine in Italia sono coperte da diritti d'autore, per questo bisogna trovare una soluzione per mantenere i costi entro un certo limite e soprattutto per poter realizzare cartine liberamente personalizzabili così da poterle adattare alle esigenza del progetto e presumibilmente alle esigenze di un eventuale cliente che richiederebbe tale servizio. Ad ogni modo la trattazione e lo sviluppo del progetto avranno ampio spazio nei capitoli seguenti.

La soluzione per lo sviluppo di questo progetto, per abbattere i limiti imposti dalla legge e dai dati forniti dalla cartografia proprietaria, è arrivata fortunatamente grazie ad alcune persone brillanti che riescono a aggirare se non addirittura ad abbattere i limiti o le difficoltà imposte da terzi, sviluppando tecnologie open source, argomento molto importante per questo documento.

Ed è proprio da questi limiti che è nato OpenStreet Map, un progetto che punta a creare e fornire dati cartografici, ad esempio mappe geografiche libere e gratuite per chiunque ne abbia bisogno. Il progetto è stato avviato proprio perché la gran parte delle mappe che si credono liberamente utilizzabili, hanno invece restrizioni legali o tecniche, come detto precedentemente limitando così la produttività e la creatività dei soggetti.

I dati di OpenStreet Map possono essere usati liberamente secondo i termini della licenza Creative Commons Attribution-ShareAlike 2.0 license che garantiscono:

- di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera[Creative Commons00]
- di modificare quest'opera [Creative Commons00]

Oltre a godere di queste proprietà, bisogna però anche seguire queste condizioni:

- Attribuzione: Devi attribuire la paternità dell'opera nei modi indicati dall'autore o da chi ti ha dato l'opera in licenza e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. [Creative Commons00]
- Condividi allo stesso modo: Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. [Creative Commons00]

e inoltre il software di OpenStreet Map è disponibile con licenza GNU GPL[OpenStreet Map00].

Questo significa che è possibile, grazie alla GNU General Public License, garantire la libertà di condividere e modificare tutte le versioni di un programma e di fare in modo che esso rimanga software libero per tutti gli utenti[GNU GPL00], in termini più pratici è consentito fare quel che si vuole con i dati di questo sistema, a patto che in

ogni prodotto derivato venga citata la fonte dei dati e chi ha contribuito e sia specificato che al prodotto derivato è applicata la stessa licenza.

Riguardo ad OpenStreet Map sarà possibile leggere all'interno di questo testo la sua storia e un'ampia descrizione così da poter fornire una vista completa del potenziale produttivo che ha portato, o porterà, in questi anni e in un certo senso della grande influenza che ha sulla comunità.

1.1 Executive Summary

1.1.2 Capitolo 2

Il Capitolo 2 descrive a che punto è arrivata la tecnologia nel settore della cartografia, dando una panoramica mirata sui principali sistemi attualmente disponibili. I temi trattati sono le tecnologie GPS spiegando nozioni di base sul loro funzionamento, Google Maps il servizio di posizionamento più utilizzato sul web e il suo diretto concorrente: OpenStreet Map. Infine viene introdotto il sistema G.I.S che è il sistema che si preoccupa di rappresentare elettronicamente la terra.

1.1.3 Capitolo 3

Nel Capitolo 3 viene fornita una descrizione dettagliata degli strumenti che vengono utilizzati per sviluppare questo progetto, concentrando l'attenzione come è strutturato e cosa ha da offrire OpenStreet Map, vero e proprio fulcro dell'intero sistema produttivo proposto in questa tesi. Viene introdotto uno strumento importante di OpenStreet Map, ovvero Mapnik, il motore di rendering che permette, previa modifica, di ottenere una personalizzazione grafica delle cartine in questione, ma viene introdotto anche PostgreSQL che è il database adibito alla gestione dei geo dati.

1.1.4 Capitolo 4

In questo Capitolo, si affronta direttamente l'argomento della testi, ovvero viene spiegato come realizzare le mappe personalizzate e libere da qualsiasi licenza. La descrizione è accompagnata da tutorial che spiegano come preparare la macchina al lavoro che deve svolgere, ma anche da frammenti di codice per mostrare come poter far interagire tutti gli elementi del progetto per cercare di ottenere un'automatizzazione quasi completa del processo, inoltre ci sono immagini e da esempi utili a rendere più

chiaro il passaggio dal concetto astratto del problema ad uno sviluppo concreto di tutto il processo creativo del progetto.

1.1.5 Capitolo 5

Qui si possono leggere i commenti sul lavoro finito, valutando e giudicando cosa si è ottenuto. I risultati prodotti sono da considerarsi positivi, secondo il Tutor aziendale che ha seguito da vicino il progetto, perché tutte le problematiche hanno ricevuto una equa trattazione, portando così a valutare e a sviluppare ogni aspetto. Indubbiamente il processo di questa tesi non è esente da errori o da possibili miglioramenti, infatti verrà introdotto un metodo alternativo per una parte del progetto.

Il fatto di aver realizzato o quanto meno valutato ogni punto richiesto per la riuscita del lavoro ha portato ad una consapevolezza maggiore conferendo più certezze sulla validità del lavoro svolto.

1.1.6 Capitolo 6

Capitolo conclusivo di questa testi, che offre la possibilità di una valutazione generale del lavoro svolto pensando anche a come è possibile migliorare il sistema o a quali sviluppi futuri può portare. In generale il progetto può considerarsi concluso perché l'obbiettivo primario era quello di verificare al validità del lavoro e come detto nel Capitolo 5 questo punto è stato soddisfatto, ma questo non preclude miglioramenti, infatti le informazioni di OSM sono in costante aumento e questo può portare solo ad un miglioramento delle cartine, perché il quantitativo di dati diventa sempre più superiore e quindi si ottengono mappe più accurate.

Ovviamente questo è solo un vantaggio per il progetto, ma il limite è imposto dal flusso delle informazioni che è indipendente da questo progetto, ma è legato solo hai mappers che devono dedicarsi alla mappatura di nuovi territori.

2

STATO DELL'ARTE

Il Capitolo 2 descrive a che punto è arrivata la tecnologia nel settore della cartografia, dando una panoramica mirata sui principali sistemi attualmente disponibili. I temi trattati sono le tecnologie GPS spiegando nozioni di base sul loro funzionamento, Google Maps il servizio di posizionamento più utilizzato sul web e il suo diretto concorrente: OpenStreet Map. Infine viene introdotto il sistema G.I.S che è il sistema che si preoccupa di rappresentare elettronicamente la terra.

2.1 Panoramica dei principali sistemi di navigazione Web e GPS

In questi anni si è riscontrato uno sviluppo impressionante di sistemi di navigazione portatile e non, portando quasi qualsiasi calcolatore, grande o piccolo che sia, ad avere integrato un sistema di navigazione. Per alcuni di questi è addirittura un valore aggiunto poter vantare un GPS integrato nelle proprie funzione, basti pensare ai vari Nokia, iPhone e affini.

Di seguito verrà fatta una panoramica generale sui principali sistemi di posizionamento.

2.1.1 Google Maps

Google Maps è un servizio che consente di visualizzare mappe nel browser web in uso e senza ombra di dubbio, si può considerare la mappa digitale più famosa del mondo[Romagnolo00]. In base alla posizione di interesse dell'utente, è possibile visualizzare mappe semplici o personalizzate, infatti grazie alle sue funzionalità è possibile vedere l'area interessata in modalità satellite, mostrando delle fotografie satellitari a diverse scale di zoom, in modalità mappa ove è possibile avere una visione

molto più tradizionale fornendo una visione simile ad un'atlante stradale e infine, Google Maps consente di usare la funzione Street View che permette all'utente di immergersi nelle strade di suo interesse grazie a delle immagini in alta definizione delle strade vere e proprie[Romagnolo00].

Google Maps non è solo questo. infatti oltre alla ricerca delle strade, consente anche di ricercare le informazioni sulle attività commerciali locali, come ad esempio gli indirizzi, le informazioni di contatto e le indicazioni stradali per raggiungere tali attività. È dotato inoltre di un sistema molto intuitivo e semplice per l'utente permettendogli di trascinare le mappe per visualizzare istantaneamente le sezioni adiacenti, oppure passare velocemente da una visuale in modalità mappa ad un'immagine satellitare della località desiderata su cui è possibile eseguire lo zoom e la panoramica[Romagnolo00].Oltre ai risultati di ricerca, potrebbero essere forniti anche annunci Google AdWords con estensioni di località considerati pertinenti alla ricerca dell'utente. Un annuncio può essere pubblicato come annuncio di testo o annuncio in un formato più completo, con un'icona dell'attività commerciale e un indicatore di mappa che si espande mostrando un'immagine dell'attività commerciale. Questi annunci sono associati a un punto specifico sulla mappa, ad esempio un negozio o un ristorante. Sulle mappe di minori dimensioni, se un utente facesse clic sull'indicatore di mappa associato, si aprirebbe una finestra informativa contenente ulteriori informazioni sull'attività commerciale.

Google Maps è un servizio gratuito su un computer desktop o portatile; è sufficiente disporre di una connessione Internet e di un browser web supportato.

2.1.2 Global Positioning System (GPS)

Il sistema in oggetto, la cui denominazione completa è “NAVSTAR GPS”, abbreviazione di “Navigation Satellite Timing And Ranging Global Positioning System” (sistema di posizionamento globale mediante misure di tempo e distanza rispetto a satelliti per
navigazione) è stato realizzato a partire dal 1975 circa, a cura dello U.S. Department of Defence; il sistema, concepito per un posizionamento ad alte prestazioni per scopi militari in
particolare per la missilistica, è impiegabile anche per usi civili quali la navigazione l’utilizzo
topografico, ecc.

La costellazione di satelliti cui il sistema fa riferimento è costituita da 24 satelliti, progettata in modo da renderne visibili fino a 12 contemporaneamente, da ogni punto della superficie terrestre, in ogni ora del giorno e della notte (si vedrà come questa sia condizione
indispensabile per l’applicabilità del metodo). Il sistema permette di eseguire il posizionamento di punti appartenenti alla superficie terrestre, ovvero la determinazione della posizione dei punti interessati in un assegnato sistema di riferimento.

Il posizionamento che si ottiene con il GPS è tridimensionale, quindi contemporaneamente
planimetrico e altimetrico.

2.2 Introduzione ai sistemi G.I.S.

Solitamente, per spiegare di cosa si occupa un sistema G.I.S. è sufficiente sciogliere l'acronimo e tradurre quel *Geographical Information Systems* con un "Sistemi informativi territoriali". Non sempre però questo basta, pertanto si cerca di scendere su definizioni più semplici e, se vogliamo, anche troppo banali. Una di queste può essere "cartografia elettronica" [Napolitano01], pertanto, visto che questa tecnologia si occupa di rappresentare in formato elettronico la Terra, ma prima di approfondire il tema dei G.I.S., viene presentato un breve sunto della loro storia.

Nascono nella seconda metà degli anni '60 e inizialmente vengono impiegati prevalentemente in ambienti di ricerca a causa degli elevati costi dovuti alla componente grafica, ad esempio alcuni dei primi fruitori sono stati il Canada Land Inventory, l' Esercito degli USA e infine alcune industrie petrolifere. Negli anni '70 e '80 inizia una lenta, ma costante diffusione di questi sistemi nelle grandi aziende private, grazie anche allo sviluppo e ai miglioramenti tecnologici hardware che coinvolgono sia la resa grafica a livello video, sia quella che ne permette la riproduzione su carta: le stampanti. Infine negli anni '90 c'è la conferma definitiva di questi sistemi che sono presenti in moltissimi ambienti, grazie soprattutto all'aumento dei PC [Sedita00].

La citazione che segue riassume a pieno la grande adattabilità di questi sistemi e ne giustifica il suo sviluppo:

"...la cartografia non più solo rivolta ad indicare – con sempre maggiore precisione – dove sono e che forma hanno gli oggetti, naturali o artificiali, ma anche attenta a delineare le caratteristiche quantitative di "cose" e "fenomeni", le loro correlazioni, per rintracciare una possibile spiegazione e quindi formulare leggi di comportamento dei fenomeni stessi..."[Lodovisi00].

G.I.S. memorizza le informazioni geografiche come una collezione di layers (strati) tematici che possono essere tra loro relazionati tramite collegamento e sovrapposizione geografica. Questo semplice ma estremamente potente e versatile concetto è applicato per risolvere diversi problemi reali quali ottimizzazione di percorsi, applicazioni di pianificazione urbanistica, modelli di circolazione atmosferica, ecc.

2.2.1 OpenStreet Map

"OpenStreet Map è una mappa liberamente modificabile dell'intero pianeta. È fatta da persone come te. OpenStreet Map permette a chiunque sulla Terra di visualizzare,

modificare ed utilizzare dati geografici con un approccio collaborativo." Steve Coast creatore di OpenStreet Map [Delucchi00].

OpenStreet Map è un progetto simile a Wikipedia per le mappe, permette la creazione delle stesse e mette a disposizione dati geografici liberi, come ad esempio le mappe stradali. Il progetto è nato perché gran parte delle mappe che si pensa siano liberamente utilizzabili hanno restrizioni tecniche o legali che non ne consentono l'utilizzo nei settori più disparati, ed in generale non sono utilizzabili in altre opere derivate, per scopi creativi, produttivi o in altre maniere inaspettate[de Rossi00].

Come detto poc'anzi, esiste una similitudine tra OpenStreet Map e Wikipedia, infatti chiunque può dare il proprio contributo, aggiungendo, modificando o eliminando informazioni in strade, edifici, punti d'interesse. Di seguito i punti chiave che riguardano lo stesso spirito che anima OpenStreet Map e Wikipedia[de Virgilio00]:

- sistema di tracciamento delle modifiche, storico dei cambiamenti
- libertà fondamentali
- attività collaborativa contemporanea basati completamente su software libero
- importazione da altre fonti autorizzate (previa compatibilità licenza)

Le mappe di OpenStreet Map, oltre ad assere molto attendibili e dettagliate, sono anche prive di Easter Eggs, errori inseriti volutamente dai proprietari delle mappe per tutelarsi dalle copie non autorizzate, l'immagine seguente è un chiaro esempio di Easter Eggs:

Illustrazione 1: Esempio di Easter Eggs creato per Google Maps

Importante anche sottolineare che le fonti dei dati raccolti in OpenStreet Map possono provenire da varie sorgenti, ma sono tutte accomunate da una sola caratteristica: sono dati liberi. Infatti molti dei dati che mette a disposizione questo ambizioso progetto, sono frutto del lavoro dei suoi utenti che con i propri ricevitori GPS raccolgono dati, inoltre è anche consentita l'importazione di geo dati rilasciati con licenza libera, ad esempio donazioni di dati da parte di enti pubblici o privati, ma anche è permessa l'importazione di dati ricavati da foto aeree anch'esse rilasciate con licenza

libera e infine, è permessa l'importazione di dati ricavati da cartografie il cui copyright è scaduto. Quindi ancora una volta, l'importante è che i dati siano a licenza libera.

I dati vettoriali sono comunque liberamente scaricabili e non di rado vengono renderizzati da singoli utenti, gruppi, università secondo i propri scopi. Non di meno attualmente si sta lavorando alla creazione di buoni convertitori per ricavare da OSM delle buone mappe da sostituire a quelle commerciali usate nei comuni navigatori GPS.

Tutte questi dati, oltre ad essere disponibili a tutti, sono in constate crescita. Infatti, gli oltre 100.000 iscritti attuali si sono organizzati nei vari progetti nazionali per potersi organizzare meglio, gli strumenti principali di comunicazione sono le onnipresenti mailing list e l'utilissimo Wiki soprattutto per la definizione dei vari tag definiti e votati di volta in volta se ne presenti la necessità.

Una delle cose più gradite alla comunità è l'organizzazione di ”Mapping Party” dove intere città o quartieri vengono battuti centimetro per centimetro da tutti i mappers della zona per creare una mappa più dettagliata possibile. Il progetto inoltre provvede a fornire una mappa ufficiale renderizzata ogni settimana che sempre più spesso viene usata nei siti di tutto il mondo per non doversi legare a Google per questo tipo di servizio ma, per fortuna, sempre più spesso anche per la qualità della mappa stessa.

In ogni caso questo è solo una piccola introduzione di questo interessante progetto, che verrà trattato dettagliatamente nel Capitolo 3, tutto dedicato al funzionamento di OpenStreet Map e per concludere questa introduzione seguiranno i punti salienti della sua storia:

- Il progetto è stato fondato nel 2004 da Steve Coast
- Nel 2006 OpenStreet Map diventa una fondazione
- Nel Settembre 2007 inizia l'importazione dei dati TIGER grazie anche a Automotive Navigation Data, importante azienda che opera da 30 anni nel capo della cartografia digitale, dona i dati di Olanda, India, Cina
- Maggio 2008 68.000 utenti iscritti, 3400 attivi ogni settimana (40.000 iscrizioni nel solo mese di giugno) 510.000.000 di punti GPS - (350.000.000 nel solo giugno)
- 2008: donazione/investimento da parte di Cloud Made di 2,4 milioni di euro al progetto
- Aprile 2009: la FAO dona i dati cartograci di tutti i paesi africani
- Nel 2009 vengono superati i 100.000 iscritti

3

RACCOLTA DEI REQUISITI E SCELTA DEGLI STRUMENTI

Nel Capitolo 3 viene fornita una descrizione dettagliata degli strumenti che vengono utilizzati per sviluppare questo progetto, concentrando l'attenzione come è strutturato e cosa ha da offrire OpenStreet Map, vero e proprio fulcro dell'intero sistema produttivo proposto in questa tesi. Viene introdotto uno strumento importante di OpenStreet Map, ovvero Mapnik, il motore di rendering che permette, previa modifica, di ottenere una personalizzazione grafica delle cartine in questione, ma viene introdotto anche PostgreSQL che è il database adibito alla gestione dei geo dati.

3.1 Descrizione di OpenStreet Map

Dopo aver dato un primo sguardo generale a OSM (per comodità da qui in poi OpenStreet Map verrà chiamato con il suo acronimo), le pagine che seguiranno saranno dedicate ad una vista più dettagliata di questo brillante a ambizioso progetto, OSM appunto.

È possibile schematizzando il funzionamento tecnico di OSM e notare che ci sono cinque punti fondamentali sulla quale si basa l'intero progetto. Di seguito i punti sottostanti verranno trattati:

- Raccolta dei dati
- Upload dei dati
- Creazione / Editing dei dati OSM
- Etichette e aggiunta dei dettagli
- Rendering delle mappe

3.1.1 Raccolta dei dati

OSM è un progetto indipendente gestito solamente da persone comuni o per meglio definirli “appassionati”, come già indicato in precedenza, e proprio grazie a queste persone che è stato possibile sviluppare e raccogliere dati per questo progetto. Ebbene si, può sembrare bizzarro, ma come facevano gli esploratori nell'antichità, questi “appassionati” armati con diversi strumenti vanno in giro e raccolgono le informazioni per OSM.

Esistono varie metodologie per la raccolta dei dati da inserire in OSM, di seguito verranno proposte le due tecniche principali di acquisizione:

1. GPS: questo è il metodo più comune di raccogliere i dati per OSM, e per una mappatura completa delle aree è essenziale. Se non si ha un GPS allora ci si può basare sui tracciamenti GPS che altri utenti hanno messo a disposizione alla comunità di OSM, inoltre ci si deve assicurare che i dati siano acquisiti o convertiti (prima della pubblicazione) nel sistema geografico di riferimento WGS-84 che è il sistema di riferimento predefinito per i ricevitori GPS.
2. Le proprie fotografie o mappe: si deve essere sicuri che queste siano completamente libere da copyright prima di usarle per OSM. Molti dei dati non sono liberi da copyright come spesso si potrebbe credere.

È importante precisare che ci sono molti tipi di dati che si possono aggiungere a OSM: dalla più comune caratteristica quale il nome della strada fino al più fine dettaglio che include ad esempio il parco, la buca delle lettere, e specifici permessi di accesso. Ogni persona ritiene che alcune caratteristiche siano più importanti rispetto ad altre, tipicamente influenzate dal loro metodo principale di spostamento. Per esempio le persone che usano spesso le strade provinciali sono più facilmente interessate alle stazioni di rifornimento rispetto ai nomi degli alberghi.

Spesso i *mappers* aggiungono sempre più caratteristiche nel tempo, ma iniziano con quelle basilari. Il seguente è un esempio sull'ordine in cui si dovrebbe gradualmente aggiungere dettagli:

- Tipo e nome della strada
- Parcheggio
- Direzione della strada (e se è a senso unico o a doppio senso)
- Limitazioni di accesso: ad esempio accesso permesso solo ai mezzi di trasporto pubblico (autobus, taxi)
- Utilizzo principale dell'area: per esempio residenziale
- Percorsi pedonali, ciclabili e altri tipi
- Edifici con uno specifico uso, come ad esempio hotel/stadi/ecc. e i loro nomi
- Dettagli fini, come le aree circostanti (es. boschi), strade di accesso, incroci, ponti e gallerie
- Ulteriori luoghi degni di nota: panorami, monumenti/memoriali, fontane, etc.

3.1.2 Upload dei dati

Ottenuti i dati è necessario condividerli e per fare questo OSM offre la possibilità di upload dei dati comodamente dal proprio sito internet, bisogna però attenersi a delle operazioni semplici da effettuare. Di seguito i passi necessari per la pubblicazione dei dati in OSM:

1. Salvare i proprio file in formato GPX
2. Pubblicare in OSM
3. Scaricare i dati in JOSM

Le prime due operazioni sostanzialmente non meritano grandi commenti perché ad esempio la prima operazione dipende dal modello di ricevitore GPS che si utilizza e quindi sta all'utente sapere come salvare o convertire in modo corretto le informazioni che ha ottenuto, mentre per il secondo punto, pubblicare in OSM, si tratta di una semplice operazione di upload che fornisce il sito di OSM.

Sono operazioni semplici e meccaniche che fondamentalmente servono solo per spostare i dati dal dispositivo GPS al database di OSM.

L'unica nota potrebbe essere legata al terzo punto che introduce un uno strumento di OSM molto importante per la comunità di OSM, si tratta di JOSM, ma è meglio rinviare la trattazione di questo argomento nel paragrafo seguente.

3.1.3 Creazione / Editing dei dati OSM

Naturalmente dopo aver ottenuto i dati dal rilevamento effettuato sul campo, bisogna avvalersi di un editor che consente la modifica e/o la gestione dei dati GPS e dei dati di OSM.

Come accennato prima, si tratta di JOSM che supporta il caricamento delle tracce GPX dei dispositivi GPS e i dati GPX dall'archivio OSM, ma è anche in grado di gestire altre tipologie di dato come i nodi, rotte, metadati, etichette (tag) e relazioni (relation) già esistenti negli archivi di OSM.

Il fatto di dover prima effettuare l'upload delle tracce GPS appena rilevate sul server di OSM e poi scaricare nuovamente la stessa area geografica, serve fondamentalmente a legare eventuali dati esistenti già in quella zona geografica, ovvero se dovessero esistere già altri dati dovuti ad un precedente rilievo bisogna legare i nuovi dati con quelli vecchi, e ciò possibile farlo usando proprio JOSM. Segue ora una breve descrizione per mostrare come vengono rappresentati i dati su OSM che sono composti dai seguenti elementi:

- **Nodi:** I punti che sono usati per tracciare i segmenti tra di essi.
- **Vie:** Una lista ordinata di nodi, visualizzati come connessi da segmenti nel programma di modifica.
- **Vie chiuse:** Le vie chiuse sono vie che si chiudono. Sono usate per descrivere delle aree come i parchi, i laghi o le isole.

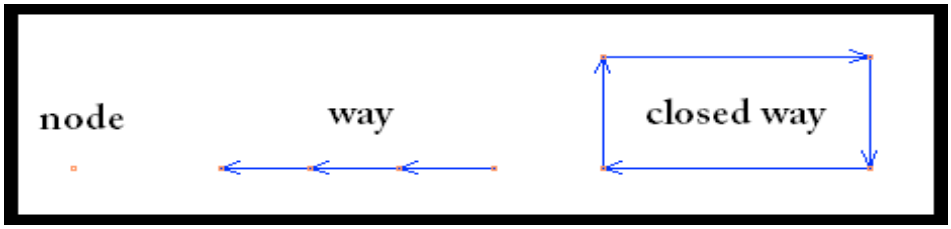

Illustrazione 2: Rappresentazione dei dati si OSM formati degli elementi: nodi, vie e vie chiuse[OpenStreet Map00]

3.1.4 Etichette e aggiunta dei dettagli

Come per ogni altro prodotto, anche nel mondo della cartografia esistono prodotti migliori e prodotti peggiori, la differenza sostanziale che determina o no il successo di una mappa dipende dai dettagli. Infatti dopo che si è appena una completo il lavoro di editing con JOSM di una via via cittadina tracciata con il ricevitore GPS, è necessario aggiungere delle etichette in modo da arricchire il tracciato fornendo così maggiori informazioni per rendere la mappa più accurata.

Esistono varie etichette, alcune delle quali più comuni di altre, come ad esempio le indicazioni per le autostrade, strade provinciali, fiumi, laghi, parchi, città, ma anche teatri, cinema o addirittura esistono alcune etichette ancora più esotiche come quelle utilizzate per identificare le fontanelle delle città. Non appena si sono fatte delle modifiche, le stesse si possono pubblicare in OSM. L'immagine che segue è un rappresentazione di come si vedono i dati all'interno di JOSM, tuttavia, è opportuno precisare che JOSM non consente una visuale molto chiara delle informazioni, o per meglio dire, il suo modo di rappresentare le informazioni e quindi le mappe, a livello visivo è più chiaro solo agli "addetti ai lavori".

Per meglio comprendere questa affermazione è consigliato guardare l'immagine sottostante per rendersi conto della rappresentazione troppo schematiche della mappa:

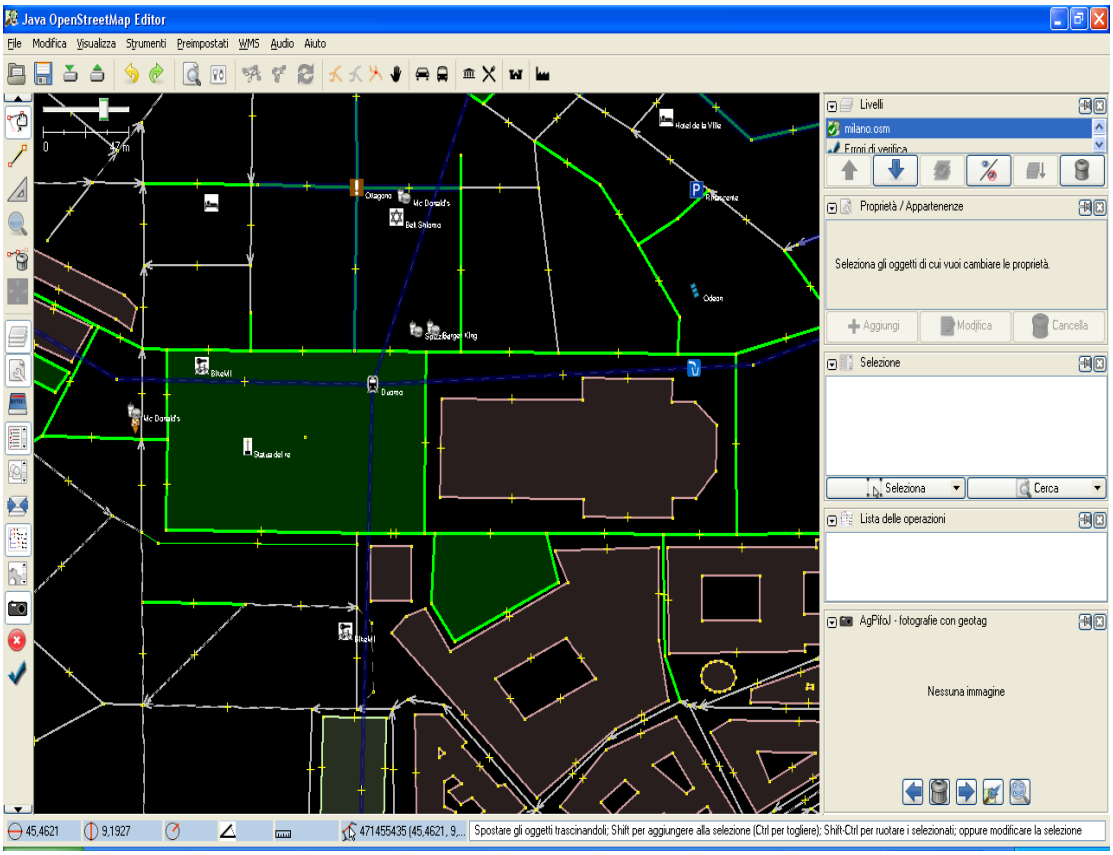

Illustrazione 3: Veduta di Piazza del Duomo in Milano tramite JOSM

3.1.5 Rendering delle mappe

Le operazioni fin ora descritte hanno portato ad aggiungere tracciati ad OSM, etichettato i dati in questione, e pubblicati su OSM. Manca però la parte conclusiva che ai fini del tema di questa tesi, ha ricoperto un ruolo essenziale, si tratta del rendering.

Il sistema di OSM offre diverse alternative per realizzare il rendering delle informazioni, e gli strumenti più famosi e utilizzati sono Osmrender e Mapnik. Ai fini di questo progetto è stato utilizzato Mapnik per la sua maggiore adattabilità al problema, quindi ne seguirà una descrizione dettagliata.

Mapnik è un programma scritto in linguaggio Python molto rapido e preciso nella realizzazione delle immagini. In altri termini, il concetto fondamentale che sta alla base di questo motore di rendering è quello di accedere alla sorgente dei dati, e "disegnare" tali informazioni, senza considerare le relazioni che ci sono tra i vari nodi. Più avanti questa affermazione sarà più chiara.

Questo software, o per meglio dire *toolkit*, per lavorare e quindi produrre mappe, deve poter accedere ai geo dati e per far ciò ha a disposizione molte strade, infatti è in grado di leggere shapefile ESRI, PostGIS, TIFF raster, file con estensione osm, o

qualsiasi GDAL OGR. Nel caso in esame, la sorgente dati sarà contenuta in PostgreSQL che è un completo database relazionale ad oggetti rilasciato con licenza libera(stile Licenza BSD). Spesso viene abbreviato come "Postgres", sebbene questo sia un nome vecchio dello stesso progetto.

Questo database già consente la gestione dei dati di tipo geometrico, ma con l'estensione . PostGIS aggiunge funzioni geospaziali e due tabelle di metadati che rende più agevole la gestione dei dati OSM all'interno del database.

Lo schema di database è stato ottimizzato per l'utilizzo con Mapnik, infatti contiene tabelle come planet_osm_point, planet_osm_line, planet_osm_polygon per rispettivamente punti, linee e poligoni, insomma proprio i dati sono stati raccolti. In aggiunta vi è una tabella planet_osm_roads per alcun particolari accorgimenti sui dati di linea ad alcuni livelli di zoom particolari.

I geo dati di ogni nazione sono forniti da OSM però sono contenuti in file con estensione osm, i quali devono essere convertiti per poter essere immessi all'interno di Postgres, quindi per far ciò si può utilizzare *OSM2PGSQL*, una utility di conversione con perdita di dati.

Caratteristica interessante quest'ultima perché snellisce molto la mole di dati, ricordando che dentro i file osm è presente qualsiasi nodo di una mappa, ma questa sua particolarità aggiunge solo alcuni "tag certi" perdendo così la connessione e le relazioni tra i vari nodi trasformandoli in linee e poligoni, ad esempio non interessa ai fini del rendering sapere se Via Ponzio incrocia Via Bassini.

OSM2PGSQL ha due modalità di funzionamento normale e slim. La modalità normale utilizza la RAM come depositò dati temporaneo, mentre agli inizi del 2009, è stato necessario utilizzare slim-mode per le importazioni dei dati su sistemi a 32 bit. Questo cambio di modalità è dettato dal fatto che sono troppi nodi da memorizzare nella RAM a causa del fatto che OSM è diventato sempre più popolare e quindi la quantità di dati è cresciuta esponenzialmente[Delucchi00]. Utile precisare che questa limitazione non si applica ai sistemi a 64 bit. Dopo l'esecuzione di *OSM2PGSQL*, si ottengono all'interno del database tutte le informazioni della area geografica interessata

Finita questa parentesi sulla sorgente dei dati si può tornare a parlare di Mapnik, continuando ad analizzare gli strumenti che lo compongono. Mapnik è fornito con due script scritti in Python che consentono di realizzare il rendering delle mappe. Il primo, *generate_tiles.py*, consente di spezzettare in tante immagini, a diversi livelli di zoom, una certa area geografica, per rendere più chiaro questo argomento, si può supporre di avere interesse a ottenere la mappa di Milano e che di questo territorio si abbia l' interesse di avere la possibilità di zoomare liberamente su qualsiasi area della città, proprio come si fa su internet quando si utilizza OSM o *Google Maps*.

Dividendo il territorio di Milano in tante piccole immagini(*tiles*, letteralmente significa mattonelle), è possibile ottenere il massimo del dettaglio di una piccola porzione, tiles appunto, ma mettendo insieme tutte le tiles si otterrebbe l'intera immagine della città.

Il lavoro compiuto da questo script è proprio quello di creare una suddivisione di determinate aree geografiche, salvando dentro a delle directory ordinate le immagini di

dimensioni 256x256 pixel, seguendo una classificazione “matriciale”, ovvero la prima cartella conterrà la prima immagine in che si collocherà in alto a sinistra quindi in posizione [1][1], mentre la seconda immagine sarà in posizione [2][1] e così via[Delucchi00]. La figura che segue è la rappresentazione di Milano a zoom 11 e mostra la disposizione delle tiles:

Illustrazione 4: Rappresentazione di Milano ottenuta utilizzando lo script generate_tiles.py fornito con Mapnik

Il secondo script, *generate image.py*, consente di creare delle immagini “statiche” di una certa area geografica, ovvero una singola immagine con uno zoom prefissato garantendo comunque un dettaglio molto elevato. Anche in questo caso segue un'immagine che dimostra quanto appena detto:

Illustrazione 5: L'immagine generata con lo script generate image.py, fornisce una rappresentazione dell'Italia

Mapnik però non dispone solo dei due script che appena mostrati, ma è fornito anche con un file XML dalle dimensioni decisamente importanti, ben 7846 righe di codice, che consente di definire gli stili grafici delle mappe. Questo file può essere tranquillamente modificato tramite qualsiasi editor XML, ma ora segue una descrizione delle caratteristiche del file in questione, ovvero osm-template.xml.

Il file osm-template.xml contiene le informazioni necessarie per realizzare il rendering di una mappa con le impostazioni base di OSM. Le immagini che mostrato poco prima, Illustrazione 4 e Illustrazione 5, sono state realizzate proprio con questo file XML.

Si nota subito che i colori delle strade non corrispondono ai colori convenzionali usati in Italia per questo motivo ai fini del progetto è necessario creare uno stile personale che si rifà il più possibile allo stile visivo delle cartine nazionali e per ottenere questa personalizzazione bisogna modificare il file sopracitato.

Il file osm.xml (per brevità verrà chiamato sempre così) a prima vista sembra un file molto articolato ed effettivamente è così, ma con un po di pazienza si può entrare nella struttura e capire di più[Pavlenko00]. Ogni elemento, come la terra, le autostrade, i parchi nazionali, le vie delle città hanno delle proprie regole all'interno di questo file, nelle quali sono definite regole di visibilità a diversi zoom, colori, simboli, font ecc…

Prima viene definito uno *stile* per ogni elemento, prendendo in esame le autostrade, è possibile trovare la loro definizione in *<Style name="roads">* e all'interno di questo stile sono definite delle regole che permettono di modificare il livello. I campi più rilevanti sui quali agire sono:

- Filter: definisce il tipo di informazione che si vuole trattare, ovvero il soggetto del quale si vuole realizzare il rendering. Nel caso preso in esempio Filter ha il valore highway (autostrade).
- MaxScaleDenominator: valore di tipo intero che specifica fino livello di scala si può arrivare con il rendering.
- MinScaleDenominator: valore di tipo intero che specifica il livello minimo dal quale si può partire per il rendering.
- CssParameter: qui vengono definite le regole CSS per il rendering, scegliendo il colore, piuttosto che la dimensione di ciò che si preferisce.

Altro elemento importante di cui è utile tenerne conto è lo *ShieldSymbolizer*. ShieldSymbolizer permette di inserire un'immagine all'interno della mappa, come ad esempio il cartello(shield per l'appunto) delle autostrade. Anche in questo caso seguono i campi del comando ShieldSymbolizer:

- Name: stampa il campo name, ovvero il nome della città piuttosto che il nome dell'autostrada.
- Face_ name: indica il tipo di font che vogliamo utilizzare per l'attributo Name.
- Size: definisce la dimensione del testo contenuto nel campo Name.
- Fill: è il colore del testo che verrà stampato.
- Placement: può essere di tipo point o line. Point, serve per indicare se si vuole mettere il simbolo che identifica ad esempio una strada statale, sulla linea che rappresenta la strada, o se si preferisce posizionarlo accanto alla rappresentazione della strada, line.
- File: va indicato il percorso del file immagine riguardante un determinato simbolo affinché il sistema possa caricare e quindi effettuare il rendering dell'immagine.

- Type: indica il formato immagine del file sopracitato. Può essere di tipo png o jpeg.
- Width: corrisponde alla larghezza del file immagine indicato.
- Height: corrisponde all'altezza del file immagine indicato.
- Min_distance: indica la distanza minima che ci deve essere tra uno shield e l'altro, fornendo quindi "un'area di rispetto". Questo discorso vale anche tra shield differenti.
- Spacing: definisce la spaziatura tra le occorrenze ripetute dello stesso shield.
- Dy: indica il valore di scostamento che deve avere il testo dall'immagine.

Dopo aver indicato alcune delle regole grafiche fondamentali da utilizzare per la realizzazione delle mappe, viene ora introdotto un aspetto importante sempre legato al file osm.xml e cioè la definizione dei *Layer* definiti per ogni elemento. All'interno dei Layer, possiamo trovare le istruzioni che ci consentono di ottenere i dati da una sorgente che verranno, secondo le regole grafiche viste prima, mandati al motore di rendering.

Per non discostarsi dall'esempio fatto prima sulle regole grafiche viene ora proposto il Layer *roads* che accede al database di Postgres per ottenere i dati da processare:

```
<Layer name="roads" status="on" srs="+proj=merc +a=6378137 +b=6378137
+lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null
+no_defs +over">
    <StyleName>roads</StyleName>
    <Datasource>
    <Parameter name="type">postgis</Parameter>
      <Parameter name="host">localhost</Parameter>
      <Parameter name="port">5432</Parameter>
      <Parameter name="user">'Username'</Parameter>
      <Parameter name="password">"Password"</Parameter>
      <Parameter name="dbname">'gis' </Parameter>
      <Parameter name="table">
      (select way,highway,railway,
       case when tunnel in ('yes','true','1') then 'yes'::text else
tunnel end as tunnel
       from planet_osm_roads
       where highway is not null
          or railway is not null
       order by z_order
      ) as roads
      </Parameter>
      <Parameter name="estimate_extent">false</Parameter>
      <Parameter name="extent">-20037508,-
19929239,20037508,19929239</Parameter>
    </Datasource>
</Layer>
```

Come si può notare dopo aver definito il nome del layer, notiamo che il campo *srs* contiene diverse definizioni difficilmente capibili ad un primo guardo. Niente paura, sono solo le proiezioni per interpretare i geo dati contenuti nel database PostgreSQL. Di norma conviene lasciarli con le impostazioni di default su ogni layer.

Proseguendo nella lettura del codice si nota che ci sono dei comandi semplici per accedere al database, merita invece un interesse maggiore l'istruzione *<Parameter name="table">*.

Questa istruzione effettua una query sul database di OSM, fornendo così tutte le informazioni utili a Mapnik: il nostro motore di rendering. Da notare che per ogni livello (*layer*) si deve definire una query diversa, così da poter estrarre città, paesi, strade, ma anche zone di interesse naturalistico, quali parchi, fiumi ecc…

Le ultime due istruzioni invece riguardano sempre le proiezioni dei dati, quindi conviene lasciarle invariate.

4

DESCRIZIONE DEL SISTEMA

In questo Capitolo, si affronta direttamente l'argomento della testi, ovvero viene spiegato come realizzare le mappe personalizzate e libere da qualsiasi licenza. La descrizione è accompagnata da tutorial che spiegano come preparare la macchina al lavoro che deve svolgere, ma anche da frammenti di codice per mostrare come poter far interagire tutti gli elementi del progetto per cercare di ottenere un'automatizzazione quasi completa del processo, inoltre ci sono immagini e da esempi utili a rendere più chiaro il passaggio dal concetto astratto del problema ad uno sviluppo concreto di tutto il processo creativo del progetto.

4.1 Introduzione

Ideazione e verifica di un processo di trasformazione di database contenenti dati strutturati per realizzare Guide Tematiche (directory) con cartine stradali da pubblicare su carta riproducibile senza vincoli di licenza.

Obiettivi correlati:

- Privilegiare le tecnologie open source
- Sviluppare un prototipo basato su una base dati reale (su un tema a scelta)
- Trovare il bilanciamento più conveniente tra metodologie automatiche e lavorazioni manuali

4.2 Descrizione del sistema

Seguirà ora la descrizione della realizzazione del sistema di questo testo, facendo una panoramica dei punti cardine del progetto, quelli dove è importante focalizzare l’attenzione per sviluppare e comprendere questo sistema.

Prima di tutto è utile trovare degli strumenti che consentano di utilizzare dati cartografici liberi da licenza, ma questo aspetto è già stato ampiamente affrontato nel Capitolo 3, quindi è possibile passare direttamente al punto successivo, ovvero la *suddivisione in sezioni.* Si tratta di un aspetto importante perché una volta ottenuti i dati cartografici si devono valutare metodi per ripartire l'Italia in modo omogeneo per densità di dati, e questo comporta relative implicazioni sulla scelta delle scale delle singole cartine. Gli ultimi due aspetti da tenere in considerazione riguardano la *mappatura su carta* e la *costruzione dell'indice geografico*. Il primo punto è legato alla qualità grafica fornita dalle mappe e alla distribuzione delle informazioni che le cartine devono fornire per le aziende, mentre il secondo punto serve al completamento dei dati iniziali con le coordinate geografiche di lettura della cartine impaginate.

A questo punto della trattazione sembra tutto fin troppo astratto, perciò prima di proseguire con gli sviluppi di questi temi, è necessario pensare al proprio ambiente di lavoro, come organizzarlo e prepararlo per il lavoro che seguirà. Verrà quindi introdotto, nel paragrafo seguente, un esaustivo tutorial che spiegherà come rendere pronta la propria postazione di lavoro.

4.3 Tutorial

- Installare Postgres 8.4, in alternativa vanno benissimo le precedenti versioni
- Installare estensione Postgis reperibile in rete per le versioni precedenti alla 8.4 di PostgreSQL, altrimenti se si utilizza l'ultima versione di Postgres la si può trovare sotto
  start - > programmi -> PostgreSQL 8.4 e aprite l'applicazione *Application Stack Builder* e scaricare l'estensione Postgis sotto la voce *Spatial Extensions*.
- Aprire PostgreSQL e creare un nuovo ruolo, quindi andremo su **ruolo utente -> nuovo ruolo utente** impostando i seguenti parametri:

  - Come nome del ruolo diamo il nome che abbiamo dato all'account Windows
  - In privilegi di ruolo spunteremo tutto

- Ora si va su database e con il tasto destro creiamo un nuovo database con i seguenti parametri:

  - Nome database: gis
  - Proprietario: nome del ruolo creato prima
  - Codifica: UTF8
  - Modello: template_postgis
  - Tablespace: pg_default
  - Aggiungeremo al database alcuni requisiti tramite una query:

```
GRANT ALL ON DATABASE gis TO public;
GRANT ALL ON DATABASE gis TO " account windows ";
COMMENT ON DATABASE gis IS 'OSM';
```

- Clicchiamo sul nostro database, si può notare che sulla barra dei menù si attiverà la funzione SQL, apriamola ed eseguiamo una query con i seguenti comandi:

    - ALTER TABLE geometry_columns OWNER TO NOME_ruolo;
      ALTER TABLE spatial_ref_sys OWNER TO NOME_RUOLO;

- Il prossimo passo sarà quello di andare a modificare il file pg_hba.conf in modo che l'accesso al server possa essere fatto senza l'utilizzo di una password, altrimenti il programmino OSM2PSQL non riuscirebbe ad accedervi.
  File->Apertura di pg_hba.conf
  Selezioniamo il file che nel mio caso ha il seguente percorso:
  C:\Program Files\PostgreSQL\8.4\data
  Una volta aperto modifichiamo il metodo in 'trust' e facciamo ricaricare il server.
- A questo punto bisognerà scaricare il file OSM (italy.osm) che conterrà tutte le informazioni riguardanti i nodi, le vie e le relazioni tra esse della area che si vorrà renderizzare.
- Scaricare l'utility OSM2PGSQL e copiarla in **C:\Program Files\PostgreSQL\8.4** (o percorso simile)

- Apriamo ora DOS, posizioniamoci nella cartella dove abbiamo copiato OSM2PGSQL ed eseguiamo i seguenti comandi:

    - psql -d gis -f dir_di installazione_di_OSM2PGSQL\900913.sql
    - osm2pgsql -m -s -d gis nome_area.osm, come spiegato in precedenza questo è il commando per eseguire l'utility in modalità slim.

- Installare Python 2.5 seguendo la procedura guidata d'installazione
- Scaricare Mapnik ver. 0.6.1 – win 32

    - Decomprimere la cartella in **c:\mapnik_0_6_1**
    - Settare le *Variabili d'Ambiente* aggiungendo alla variabile PATH **c:\mapnik_0_6_1\lib**
    - Aggiungere **c:\mapnik_0_6_1\site-packages** alla variabile PYTHONPATH opportunamente creata
    - Aprire la console di Python e digitare

      ```
      from mapnik import *
      ```

      se non ci sono errori allora tutto è andato a buon fine

- A questo punto creeremo una nuova cartella all'interno di Mapnik di nome 'osm' con all'interno una cartella di nome boundaries che conterrà i file presenti negli archivi *processed_p.zip* e *world_boundaries-spherical.tgz* scaricabili tramite i

seguenti indirizzi *http://hypercube.telascience.org/~kleptog/processed_p.zip* e *http://tile.openstreetmap.org/world_boundaries-spherical.tgz*, una cartella vuota di nome 'tiles' che sarà poi riempita dai tiles che si andranno a renderizzare, infine
una cartella *symbols* in cui copieremo tutti quei simboli, scaricabili da *http://svn.openstreetmap.org/applications/rendering/mapnik/symbols*, usati durante il render per creare strade punti di interesse etc.

- All'interno della cartella *osm* copieremo i file Python di configurazione: *generate_tiles.py, set-mapnik-env, osm-template.xml, customize-mapnik-map* tutti scaricabili da *http://svn.openstreetmap.org/applications/rendering/mapnik/*

- Modificare il file osm-template.xml con i seguenti parametri:

  - `SYMBOLS_DIR = C:/mapnik-0_5_1/osm/symbols`
  - `WORLD_BOUNDARIES_DIR = C:/mapnik_0_6_1/osm/boundaries`
  - `DBHOST = localhost`
  - `DBPORT = 5432`
  - `DBNAME = 'gis'`
  - `DBUSER = 'NOME_RUOLO'`
  - `DBPASS = "LA PROPRIA PASSWORD"`
  - `PREFIX = "planet_osm"`

- Una volta sostituite queste variabili con i dati, bisognerà modificare alcune cose all'interno del file generate_tiles.py in fondo al file al posto di:

```
if __name__ == "__main__" :
home = os.environ['HOME']
try:
mapfile = os.environ['MAPNIK_MAP_FILE']
except KeyError:
mapfile = home + "/osm-local.xml"
try:
tile_dir = os.environ['MAPNIK_TILE_DIR']
except KeyError:
tile_dir = home + "/osm/tiles/"
```

bisognerà inserire:

```
if       __name__       ==       "__main__":
home        =        "C:/mapnik_0_6_1/osm"
mapfile    =    home    +    "/osm-template.xml"
tile_dir = home + "/tiles/"
```

Dopo aver fatto tutto ciò sarà possibile far partire, tramite la console di Python o semplicemente facendo doppio click su un file, lo script *generate_tiles.py* o lo script *generate_image.py*.

4.4 Suddivisione in sezioni

Un aspetto rilevante, come accennato prima, per il progetto, è quello di fornire delle mappe di facile lettura e quindi dotate di uno stile visivo chiaro e semplice da interpretare solo con un colpo d'occhio. Ciò porta senza ombra di dubbio ad una creazione di un nuovo layout grafico, ma questo nuovo stile da solo servirebbe a poco, bisogna infatti trovare anche il modo di poter ripartire le informazioni geografiche fornite dalle cartine in modo quanto più possibile omogeneo e chiaro. Dunque nasce il problema di come suddividere in modo corretto l'Italia.

Realizzare le cartine sulle singole regioni come "isole fluttuanti" slegate le une dalle altre avrebbe prodotto solo gran confusione, facendo perdere qualsiasi connessione stradale all'utente finale. La soluzione migliore prodotta è quella di cercare di mantenere una suddivisione regionale, ma mostrando anche parte dei territori delle regioni circostanti, che ovviamente verranno poi completati nelle cartine successive. Ad esempio, una regione molto vasta è la Lombardia e non sarebbe utile ai fini di questo progetto mostrare tutto il suo territorio in una sola mappa, ma invece "spezzare" la Lombardia in due e mostrare a Ovest una piccola parte del Piemonte e a Est mostrare una parte del Veneto, avrebbe sicuramente prodotto una visione più chiara del territorio globale, garantendo così all'utente finale di poter mantenere una chiarezza delle informazioni.

Continuando a ragionare su una possibile ripartizione del territorio nazionale si passa per un punto decisamente semplice, ma utile per arrivare al risultato finale, ovvero tutti quanti sanno perfettamente che l'Italia è accerchiata dai mari e si estende lungo una diagonale, ebbene queste due banali osservazioni sono state sicuramente un punto di partenza per arrivare allo scopo. Si può notare dall'illustrazione seguente che l'area riguardante il nord Italia è stata semplice da gestire perché qualitativamente è possibile approssimarla ad un rettangolo, il quale è stato a sua volta suddiviso in nove parti.

Ora che la divisione è stata realizzata è possibile iniziare a ragionare a come implementare un codice che permetta di generare le mappe in modo automatico seguendo questa divisione territoriale. Il modo più efficiente per svolgere tale compito è stato quello di adattare lo script *generate image.py* alle esigenze di progetto.

Ad una prima riflessione si può pensare di utilizzare un altro strumento, ovvero *generate tiles.py,* perché consente già di per se di realizzare una sorta di suddivisione territoriale, come è stato spiegato nel Capitolo 3 paragrafo 3.1.5, ma una delle ragioni principali che ha allontanato l'attenzione da tale script era che non servivano diversi livelli di zoom, mentre *generate image.py* ha le caratteristiche di base per poter essere sfruttato, infatti questo script genera delle immagini statiche di una prefissata area

geografica, inoltre gode di una proprietà molto utile rispetto al suo “avversario”, perché *generate image.py* è molto più stabile e richiede meno risorse al sistema, quindi è possibile ottenere le immagini con una maggiore velocità rispetto allo script delle tiles.

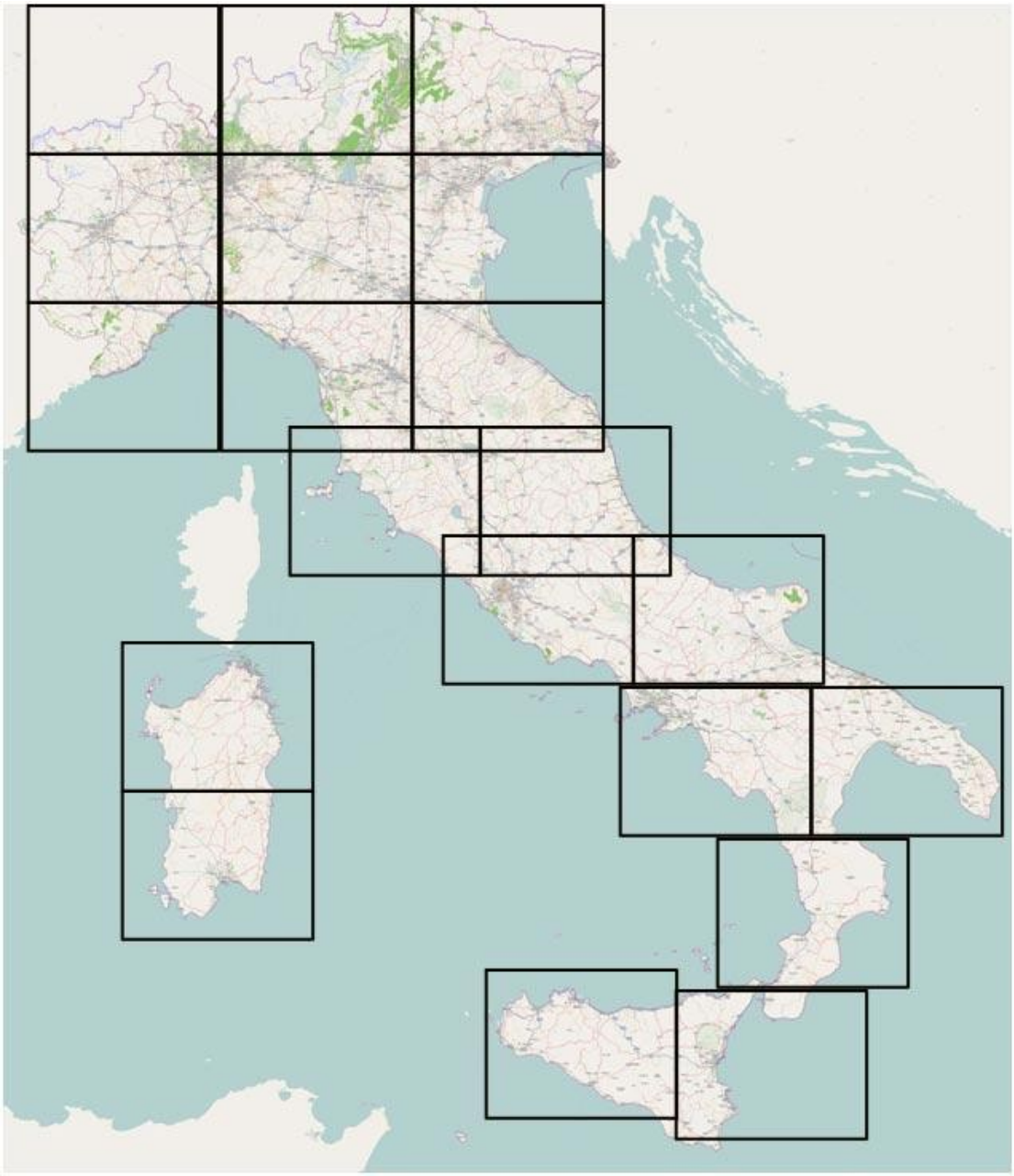

Illustrazione 6: Schema di suddivizione del suolo italiano per fornire una equa ripartizione delle imformaizoni geografiche

È doveroso far presente che anche se *generate tiles.py* non è lo script migliore da utilizzare in questa circostanza, la sua logica di base è stata utile all'adattamento dello script scelto per realizzare la suddivisione dell'Italia. Si potrebbe quindi affermare che

per ottenere un codice che potesse portare al risultato voluto, si è scelta la logica di *generate tiles.py* sfruttando però le performance di *generate image.py*.

Il concetto di base che sta dietro al codice è generare immagini del suolo italico partendo dal vertice in alto a sinistra e muovendosi verso destra ovviamente spostandosi anche verso il basso nei punti che richiedono questo movimento. Segue una schematizzazione del processo:

- definisco le coordinate iniziali di una certa area geografica (nota anche come bbox, boundary box).
- setto la dimensione dell'immagine della quale si vuole ottenere il rendering
- si procede al rendering vero e proprio.
- una volta ottenuta l'immagine e salvata su file si deve passare all'incremento delle coordinate, così da poter renderizzare la prossima immagine e quindi la prossima sezione di territorio.

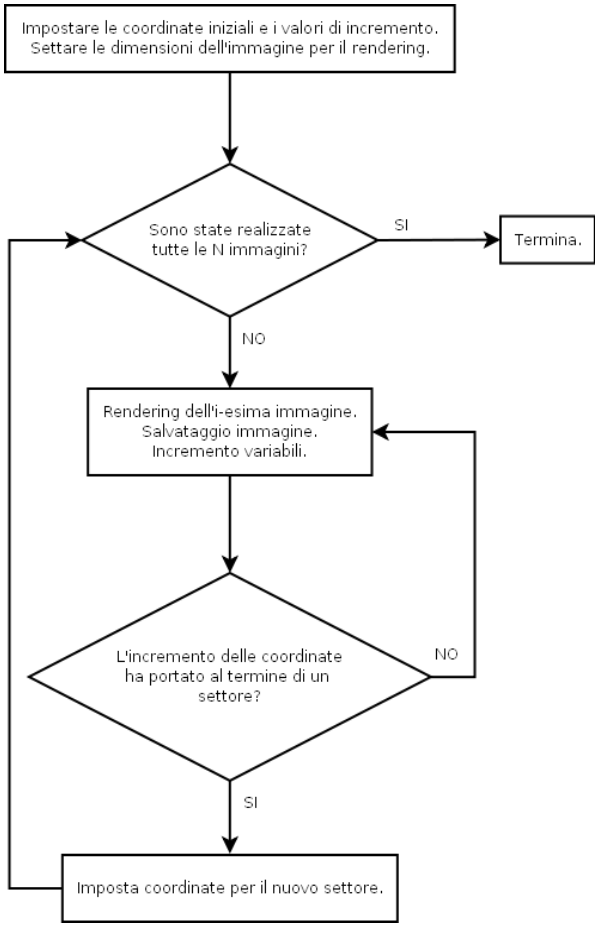

Illustrazione 7: Diagramma di flusso del codice "generazione automatica tavole.py" per la suddivisione in settore dell'Italia

Ovviamente ci sono diversi punti critici, come ad esempio quando si raggiunge il confine est della nostra nazione, perché qui il cambio di variabile deve essere fatto sia lungo x che lungo y. A questo punto si ottengono tutte le immagini del nostro bel paese. Il codice "*generazione automatica tavole.py"* [A.A.1] descritto sopra sarà disponibile nell'Appendice di questo testo.

Eseguendo questo codice si ottengono le immagini, ma sono ancora lontane dal risultato che si vuole ottenere, infatti devono essere personalizzate a livello grafico.

4.5 Mappatura su carta

Come ampiamente descritto nel Capitolo 3 paragrafo 3.1.5 la personalizzazione grafica avviene tramite la modifica del file osm.xml, ma la trattazione di questo argomento non andrà ad analizzare in modo dettagliato le singole istruzioni per la modifica, perché sono state già mostrate in precedenza, ma bensì questo testo si preoccuperà prevalentemente di illustrare quali sono state le linee guida seguite per giungere al risultato richiesto da questo progetto.

Come già detto, l'Illustrazione 4, è il prodotto dello stile predefinito di OSM che segue i colori e gli stili delle mappe stradali inglesi, non che siano totalmente incomprensibili per noi italiani, ma sono sicuramente diverse dal nostro modo di vedere cartine e quindi un utente un po distratto potrebbe essere un po spiazzato. Dunque bisogna poter garantire a tutti gli utilizzatori di questo strumento di poter capire al volo quello che stanno guardando, perciò occorre modificare i colori ad esempio delle autostrade o delle provinciali, ma anche rimuovere alcune informazioni che ai fini del progetto risulterebbero superflue e in alcuni casi potrebbero addirittura appesantire di molto la leggibilità delle cartine precludendo così un'eventuale arricchimento futuro con le informazioni delle aziende. Inutile precisare che la personalizzazione della mappe è lasciata alla libera fantasia degli sviluppatori e perciò la modifica descritta sarà solo una delle tante possibilità che si possono ottenere senza avere la pretesa di essere considerata la soluzione definitiva per il suolo italiano.

Sempre facendo riferimento all'Illustrazione 4 è possibile notare che le autostrade sono di colore blu mentre storicamente le autostrade in Italia sono sempre state rappresentate in verde o in verde con una striscia gialla all'interno, mentre sempre facendo riferimento alla figura, si nota che le provinciali sono rosa mentre in Italia hanno sempre seguito una colorazione mattone. Sempre nello stile predefinito di OSM si possono notare addirittura le vie cittadine che però a livelli di zoom elevati, come nel caso in esame, non risulterebbero chiare e perciò è stato deciso di eliminarle fornendo così una veduta più chiara della mappa, così come parchi nazionali o indicazioni geografiche come monti, o ancora indicazioni amministrative come i confini comunali sono stati rimossi sempre per favorire la leggibilità della mappa.

Non sono state effettuate solo operazioni di modifica di colori o eliminazioni, ma sono stati aggiunti alcuni dettagli sempre con lo scopo di migliorare la leggibilità delle

informazioni, come ad esempio sono stati inseriti del “pallini” nei pressi dei grandi centri abitati, dei medi e dei piccoli così da rispettare lo schema classico delle mappe nel nostro paese. Sicuramente però l'Illustrazione 8 sarà più esaustiva di qualsiasi parola che si può usare per descrivere la personalizzazione. Ultima nota sull'immagine che seguirà è che questa immagine riguarda una delle zone con la maggiore quantità di informazioni, rendendo molto difficile far apparire il giusto quantitativo di informazioni, quindi questa area geografica è stata molto interessante perché si può dire che la personalizzazione grafica è stata calibrata proprio su questo territorio trasformandolo in un *benchmark*., così da poter garantire, visto che le altre aree dell'Italia non son così piene di informazioni, che con il diminuire dei dati la qualità grafica potesse rimanere sempre fedele allo standard creato per l'area lombarda.

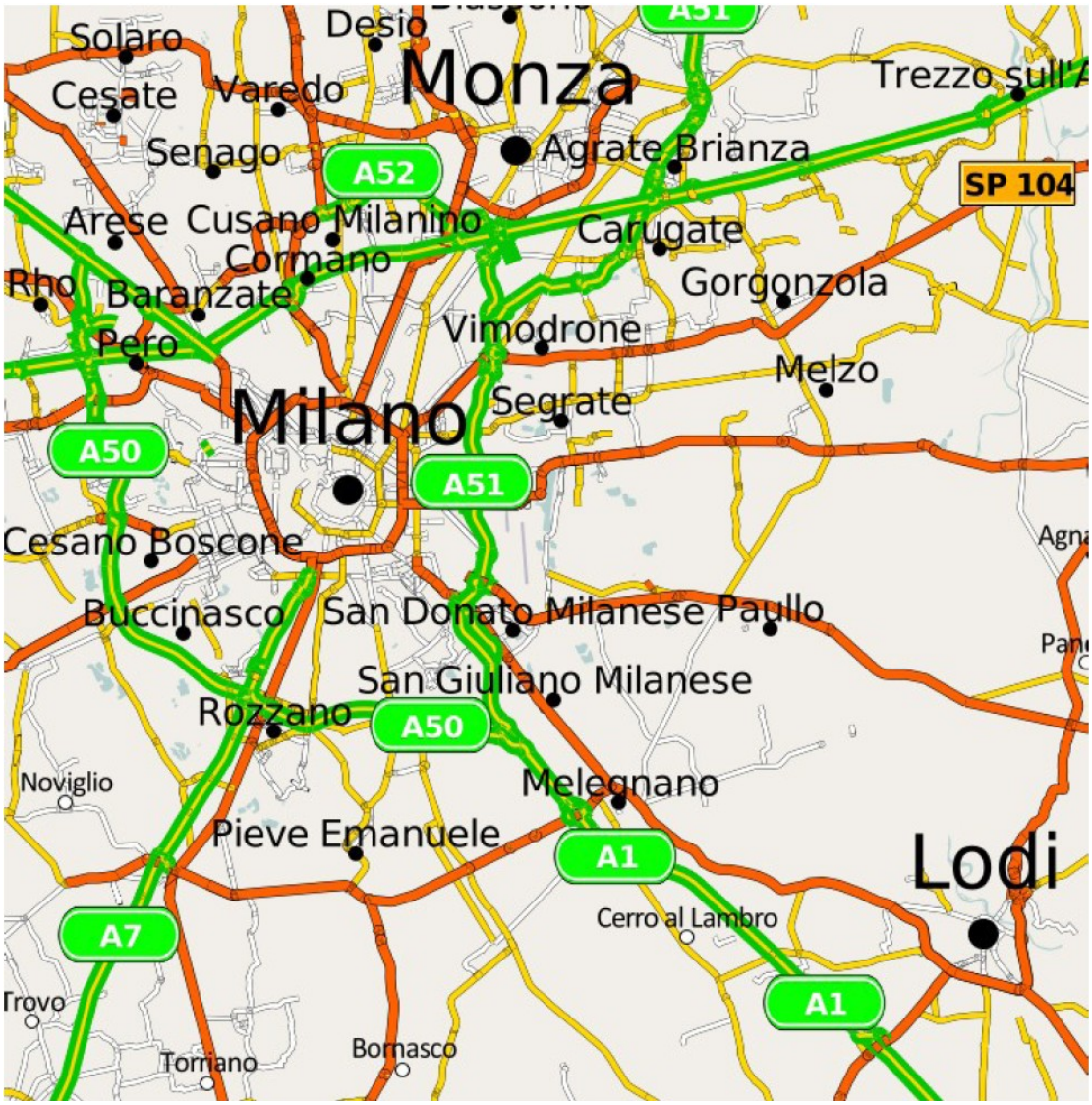

Illustrazione 8: Rappresentazione del territorio lombardo con la personalizzazione creata per il progetto

4.6 Costruzione dell'indice geografico

A questo punto del progetto sono stati fatti due passi in avanti, ovvero è possibile ottenere immagini del territorio e gestire a piacere la personalizzazione grafica delle immagini, ma ora si affronta il primo punto per l'arricchimento delle cartine con le informazioni per le aziende, prima però è necessario effettuare alcune operazioni per agevolare il lavoro. L'idea da seguire in questo frangente è quella di poter inserire dei marker all'interno delle mappe appena create, con lo scopo di indicare la presenza delle aziende o delle attività presenti in prossimità delle città che le ospitano. Dunque sorge un problema. È difficile sapere esattamente dove siano collocate tutte le città di Italia, e quindi risulta difficile poter posizionare correttamente i marker, per questo motivo ora verrà affrontata una soluzione che consente allo sviluppatore di poter ovviare a questo problema. La risposta in parte la si può trovare nel codice "*generazione automatica tavole.py"* usato per poter generare le cartine, ebbene si perché quel codice consente di creare delle mappe partendo da una bbox, ricordo che la bbox è un'area geografica, e che per rappresentare i dati in forma grafica deve accedere al database PostgreSQL tramite il file osm.xml. Quindi come prima cosa è necessario ottenere tutte le bbox, ma per far ciò basta modificare il codice sopracitato eliminando la parte di rendering che sarebbe risultata superflua dato che le mappe sono state elaborate in precedenza. Anche in questo caso nell'Illustrazione 9 è presente il diagramma di flusso per spiegare il funzionamento di quanto appena citato e nell'Appendice sarà disponibile il codice "*bbox città.py"* [A.A.2].

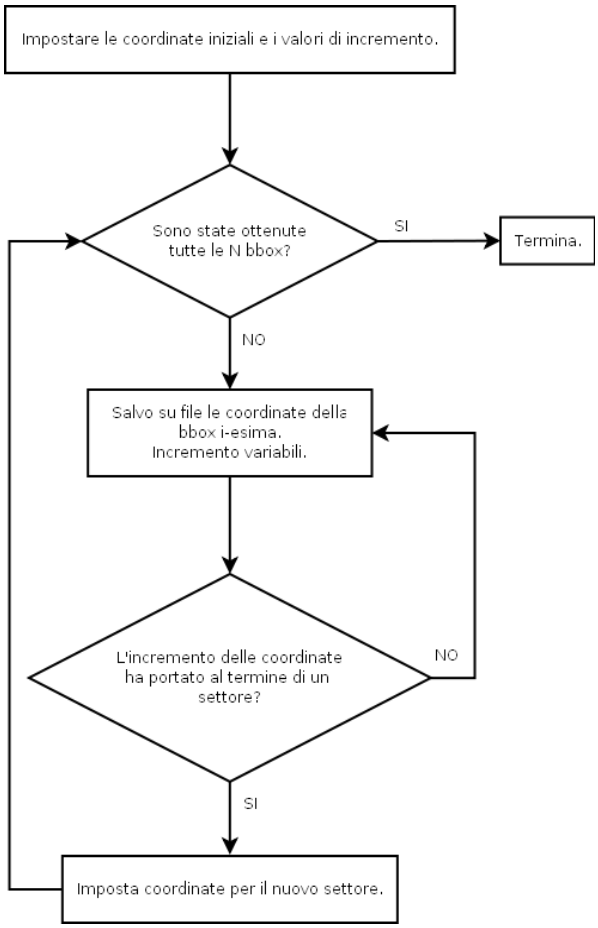

Illustrazione 9: Diagramma di flusso del codice "bbox città.py" per la raccolta di tutte le città di Italia

Ottenuti tutti i dati che a noi interessavano, si può procedere all'elaborazione di tali informazioni per iniziare realmente a costruire l'indice geografico.

Con le informazioni appena ottenute possiamo sfruttare alcune particolari capacità di PostgreSQL, per l'esattezza delle funzioni che introduce la sua estensione postgis sull'elaborazione dei geo-dati. Infatti, i dati che verranno forniti a postgis devono essere processati attraverso query, ma sfruttando delle funzioni che un normale DBMS non possiede. Tale funzione è la seguente: *SetSRID('BOX3D(xmin ymin, xmax ymax)'::box3d,srid).*

Quest'istruzione consente di estrarre, tramite query, tutti gli elementi che fanno parte di una certa bbox. Vediamo ora un esempio:

```
SELECT *
FROM planet_osm_point
WHERE way && SetSRID('BOX3D(xmin ymin,xmax ymax)'::box3d,srid)
                    AND place IS NOT NULL;
```

- planet_osm_point = è la tabella dei punti di OSM, che contiene tutte le località, le strade, le autostrade, ecc...
- way = colonna delle geometrie della tabella planet_osm_point.
- xmin ymin,xmax ymax = coordinate minime e massime della bbox. Attenzione bisogna inserire le bbox estratte poco prima.
- srid = numero dello srid (sistema di riferimento/proiezione) utilizzato di solito è 4326 (WGS84) o 900913 proiezione di google o OpenStreetMap.

Scritta in quel modo, la query precedente estrae tutte le località nella bbox settata, da città grosse come Milano a frazioni di comuni piccolissimi, se invece si volesse estrarre solo le cittdine, si può utilizzare questa variante:

```
SELECT *
FROM planet_osm_point
WHERE way && SetSRID('BOX3D(xmin ymin,xmax ymax)'::box3d,srid)
                    AND (place='city' or place='town' or place='village');.
```

Se si desidera aggiungendo anche *place = 'hamlet'* si può scendere ulteriormente di un livello, ma hai fini del progetto tali informazioni risulterebbero inutili, comunque era giusto precisare che è possibile definire con diversi gradi di profondità quali informazioni ottenere tramite tale comando.

Ora che è stato illustrato come sia possibile sapere quali città sono contenute all'interno di una determinata area, si può iniziare a pensare come poter comunicare questa informazione al database che gestisce le aziende. Il problema è stato risolto utilizzando *PHP*. È stato creato uno script capace di collegarsi al db di OSM, effettuare per ogni bbox una query per estrarre le città di quell'area geografica. Una volta ottenuta la lista è stato creato un file per ogni bbox contenente le città che appartengono alla bbox, ma vediamo ora nel dettaglio come è strutturato tale script che chiameremo *"da bbox a* tavole.php" [A.A.3].

In primo luogo bisogna collegarsi al database di PostgreSQL così da poter effettuare più avanti le query che servono, inoltre bisogna aprire anche il file contenete le bbox. Ora si può procedere in questo modo: si preleva la bbox-iesima dal file e si esegue la query per estrarre tutte le città presenti in quella zona, infine le città che si ottengono al termine della query, verranno salvate all'interno di un file di testo, quindi ci saranno tanti file di testo quante sono le bbox. I file nominati come *tavola* saranno seguiti da un numero progressivo in modo tale da poter associare rapidamente questi file con i file immagine generati tramite Mapnik.

Di seguito come al solito è presente il diagramma a blocchi del codice sopra spiegato:

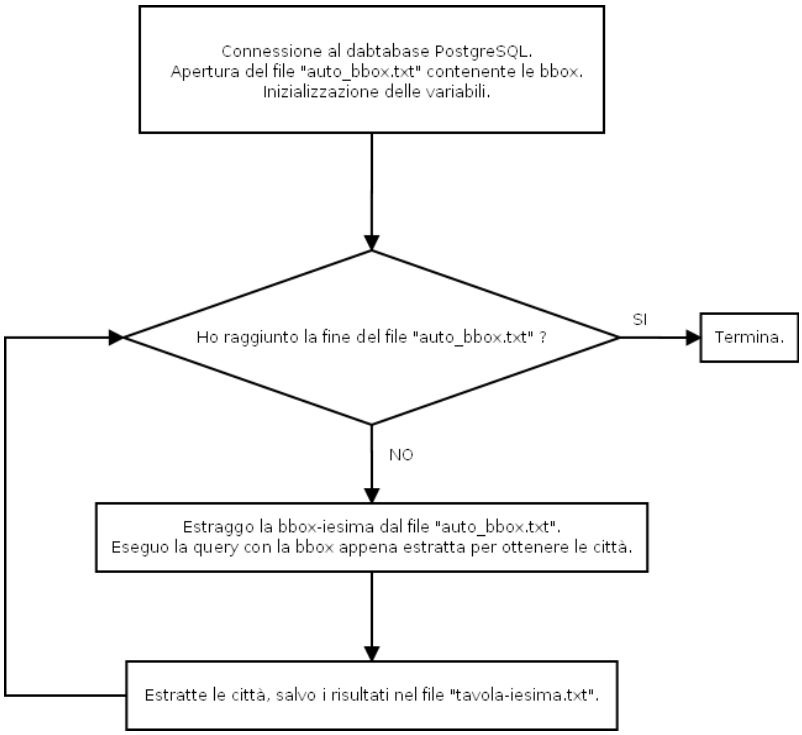

Illustrazione 10: Diagramma di flusso del codice "da bbox a tavole.php" per l'estrazione delle città presenti in una determinata area geografica.

Dopo l'esecuzione dello script avremo ottenuto una serie di file di testo aventi al loro interno le città di una determinata area geografica. Se si volesse controllare, si potrebbe prendere un file immagine ottenuto tramite Mapnik, guardare il file tavola-iesimo.txt associato e notare che sono presenti tutte le città del file immagine.

Prima di proseguire è necessario introdurre un nuovo strumento che sarà utile per questo progetto, perché per il momento ci siamo sempre occupati di layout grafici e di recuperare dati, ma ora si è giunti al momento di gestire le informazioni ottenute con il un database che contiene i dati di tutte le aziende che a noi interessano. Colgo l'occasione per ringraziare l'azienda Vertigo che mi ha fornito un database di prova completo con i dati di molte aziende operanti nel settore elettronico, con il quale ho potuto completare il progetto.

Vediamo ora come è strutturato questo database Aziende.

ID	NOME AZIENDA	INDIRIZZO	CITTA'	PROVINCIA	TELEFONO

Tabella 1: Rappresentazione del database Aziende

Si nota subito che è un database molto semplice che fornisce un elenco di aziende indicandone i dati essenziali. Inoltre tutti i campi presenti non hanno bisogno di

spiegazioni particolari, mentre ora seguirà una breve spiegazione sulla semplice modifica che è stata apportata a questo database per adattarlo alle necessità del progetto.

È chiaro che le informazioni base delle aziende sono tutte presenti in questo database, manca un campo che semplifica di molto il lavoro da svolgere e che permette di ricollegare tutto il lavoro già svolto poco fa sull'estrazione delle città di un'area geografica, ovvero il campo TAVOLA.

Questo semplice campo, aggiunto alla tabella che è appena stata mostrata, serve per inserire al suo interno la corrispondenza di una città presente, così da poter fornire un'indicazione più precisa. Anche in questo caso, con l'ausilio di uno script possiamo ottenere l'inserimento automatico di queste informazioni.

Infatti lo script "*inserisci tavole in db* Aziende.php"[A.A.4], dal titolo fin troppo chiaro, effettua la lettura di tutti i file di testo creati precedentemente e residenti all'interno della cartella *tavole.* Letto il contenuto di un degli "*n tavola file"*, lo scrip controlla se è presente una corrispondenza sia nel dbAzinede sia nel file tavola che sta processando.

Ad esempio nel file *tavola 8.txt*, è presente Milano, a questo punto lo script va a cercare se dentro il db Aziende è presente sotto il campo *CITTA'* il dato Milano, se il controllo ha esito positivo allora inserisce nel campo *TAVOLA* il nome del file, in questo caso tavola 8, contenetene quindi la corrispondenza. Questa procedura ovviamente viene eseguita per ogni città contenuta nel db Aziende.

Segue un digramma esplicativo di quanto appena detto:

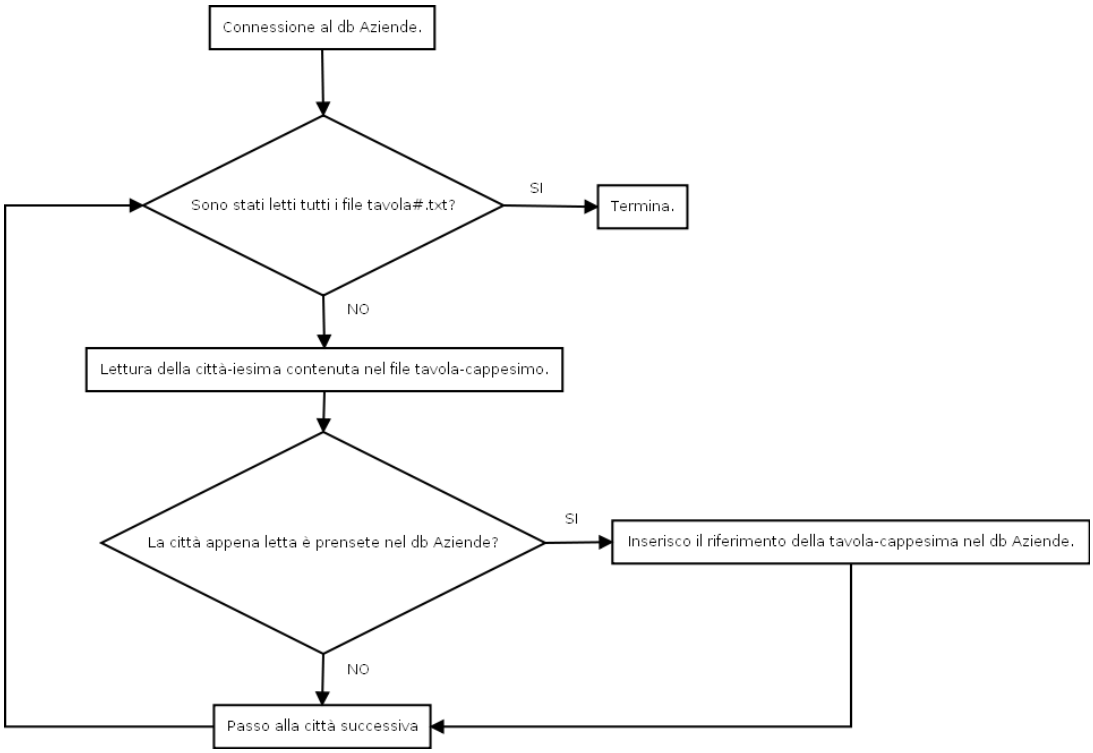

Illustrazione 11: Diagramma di flusso del codice "inserisci tavole nel db Aziende".

4.7 Creazione dei Marker

Giungiamo ora ad una delle fasi finali del progetto. Riassumiamo in pochi punti quello che fino ad ora è stato realizzato:

- Generazione grafica personalizzata di tutte le mappe.
- Creazione di un database arricchito di informazioni.
- Corrispondenza tra database Aziende e mappe grafiche.

Come oramai è ben noto bisogna indicare sulla mappa tramite un marker, un segnale grafico, che evidenzia in quel punto la possibile trovare un'azienda. La generazione di tali indicatori non è sicuramente un problema insormontabile infatti ora verrà presentata una soluzione a tale quesito. Ancora una volta tramite PHP è possibile trovare una risoluzione al problema e più precisamente con l'ausilio della libreria GD.

Breve nota sulla libreria GD di PHP: la libreria GD serve per la creazione di immagini tramite PHP, sono infatti in grado di creare tramite semplice codice PHP numerosi tipi di immagini, quali jpeg, png, tiff ecc... Ovviamente per rimanere fedeli a tutti gli altri strumenti di questo progetto, le librerie GD sono open source. L'utilità di questa libreria è enorme, basti pensare alla creazione di grafici o report in tempo reale, analizzando i dati da un database.

Tornando al progetto, l'idea di base da seguire ora per realizzare automaticamente tutti i marker è la seguente:

- Accedere al db Aziende.
- Estrarre gli id.
- Creare un file immagine che contenga l'id appena estratto e salvare tale marker in modo tale da rendere più semplice localizzare la posizione in cui verrà posizionato.

Riguardo all'ultimo punto, è consigliato seguire uno schema simile a quello realizzato in precedenza quando si estraevano tutte le città di una determinata tavola, e cioè organizzare una serie di cartelle, così da gestire meglio i file appena generati. Più avanti verranno fatte considerazioni in merito. Di seguito un esempio di un marker creato tramite la procedura sopra indicata:

Illustrazione 12: Esempio di un marker creato tramite GD

É doveroso precisare che i marker possono essere interamente realizzati tramite GD, come nell'esempio mostrato poco fa, e quindi scegliere la loro forma, il colore e tanti altri parametri grafici solo utilizzando il codice, ma anche è possibile utilizzare un'immagine già esistente e modificarla tramite tale libreria.

Ad esempio se volessimo utilizzare questo atlante stradale per indicare dove sono presenti i distributori di benzina della Shell, potremmo semplicemente prendere il loro logo e aggiungere l'ID all'immagine seguendo la procedura sopra indicata. Segue un'immagine di quanto appena detto anche il diagramma di flusso di "*crea immagini id.php"* [A.A.5]:

Illustrazione 13: Esempio di marker creato partendo da un logo preesistente

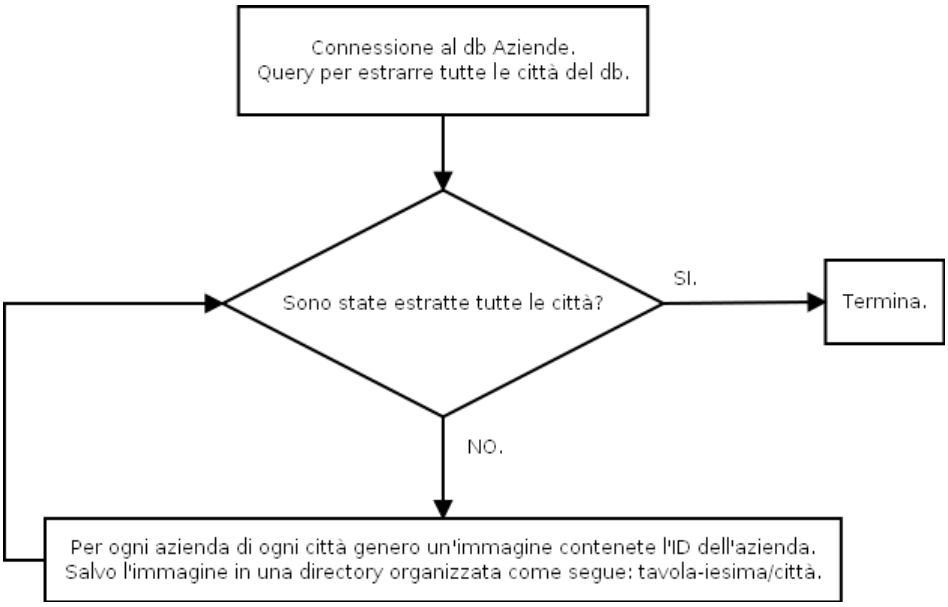

Illustrazione 14: Diagramma di flusso del codice "crea immagini id.php"

Dopo aver eseguito il codice il riscontro è immediato perché tutti i marker sono stati salvati all'interno di una serie di cartelle organizzate per tavole con al loro interno una suddivisione per città, così da permettere il raggruppamento di tutti i marker della città in questione.

L'immagine sottostante è un chiaro esempio che mostra una porzione di mappa dove sono stati applicati i marker di tre aziende. Volutamente è stato scelto di creare i marker trasparenti così da evitare di non mostrare le informazioni della mappa.

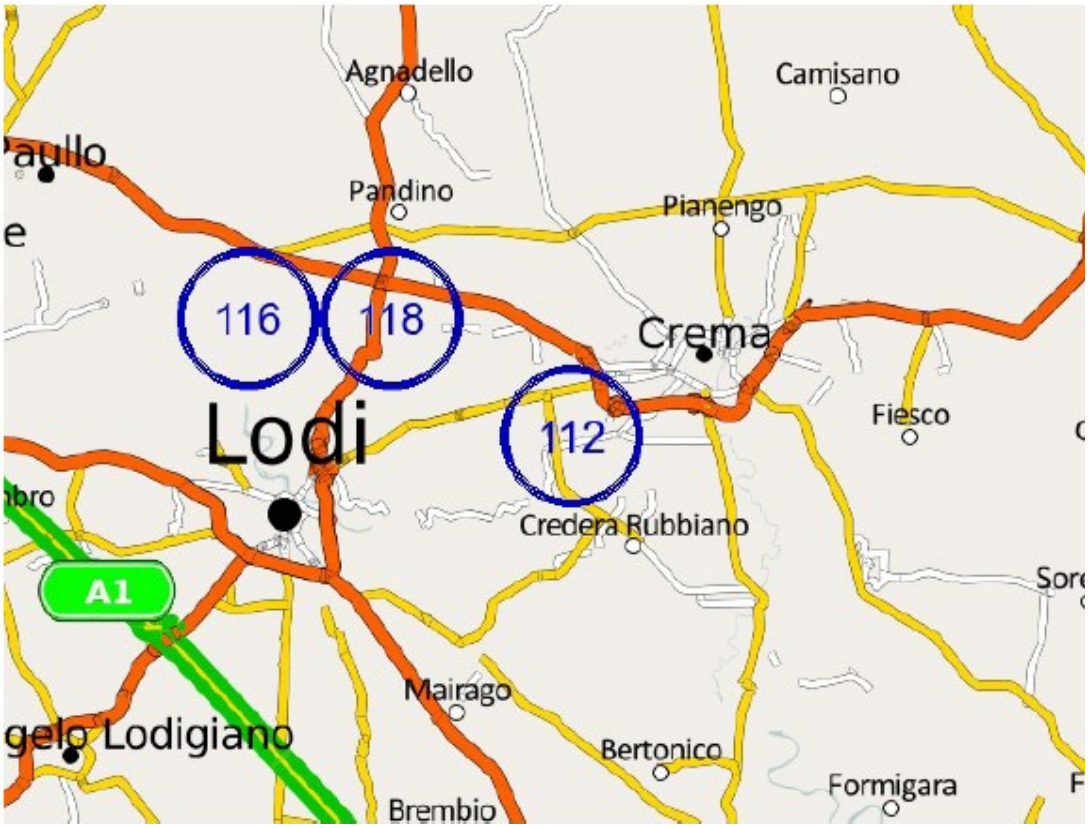

Illustrazione 15: Esempio di mappa con marker

Nel paragrafo seguente verrà presentato l'ultimo aspetto da realizzare per questo progetto e riguarderà il posizionamento dei marker.

4.8 Posizionamento dei marker

Giungiamo all'ultimo punto da sviluppare del progetto e si tratta anche di un argomento abbastanza delicato e soprattutto con alcune difficoltà.

Il problema principale del posizionamento dei marker è quello di poterli inserire in modo del tutto automatico. É un problema decisamente ostico da affrontare e ne verrà illustrato il perché. Un primo ragionamento che si può fare per automatizzare questa fase è quello di agire sul database di OSM, perché come oramai sappiamo perfettamente, al suo interno abbiamo per ogni elemento delle mappe la sua geo-posizione, frutto di lavoro da parte dei mappatori di OSM.

Questo induce sicuramente ad affermare che è possibile inserire i geo-dati delle aziende dentro questo database così da poter ottenere tutte le mappe con i marker solo con il semplice rendering della mappa, ovvero come è stato fatto nei capitoli precedenti. Questa affermazione non corrisponde assolutamente al vero perché, in primo luogo non abbiamo i geo-dati delle aziende, e di certo non sono facili da reperire, e in secondo luogo perché come accennato prima questa guida non indica l'esatta posizione delle aziende, ma visualizza sulla mappa un'indicazione della sua zona di competenza.

Inoltre, ricordando che Mapnik garantisce una zona di rispetto per ogni elemento visualizzato sulla mappa, per evitare che le informazioni si accavallino tra di loro generando solo confusione all'utilizzatore, sarebbe impossibile lasciare che un processo automatico gestisca questa situazione.

Il progetto però deve andare avanti e una soluzione a tal proposito va trovata. Probabilmente non sarà elegante, ma indubbiamente funziona e le prove effettuate lo garantiscono, quindi se non si può fare automaticamente si può fare manualmente.

Anche in questa circostanza una domanda è stata l'inizio di tutto un ragionamento, ovvero, ho le mappe, abbiamo i marker classificati per tavole e città e quindi è facile sapere in quale file immagine delle mappe posso inserire tali marker, ma come possiamo sapere con esattezza in che posizione si trova una certa città all'interno di una mappa?

Se non si sapesse nulla di geografia sarebbe impossibile sapere dove si trova esattamente una qualsiasi città e quindi qui nasce l'idea per dare supporto alla fase del posizionamento manualmente.

L'idea è quella di creare delle cartine di appoggio, ovvero, si potrà usufruire di una cartina semplificata che consenta di trovare facilmente le città che hanno delle aziende contenute nel db Aziende. Il primo passo è semplificare le cartine e quindi ancora una volta bisogna mettere mano al file osm.xml, e questa volta si devono eliminare tutti gli elementi che non sono classificati come città, snellendo quindi molto il file xml. Ora con questo file, quando si effettua il render delle mappe sappiamo che ci forniranno solamente tutte le città di una determinata zona geografica senza altre informazioni, ma lo scopo vero è mostrare solo quelle città che sono contenute nel db Aziende.

Tramite PHP allora si può modificare ulteriormente il file xml trattato poco prima. Il concetto che sta dietro a "*crea xml per le sole città nel db Aziende.php"* [A.A.6] è quello di inserire nel layer *placenames* del file xml, tutte le città del db Aziende, così da poter imporre a Mapnik soltanto le città che interessano.

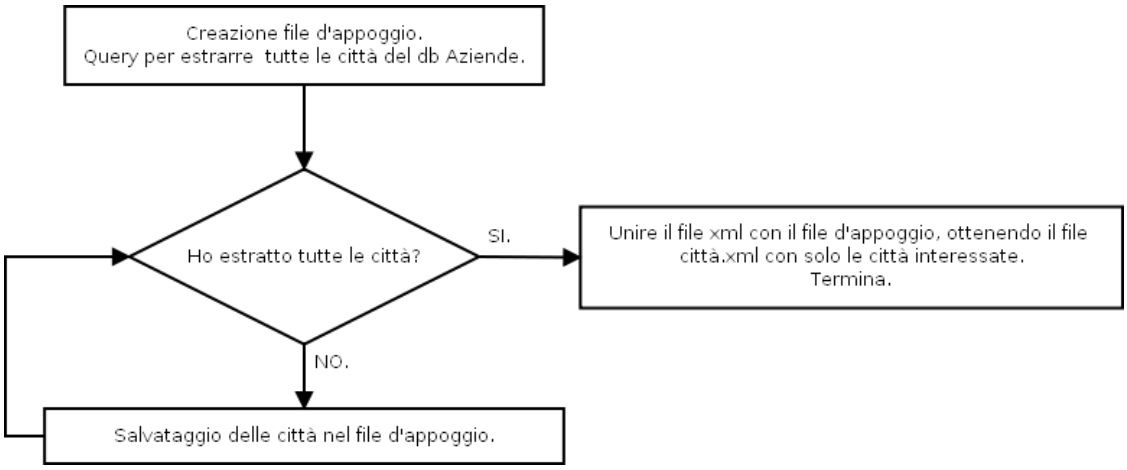

Illustrazione 16: Diagramma di flusso del codice "crea xml per le sole città nel db Aziende.php"

A questo punto si ottiene un file xml totalmente diverso rispetto ai precedenti, perché volutamente sono state tolte diverse informazioni così da consentire di rappresentare le sole città che a noi interessano. Ancora una volta sfruttando il programma scritto in Python *generazione automatica mappe.py*, modificando opportunamente il codice specificando a quale file xml deve fare riferimento, possiamo ottenere la serie delle mappe di appoggio.

Seguirà ora un esempio di mappa d'appoggio della zona del Veneto. Si può notare subito che qualsiasi informazione relativa alle autostrade, o provinciali, e a tutti gli altri elementi che non sono città o paesi, sono stati eliminati fornendo così una più semplice localizzazione delle città.

Illustrazione 17: Esempio di mappa d'appoggio. Rappresentazione della zona del Veneto.

Finalmente è possibile posizionare i marker in modo decisamente più agevole, infatti in accordo con il Tutor aziendale, si è ipotizzato un eventuale scenario dove una persona sarà incaricata di posizionare i marker.

Sfruttando le mappe d'appoggio l'incaricato al posizionamento degli identificatori di aziende, potrà guardare le mappe semplificate così da poter individuare facilmente le città che hanno delle aziende nei pressi del loro territorio, e quindi senza troppe difficoltà, con un programma di editing grafico come ad esempio Photoshop, si potrà posizionare il marker sulla mappa completa.

Il lavoro però dell'addetto al posizionamento dei marker non è finito qui, infatti bisogna rendere facile all'utente finale trovare l'azienda che gli interessa, per fare ciò dovremo aggiungere al db Aziende altri due campi per fornire la posizione di x e y all'interno di ogni mappa, così da poter realizzare un indice di tutte le aziende localizzabili per tavola e posizione su di essa.

Prima di tutto conviene realizzare una suddivisione interna delle mappe, come si era accennato prima al discorso della battaglia navale, dividendo ad esempio la mappa in quattro righe e quattro colonne come nell'esempio mostrato nell'illustrazione seguente:

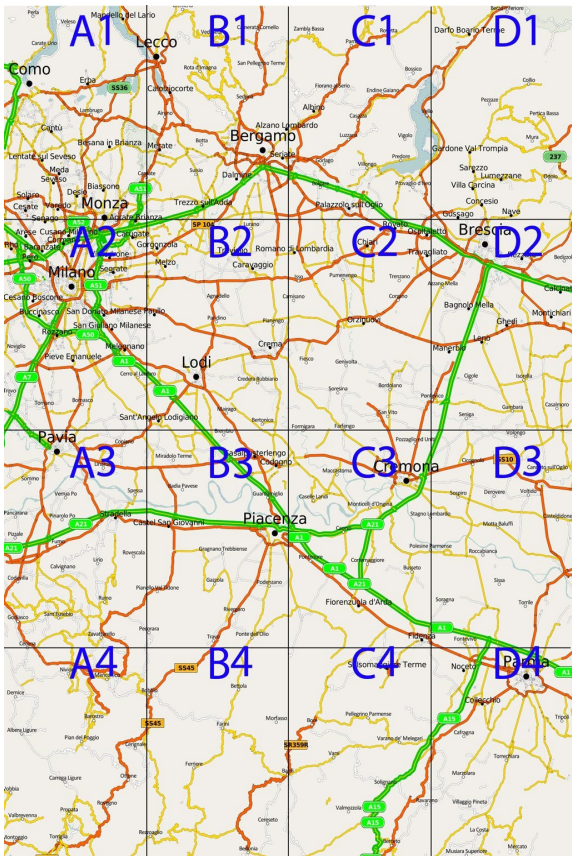

Illustrazione 18: Esempio di suddivisione interna delle tavole.

Tale suddivisione è possibile realizzarla comodamente con qualsiasi programma di editing grafico, l'illustrazione volutamente è stata realizzata in modo grossolano per permettere una spiegazione chiara del ragionamento.

Tornando al discorso del database, ogni volta che verrà inserito un pallino all'interno di una mappa bisognerà immettere i dati nel db Aziende della riga e della colonna, così che si avrà una classificazione completa delle informazioni. Perciò sarà opportuno modificare il db Aziende inserendo i campi riga e colonna ottenendo:

ID	PROVINCIA	CITTA'	NOME	INDIRIZZO	TELEFONO	TAVOLA	RIGA	COLONNA

Tabella 2: Rappresentazione della versione finale del database Aziende

A questo punto possiamo affermare di aver realizzato:

- Mappe personalizzabili graficamente di qualsiasi territorio, nel nostro caso dell'Italia.
- Mappe arricchite con informazioni utili per trovare le aziende.
- Un db Aziende che fornisce le informazioni basilari delle stesse ed è sfruttabile come indice per l'atlante.

5

RISULTATI E VALUTAZIONI

Qui si possono leggere i commenti sul lavoro finito, valutando e giudicando cosa si è ottenuto. I risultati prodotti sono da considerarsi positivi, secondo il Tutor aziendale che ha seguito il progetto da vicino, perché tutte le problematiche hanno ricevuto una equa trattazione, portando così a valutare e a sviluppare ogni aspetto. Indubbiamente il processo di questa tesi non è esente da errori o da possibili miglioramenti, infatti verrà introdotto un metodo alternativo per una parte del progetto.

Il fatto di aver realizzato o quanto meno valutato ogni punto richiesto per la riuscita del lavoro ha portato ad una consapevolezza maggiore conferendo più certezze sulla validità del lavoro svolto.

5.1 Risultati e Valutazioni

Fondamentalmente, tutte le tematiche di questo progetto sono state valutate, ma indubbiamente alcuni passaggi sono stati complicati da risolvere e in questi casi i risultati sono stati più efficaci invece che efficienti. Un esempio concreto riguarda il posizionamento dei marker, trattato nel Capitolo 4 paragrafo 8. Infatti, in quel frangente è stato sviluppato un metodo, che ora verrà introdotto, che si discostava dall'idea di partenza che indubbiamente era più elegante.

Il metodo che è stato scartato, parte dall'idea seguita per suddividere l'Italia in mappe diverse, infatti il concetto da seguire era quello di fare una suddivisione interna alle mappe. La logica di questa soluzione prevedeva, data una mappa, di effettuare una frammentazione così da poter classificare questi sotto territori.

Come si può notare, l'idea era quella di suddividere in modo equo la mappa, così da poter trattare ogni frammento come se fosse una mappa di dimensioni minori, per poter estrarre ancora una volta le bbox di questo sottoinsieme e quindi poter inserire nel db Aziende un campo che indicasse in quale porzione della mappa era possibile trovare una determinata città.

Il concetto si rifà molto al discorso della battaglia navale. Sono stati fatti diversi tentativi per realizzare, sempre con Python, un programma che permettesse di applicare il discorso appena fatto, ma il problema fondamentale è che le proiezioni dei geo-dati non consentivano un corretto rendering, sovrapponendo ogni sotto territorio. Questo generava dunque delle immagini che non corrispondevano alla suddivisione voluta. Purtroppo questo problema delle proiezioni non ha permesso di realizzare concretamente questi propositi, ma la logica del processo, a mio modesto parere, è da considerarsi valida.

Esistono diverse problematiche aperte riguardo al progetto descritto in questo testo, alcune sono agevoli da sviluppare altre addirittura dipendono da terzi, ovvero dai mappers di OSM.

Infatti, uno dei problemi principali è la mancanza o la non congruenza delle informazioni, perché OSM è gestito prevalentemente da "appassionati", i quali come sappiamo bene, armati con il loro GPS vanno in giro per il mondo e mappano i territori. Quindi finché alcune zone non verranno tracciate, gli utilizzatori e sviluppatori di OSM, non potranno mai usufruire di tali informazioni, e nel caso specifico di questo progetto la mancanza di tali informazioni precludono un livello di accuratezza elevato delle mappe.

Esistono altri problemi aperti, ma con una possibilità di soluzione più agevole, sono ad esempio il miglioramento grafico, che fondamentalmente dipende più dal gusto individuale.

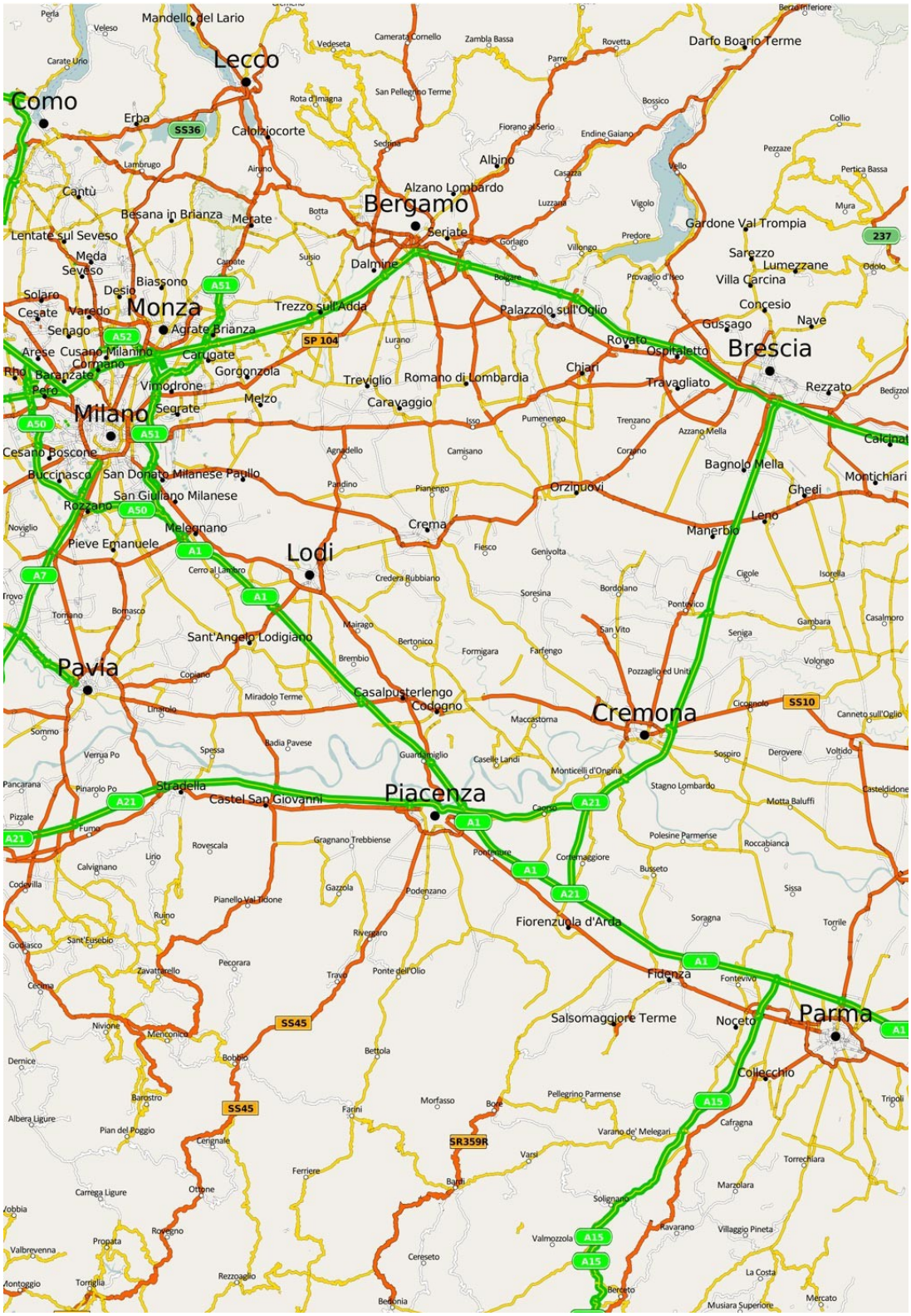

Illustrazione 19: Esempio di mappa prima della suddivisione interna

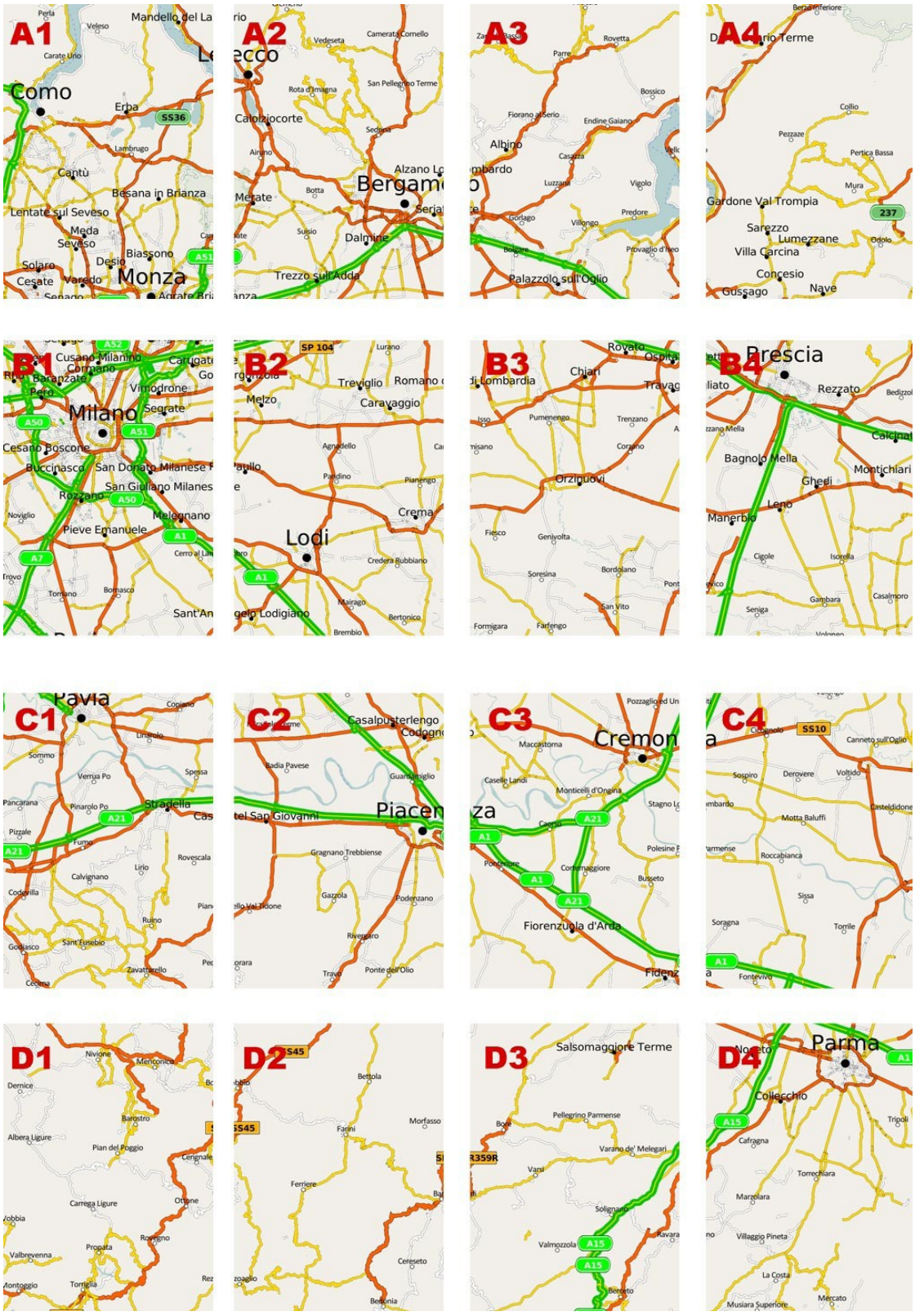

Illustrazione 20: Esempio di suddivisione interna di una mappa

6

CONCLUSIONI E DIREZIONI FUTURE

Capitolo conclusivo di questa testi, che offre la possibilità di una valutazione generale del lavoro svolto pensando anche a come è possibile migliorare il sistema o a quali sviluppi futuri può portare. In generale il progetto può considerarsi concluso perché l'obbiettivo primario era quello di verificare al validità del lavoro e come detto nel Capitolo 5 questo punto è stato soddisfatto, ma questo non preclude miglioramenti, infatti le informazioni di OSM sono in costante aumento e questo può portare solo ad un miglioramento delle cartine, perché il quantitativo di dati diventa sempre maggiore e quindi si ottengono mappe più accurate.

Ovviamente questo è solo un vantaggio per il progetto, ma il limite è imposto dal flusso delle informazioni che è indipendente da questo progetto, ma dipende solo dai mappers che devono dedicarsi alla mappatura di nuovi territori.

6.1 Conclusioni e direzioni future

Il progetto aveva lo scopo di produrre delle cartine stradali contenenti informazioni dettagliate su un ipotetico tipo di aziende e riproducibili su carta senza vincoli di licenza.

Sotto queste ipotesi, si è deciso di utilizzare OpenStreet Map, una mappa editabile completamente gratuita, e tanti altri software ad essa connessi ad esempio come PostgresSQL, Mapnik. Tutti di stampo open source.

Quindi dopo l'individuazione degli strumenti principali, è stato necessario trovare un metodo per cercare di dividere il territorio della nazione in analisi, nel caso specifico l'Italia.

Correlato al punto precedente, è stato necessario “stampare”, ovvero generare le mappe dei territori precedentemente divisi, previa modifica delle caratteristiche grafiche delle mappe stesse.

Essenziali sono stati considerati alcuni aspetti per facilitare il lavoro, come poter sapere quali sono le città che fanno parte del database che gestisce le aziende, o per poter generare automaticamente i marker per indicare la presenza di aziende in una determinata area. Il tutto è stato gestito tramite PHP e MySQL.

Anche se con buoni margini di miglioramento, questo progetto, sviluppato presso Vertigo SRL, ha lasciato soddisfatto il Tutor aziendale che ha seguito da vicino il lavoro. Si può infatti affermare che le tematiche che sono state proposte all'inizio di questo progetto, sono state tutte valutate, confermando la validità del progetto stesso. É comunque chiaro a tutti coloro che hanno preso visione di questo lavoro che alcune parti di esso, sicuramente possono migliorare, ma ciò non toglie che le riechieste poste all'inizio sono state valutate, studiate e risolte.

Un aspetto sicuramente molto interessante che potrebbe essere utile sviluppare è legato alla realizzazione delle mappe in formato SVG, perché il formato vettoriale indubbiamente può fornire un salto di qualità delle immagini indiscutibile. Basti pensare che questo progetto è rivolto alla realizzazione su carta e quindi un'immagine vettoriale sicuramente può fornire maggiori margini di accuratezza in stampa rispetto ad una semplice immagine bitmap.

Discorso interessate, ma richiede prima di tutto di migrare tutto il progetto sotto Linux, perché OSM e nello specifico Mapnik, offrono il supporto dei file SVG solo sotto questo sistema operativo.

Questo non è assolutamente un problema. Il vero problema degli SVG è legato alla quantità di informazioni che contengono, infatti una delle nostre mappe realizzate in bitmap, pesa mediamente sui 4 megabyte, mentre la stessa mappa in SVG arriva tranquillamente a pesare addirittura 16 megabyte, rendendo molto difficoltoso lavorarci sopra, perché i migliori software attualmente disponibili, come ad esempio Inkscape(open source) e Illustrator(commerciale), non riescono a sostenere la mole di informazioni e vanno in crash.

Per esperienza personale, Illustrator nemmeno riesce ad aprire la mappa, mentre Inkscape consente addirittura di effettuare qualche piccola modifica, ma non da garanzie valide sulla possibilità di salvare tali modifiche perché va in crash. Purtroppo anche la comunità di OSM riconosce il problema che comportano gli SVG e ad oggi non è ancora stata trovata una soluzione. Il giorno che si riuscirà a gestire le mappe contenenti un elevato numero di informazioni in SVG, sarà sicuramente da considerarsi un salto in avanti sostanziale.

Altre modifiche o miglioramenti che dir si voglia, ad esempio può essere quello di fornire informazioni ancor più dettagliate alle mappe inserendo i dati dei paesi

confinanti all’Italia, perché attualmente per scelte progettuali si è scelto di non inserire i dati degli altri paesi.

Oppure, un'aggiunta interessante potrebbe essere quella di realizzare mappe dettagliate anche per le grandi città, come ad esempio Milano, fornendo in modo molto più accurato l'esatta posizione delle aziende.

BIBLIOGRAFIA

M. Napolitano (2009). G.I.S.: Geographical Information Systems. Rapporto interno, Istituto Ricerca Scientifica e Tecnologica, Italia

M. Napolitano (2009). GIS e Software libero. Rapporto interno, Istituto Ricerca Scientifica e Tecnologica, Italia

S. Romagnolo (2006). Google Terza Edizione, Apogeo, Italia

F. de Virgilio (2009). Cartografia e GIS open source: da OpenStreet Map a GeoBase, Università degli Studi di Bari Dipartimento Geomineralogico, Italia

G. de Rossi (2009). OpenStreet Map, presentazione OSMIT 2009 Trento, Italia

A. Pavlenko (2007). Open Source renders the world, Society of Cartographers 43rd annual summer school, University of Portsmouth, United Kingdom

A. Pavlenko (2006). Mapnik: XML Schema Reference, University of Portsmouth, United Kingdom

L. Delucchi (2009). Introduzione a Mapnik, presentazione OSMIT 2009 Trento, Italia

A. Lodovisi (1996). Storia della cartografia, Patron Editore, Italia

S. Sedita (2009). Concetti di base sui G.I.S., Università Degli Studi di Padova, Italia

M. Caprioli (2009). Cartografia Numerica, Politecnico di Bari, Italia

M. Caprioli (2009). GPS, Politecnico di Bari, Italia

S. Mele e P. Strolin (2008). Il Sistema di PosizionamentoGlobale (GPS), INFN, Italia

K. Clarke (2001). Getting Started with Geographic Information Systems, New Jersey.

M. Lutz (2008). Imparare Python, HOPS, Italia.

A. Rossato (2009). Il licenziamento del dato geografico libero, presentazione OSMIT 2009 Trento, Italia

N. Rigacci (2010). OpenStreetMap Gli strumenti per il sorpasso, OSMIT 2010, Genova, Italia.

R. Gerardo (2010). OpenStreetMap mappe a contenuto libero, Linux Day, Italia.

A. Piemonte(2008). GIS Sistemi informativi geografici, Università di Pisa, Italia

A. Piemonte(2008). Gestione della base di dati GIS, Università di Pisa, Italia

OpenStreet Map http://wiki.openstreetmap.org/wiki/WikiProject_Italy

Creative Commons http://creativecommons.org

GNU GPL http://www.gnu.org/copyleft/gpl.html

APPENDICE A

In questa parte del testo seguiranno tutti i codici che sono stati menzionati all'interno del documento. Verranno presentati in ordine a seconda di come sono stati citati all'interno del testo.

A.A.1 Generazione automatica tavole.py

```
from mapnik import *
import sys, os

if __name__ == "__main__":
    home = "C:/mapnik_0_6_1/osm"
    mapfile = home + "/ricky_template_nuovi_colori4.xml"
    tile_dir = home + "/tiles/"

    # Dimensione immagine

    z = 1
    imgx = 1917 * z
    imgy = 2778 * z

    num_tavola = 4;

    shift = 0.215606

    # coordinate di x di partenza
    x_1 = 6.176
    x_2 = 7.599

    # coordinate di y di partenza
    y_1 = 45.771
    y_2 = 46.878

    # valori di incremento per le prossime mappe
    incr_x = 1.423
    incr_y = 1.107
```

```
        ll = (x_1,y_1,x_2,y_2)

        def func():
            print '>> Fine settore, cambio di variabili...'
            print '\nx1 = %2.4f' %x_1
            print '\nx2 = %2.4f' %x_2
            print '\ny1 = %2.4f' %y_1
            print '\ny2 = %2.4f\n' %y_2

        dest = (home + "/tavole/")
        if not os.path.isdir(dest):
                os.mkdir(dest)

        for i in range (0,38):
             map_uri = dest + "auto_tavola %d.png" %(i)
             m = Map(imgx,imgy)
             load_map(m,mapfile)
                   prj = Projection("+proj=merc +a=6378137 +b=6378137
+lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null
+no_defs +over")
             c0 = prj.forward(Coord(ll[0],ll[1]))
             c1 = prj.forward(Coord(ll[2],ll[3]))
             bbox = Envelope(c0.x,c0.y,c1.x,c1.y)
             m.zoom_to_box(bbox)
             im = Image(imgx,imgy)
             render(m, im)
             view = im.view(0,0,imgx,imgy) # x,y,width,height
             view.save(map_uri,'png')
             print 'Generata tavola numero %d' %(i)

             # mantengo le y costanti e traslo l'area lungo x

             x_1 = x_2
             x_2 = x_2 + incr_x

             # devo resettare i valori delle coordinate quando raggiungo
             # i confini dell'Italia lungo x
             if(i == 5 or i == 11):
                 x_2 = x_2 - (6*incr_x)
                 x_1 = x_1 - (6*incr_x)
                 y_2 = y_1
```

```
    y_1 = y_2 - incr_y
    func();

# sempre passo prima
if(i == 17):
    x_2 = x_2 - (4*incr_x) + shift
    x_1 = x_1 - (4*incr_x) + shift
    y_2 = y_1
    y_1 = y_2 - incr_y
    func();

if(i == 21):
    x_2 = x_2 - (3*incr_x) + shift
    x_1 = x_1 - (3*incr_x) + shift
    y_2 = y_1
    y_1 = y_2 - incr_y
    func();

if(i == 25):
    x_2 = x_2 - (2*incr_x) - shift
    x_1 = x_1 - (2*incr_x) - shift
    y_2 = y_1
    y_1 = y_2 - incr_y
    func();

if(i == 29):
    x_2 = x_2 - (3*incr_x) - shift
    x_1 = x_1 - (3*incr_x) - shift
    y_2 = y_1
    y_1 = y_2 - incr_y
    func();

if(i == 32):
    x_2 = x_2 - (3*incr_x) - shift
    x_1 = x_1 - (3*incr_x) - shift
    y_2 = y_1
    y_1 = y_2 - incr_y
    func();

if(i == 34):
    x_2 = x_2 - (4*incr_x)
    x_1 = x_1 - (4*incr_x)
```

```
            y_2 = y_1
            y_1 = y_2 - 6*shift
            func();

        ll = (x_1,y_1,x_2,y_2)

    else:
        print 'Fine'
```

A.A.2 Bbox città.py

```
from mapnik import *
import sys, os

if __name__ == "__main__":
    home = "C:/mapnik_0_6_1/osm/"
    mapfile = home + "/citta.xml"
    tile_dir = home + "/tiles/"

    # Dimensione immagine

    z = 1
    imgx = 1917 * z
    imgy = 2778 * z

    num_tavola = 4;

    shift = 0.215606

    # coordinate di x di partenza
    x_1 = 6.176
    x_2 = 7.599

    # coordinate di y di partenza
    y_1 = 45.771
    y_2 = 46.878

    # valori di incremento per le prossime mappe
    incr_x = 1.423
    incr_y = 1.107
```

```
        ll = (x_1,y_1,x_2,y_2)

        j = 0
        miofile = open("auto_bbox.txt", "w")


        for i in range (0,38):
             m = Map(imgx,imgy)
             load_map(m,mapfile)
                   prj  =  Projection("+proj=merc  +a=6378137  +b=6378137
+lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null
+no_defs +over")
             c0 = prj.forward(Coord(ll[0],ll[1]))
             c1 = prj.forward(Coord(ll[2],ll[3]))
             bbox = Envelope(c0.x,c0.y,c1.x,c1.y)

             out = (str(bbox))
             out = out.replace("Envelope(", "")
             out = out.replace(")", "")
             for k in range (0, len(out)):
                 if(out[k] == "," and (j == 0)):
                     text = out[0:k] + ' ' + out[k+1:]
                     j = j + 1
             print text

             # mantengo le y costanti e traslo l'area lungo x

             x_1 = x_2
             x_2 = x_2 + incr_x

             # devo resettare i valori delle coordinate quando raggiungo
             # i confini dell'Italia lungo x
             if(i == 5 or i == 11):
                 x_2 = x_2 - (6*incr_x)
                 x_1 = x_1 - (6*incr_x)
                 y_2 = y_1
                 y_1 = y_2 - incr_y


             # sempre passo prima
             if(i == 17):
```

```
        x_2 = x_2 - (4*incr_x) + shift
        x_1 = x_1 - (4*incr_x) + shift
        y_2 = y_1
        y_1 = y_2 - incr_y


    if(i == 21):
        x_2 = x_2 - (3*incr_x) + shift
        x_1 = x_1 - (3*incr_x) + shift
        y_2 = y_1
        y_1 = y_2 - incr_y


    if(i == 25):
        x_2 = x_2 - (2*incr_x) - shift
        x_1 = x_1 - (2*incr_x) - shift
        y_2 = y_1
        y_1 = y_2 - incr_y

    if(i == 29):
        x_2 = x_2 - (3*incr_x) - shift
        x_1 = x_1 - (3*incr_x) - shift
        y_2 = y_1
        y_1 = y_2 - incr_y


    if(i == 32):
        x_2 = x_2 - (3*incr_x) - shift
        x_1 = x_1 - (3*incr_x) - shift
        y_2 = y_1
        y_1 = y_2 - incr_y


    if(i == 34):
        x_2 = x_2 - (4*incr_x)
        x_1 = x_1 - (4*incr_x)
        y_2 = y_1
        y_1 = y_2 - 6*shift


    ll = (x_1,y_1,x_2,y_2)
```

```
        miofile.close()
        print "File generato.\nIl file è disponibile nella dir: " + home
```

A.A.3 Da bbox a tavole.php

```
<?
//Connessione al DB di postgres
$conn = pg_connect('dbname=gis user=Ricky password=root');

if(!$conn) {
    die('Connessione fallita !<br />');
}
//Sequenza di stringhe per la creazione della query che estrae tutte le città
//di una determinata bbox
$str = 'select name from planet_osm_point where way && SetSRID(\'BOX3D(';
$str2 = ')\'::box3d,900913) and (place = \'city\' or place = \'town\' or    place = \'village\') order by name asc';
//Dati per la gestione del file contenente tutte le bbox
$filename = "auto_bbox.txt";
$fp = fopen($filename, "r");
$bytes = filesize($filename);
$i = 0;
$ext = '.txt';
while(!feof($fp)){
    //Creazione di un nuovo file per contenere il risultato della query
    $new_filename = 'tavola ';
    $filename_result = $new_filename.$i.$ext;
    $fpw = fopen($filename_result, "w");
    //La bbox sarà presa di volta in volta dal file $filename
    $bbox = fgets($fp, $bytes);
    //Unione delle stringhe per creare la query
    $query_pg = $str.$bbox.$str2;
    if(!$query = pg_query($query_pg))die("Errore nella query");
        while($row = pg_fetch_array($query)) {
            $city_name = "{$row['name']}";
            fwrite($fpw, $city_name."\r\n");
            }
    fclose($fpw);
    $i = $i + 1;
```

```
    }
  fclose($fp);
  pg_close($conn);?>
```

A.A.4 Inserisci tavole nel db Aziende.php

```
  <?
  include 'config.php';
  include 'opendb.php';
  $dir = 'C:/Apache2.2/htdocs/www/IDtoIMG/tavole/';
  $i = 0;
  $file = 'tavola ';
  $ext = '.txt';
  $count = 0;
  while($i < 38)
      {
      $filename = $dir.$file.$i.$ext;
      echo "<strong><BR>".$filename."<BR><BR></strong>";
      //apro il file della directory
      $fp = fopen($filename, "r");
      $bytes = filesize($filename);
      while(!feof($fp)){
            $str_city = fgets($fp, $bytes);
            if(strcmp($str_city, "ï»¿") > 0)
                  continue;
            $str_city = addslashes($str_city);
            $str_city = trim($str_city);
            $query = "SELECT id, rtrim(citta) as citta, tavola
            FROM atlante WHERE citta like '$str_city'";
            $result = mysql_query($query) or die('Error ,query failed');
            while($row = mysql_fetch_array($result)){
                  $count++;
                  $id = "{$row['id']}";
                  echo $count.": inserisco nel db il valore della
tavola                  {$row['citta']} <BR>";
                  $ins = "UPDATE atlante SET tavola = '$file$i' WHERE
id =                    '$id'";
                  mysql_query($ins) or die('Error ,query failed');
                  }
      }
```

```
        $i++;
        fclose($fp);
    }include 'closedb.php';?>
```

A.A.5 Crea immagini id.php

```
    <?
    include 'config.php';
    include 'opendb.php';
    $query = "SELECT id FROM atlante";
    $result = mysql_query($query);
    $i = 0;
    $nome = 'cerchio';
    $est = '.png';
    $path = "C:/Apache2.2/htdocs/www/IDtoIMG/immagini/";
    if(!(is_dir($path)))
        mkdir($path);
    $query = "SELECT id, rtrim(citta) as citta, tavola FROM atlante";
    $result = mysql_query($query) or die('Error ,query failed');
    while($row = mysql_fetch_array($result)){
        if(strcmp("{$row['tavola']}", "0") == 0)
              continue;
        $img = ImageCreate(59,59);
        $darkblu = imagecolorallocate($img,0,0,192);
        $white = imagecolorallocate($img,255,255,255);
        imagefill($img,0,0,$darkblu);
        imagearc($img,29,29,58,58,0,360,$white);
        imagestring($img,1,20,25,"{$row['id']}",$white);
        $tavola = rtrim("{$row['tavola']}");
        $citta = rtrim("{$row['citta']}");
        if(is_dir($path.$tavola."/")){
              if(is_dir($path.$tavola."/".$citta."/")){
                        imagepng($img,$path.$tavola."/".$citta."/".

        $nome."{$row['id']}".$est);
                        echo "<BR>Aperta dir: ".$citta."/ e salvato il
file ".                                 $nome."{$row['id']}".
$est."<BR>";
                        imagedestroy($img);
                        $i++;
                        continue;
```

```
                    }
   else{
       mkdir($path.$tavola."/".$citta."/",0777) or die ("Could not make

       directory");
       imagepng($img,$path.$tavola."/".$citta."/".
       $nome."{$row['id']}".$est);
       echo "<BR>Aperta dir: ".$tavola.$citta."/"." e salvato il file ".
                                              $nome."{$row['id']}".
$est."<BR>";
       imagedestroy($img);
       $i++;
       continue;
       }
   }
   mkdir($path.$tavola."/") or die ("Could not make directory");
   mkdir($path.$tavola."/".$citta."/") or die ("Could not make
directory");
   echo "<BR>Creata dir: ".$path.$tavola.$citta." e salvato il file ".
                                              $nome."{$row['id']}".
$est."<BR>";
   imagepng($img,$path.$tavola."/".$citta."/".$nome."{$row['id']}".
$est);
   imagedestroy($img);
   $i = $i + 1;
   }
   include 'closedb.php';
   ?>
```

A.A.6 Crea xml per le sole città nel db Aziende.php

```
<?
include 'config.php';
include 'opendb.php';
$dir = 'C:/Apache2.2/htdocs/www/IDtoIMG/';
$fp = fopen("query citta.txt", "w");
$content = '(select way,place,capital,name,ref from planet_osm_point
where       place is not null and (';
$count = 0;
$i = 0;
fwrite($fp, $content);
$query = "SELECT distinct rtrim(citta) as citta FROM atlante";
$result = mysql_query($query) or die('Error ,query failed');
$resultcount = mysql_query($query) or die('Error ,query failed');

while($row = mysql_fetch_array($resultcount))
    $count = $count + 1;
while($row = mysql_fetch_array($result)){
    $citta = pg_escape_string("{$row['citta']}");
    if($i != $count)
          fwrite($fp,"name like '".$citta."' or \r\n\t\t");
    if($i == $count - 1)
          fwrite($fp,"name like '".$citta."') \r\n\t\t");
    $i = $i + 1;
    }
fclose($fp);
$fp = fopen("query citta.txt", "r");
$size = filesize("query citta.txt");
$contentfile = fread($fp, $size);
$filexml = "citta.xml";
$fpxml = fopen($filexml, "r");
$sizexml = filesize($filexml);
$contentxml = fread($fpxml, $sizexml);
fclose($fpxml);
$fpxml = fopen("file generato citta.xml", "w");
$str1 = '<Layer name="placenames" status="on" srs="+proj=merc
+a=6378137
+b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m
+nadgrids=@null +no_defs +over">';
```

```
   $str2 = '<Parameter name="table">';
   $str3 = '(select way,place,capital,name,ref from planet_osm_point
where       place is not null)';
   $str4 = 'as placenames</Parameter>';
   $contentxml = str_replace($str2.$str3, $str2.$contentfile.")",
$contentxml);
   fwrite($fpxml,utf8_encode($contentxml));
   fclose($fp);
   fclose($fpxml);
   include 'closedb.php';
   ?>
```

I codici che seguiranno ora, non sono stati citati nel testo, ma nei sorgenti presenti in questa Appendice se ne fa uso frequentemente, perché servono per le connessioni con i vari database.

A.A.7 Config.php

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'root';
$dbname = 'atlante'; //nome del database
?>
```

A.A.8 Closedb.php

```
<?php
mysql_close($conn);
?>
```

A.A.9 Opendb.php

```
<?php
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error
                            connecting to mysql');
mysql_select_db($dbname);
?>
```

A.A.10

```
<?php
$host = "localhost";
$user = "Ricky";
$pass = "root";
$db = "gis";

$conn = @pg_connect('dbname=$db user=$user password=$pass');

if(!$conn) {
    die('Connessione fallita !<br />');
} else {
    echo 'Connessione riuscita !<br />';
}
?>
```

APPENDICE B

Nell'Appendice B, è possibile trovare una guida operativa di come poter far funzionare tutti i sorgenti e gli strumenti mostrati e utilizzati in questo progetto. Il Capitolo 3, propone un Tutorial che è senza ombra di dubbio il primo passo da affrontare per poter preparare la macchina al lavoro, quindi invito il lettore a seguire le istruzioni riportate nel suddetto Capitolo. Bisogna precisare inoltre che tutti i codici utilizzati dovranno essere adattati secondo le proprie esigenze, ovvero bisognerà controllare che i percorsi contenuti nel codice corrispondano.

Esempio:

```
if __name__ == "__main__":
home = "C:/mapnik_0_6_1/osm" CONTROLLARE CHE CORRISPONDA
mapfile = home + "/file generato citta.xml"
tile_dir = home + "/tiles/"
```

Fatto ciò, seguirà ora una serie di programmi e strumenti che consiglio al lettore di utilizzare:

- Editor xml: WmHelp XmlPad scaricabile gratuitamente presso http://www.wmhelp.com/download.htm
- Editor PHP: PHP Editor 2.22 programma gratuito http://download.html.it/software/vedi/3648/php-editor/
- Editor grafico: Photoshop, Fireworks (a pagamento), GIMP(open source)
- Piattaforma server Web: Apache http://www.apache.org/dyn/closer.cgi
- DBMS: MySql http://www.mysql.it/downloads/
- Linguaggio Php: Php 5 http://www.php.net/downloads.php
- Gestore DMS: PhpMyAdmin http://www.phpmyadmin.net/home_page/downloads.php
- Sito di supporto: http://maps.cloudmade.com/ richiede registrazione gratuita

Queste sono solo dei consigli che mi sento di dare riguardo ai programmi da utilizzare, se il lettore preferisce usare altri programmi è libero di farlo.

Riguardo all'installazione e alla configurazione di tali software, si lascia al lettore il compito perché non è lo scopo di questa guida spiegare come eseguire queste operazioni, da notare che non si tratta di operazioni ne troppo lunghe o laboriose da fare. Procediamo ora e vediamo come eseguire il lavoro:

1. Installare i software sopracitati.
2. Seguire i passi spiegati nel Capitolo 3: Tutorial.
3. Copiare all'interno della cartella di installazione di Mapnik, generalmente **c:\mapnik_0_6_1\osm** tutti i sorgenti py forniti con questo documento. I file in questione sono:

    - Generazione automatica tavole.py
    - Bbox città.py
    - Generazione automatica solo città.py

    Inoltre, copiare anche i file xml sempre forniti con questo manuale:

    - Template.xml
    - Citta.xml

4. Aprire la console di Python **Start -> Programmi -> Python 2.5 -> IDLE(Python GUI)** e aprire il file *generazione automatica tavole.py*. Una volta aperto il codice premere F5. Terminato il programma verrà mostrato il percorso della directory contenente tutti i file immagine contenenti le tavole.
5. Eseguire ora, sempre seguendo il metodo indicato al punto precedente, il file *bbox città.py*, anche qui verrà indicato il percorso del file generato, ovvero *auto_bbox.txt*.
6. Creare ora una cartella,con il nome *file php*, all'interno della cartella di installazione di Apache, di solito la si può trovare qui**C:\Apache2.2\htdocs\www\.**
7. Copiare dentro la cartella appena creata i seguenti file:

    - Config.php
    - Closedb.php
    - Opendb.php
    - Pg_conn.php
    - Da bbox a tavole.php
    - Inserisci tavole in db Aziende.php
    - Crea immagini id.php

- Crea xml per le sole città nel db Aziende.php

8. Copiare il file *auto_bbox.txt* nella cartella contenente tutti i file php.
9. Eseguire il codice da *bbox a tavole.php* attraverso un qualunque browser, digitando nella barra degli indirizzi il seguente indirizzo http://127.0.0.1/www/file php/ e selezionando il file in questione, oppure è possibile eseguire il codice attraverso *Php Editor 2.22*, infatti cliccando nella barra del menu, alla voce Execute -> Internet Explorer (potrebbe comparire il nome del vostro browser predefinito). Eseguito il codice nella cartella **C:\Apache2.2\htdocs\www\file php**, si può notare la creazione di una nuova cartella, **tavole**, contenente tanti file di testo nominati in modo crescente.
10. Creare il db Aziende tramite PhpMyAdmin, oppure importarlo
11. Creato il database Aziende eseguire il codice presente nella cartella **C:\Apache2.2\htdocs\www\file php**, *inserisci tavole in db Aziende.php*. I file di testo creato al Punto 9 verranno letti e inseriranno nel db Aziende tutte le corrispondenze delle tavole. Il riscontro si può avere guardando il db Aziende.
12. Eseguire il codice *crea immagini id.php* come spigato prima, al termine verrà creata una cartella *immagini* suddivisa ulteriormente in cartelle contenente tutte le immagini degli id delle aziende.
13. Tornare alla cartella **c:\mapnik_0_6_1\osm** e copiare il file *Citta.xml* nella cartella **C:\Apache2.2\htdocs\www\file php\.**
14. Eseguire il codice *crea xml per le sole città nel db Aziende.php*.
15. Al termine copiare il file *file generato citta.xml* in **c:\mapnik_0_6_1\osm**
16. Eseguire il codice *generazione automatica solo città.py* ottenendo così la cartella *solo città*, avente tutte le tavole d'appoggio.