UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica

Tesi di Laurea

Progettazione e sviluppo di un'applicazione GIS di supporto ad interventi medici nel Terzo Mondo

Relatore: Prof. Massimo Rumor

Correlatore: Dott. Sandro Savino

Laureando: Luca Di Ielsi

500102-IF

Anno Accademico 2012-2013

INDICE

1. INTRODUZIONE

1.1 Scopo della tesi

In seguito all'incontro con il dottor Giovanni Putoto, responsabile della programmazione di Medici con l'Africa Cuamm, è stata valutata la possibilità di fornire un aiuto concreto all'azione umanitaria che attualmente l'ONG con sede a Padova compie in alcuni Stati dell'Africa, tramite l'impiego di strumenti informatici. La tesi tratta di come sia stato affrontato e realizzato il lavoro, partito da un colloquio iniziale con il quale si sono raccolti i dati disponibili e i requisiti da soddisfare, proseguito con la loro analisi e con il progetto di vari compiti che potessero essere di aiuto al Cuamm per aspetti quali l'organizzazione e la gestione delle risorse a disposizione, e terminato con la realizzazione di questi ultimi.

1.2 Il CUAMM[1]

Nata nel 1950, Medici con l'Africa Cuamm (Collegio Universitario Aspiranti e Medici Missionari) è la prima ONG in campo sanitario riconosciuta in Italia e la più grande organizzazione italiana per la promozione e la tutela della salute delle popolazioni africane.

Figura 1 - Il logo del CUAMM

[1] http://www.mediciconlafrica.org

Realizza progetti a lungo termine in un'ottica di sviluppo, intervenendo con questo approccio anche in situazioni di emergenza, per garantire servizi di qualità accessibili a tutti.

A tale scopo si impegna nella formazione in Italia e in Africa delle risorse umane dedicate, nella ricerca e divulgazione scientifica in ambito tecnico di cooperazione sanitaria, nell'affermazione del diritto umano fondamentale alla salute per tutti, anche dei gruppi più marginali, diffondendo nelle istituzioni e nell'opinione pubblica i valori della solidarietà e della cooperazione tra i popoli, della giustizia e della pace.

In 60 anni di storia…

- 1292 operatori, tra medici, paramedici e tecnici, hanno prestato servizio specialmente nei paesi dell'Africa sub-Sahariana con un periodo medio in servizio di 3 anni e 4 mesi;
- 950 studenti sono stati ospitati nel collegio, 680 italiani e 270 studenti ospitati da 35 paesi del Sud del mondo;
- 160 i programmi realizzati in collaborazione con il Ministero degli Affari Esteri, Unione Europea e varie agenzie internazionali;
- 214 le strutture sanitarie seguite, di cui 35 ristrutturate o costruite ex novo e attrezzate;
- 40 i paesi di intervento in Asia, America Latina, Medio Oriente e soprattutto Africa.

Oggi…

Medici con l'Africa Cuamm è attualmente presente in Angola, Etiopia, Mozambico, Sud Sudan, Tanzania, Uganda con:

- 78 operatori: 46 medici, 4 paramedici, 28 tecnici e amministrativi
- 37 progetti di cooperazione principali e un centinaio di micro-realizzazioni di supporto, con i quali appoggia:
- 15 ospedali
- 25 distretti (per attività di sanità pubblica, assistenza materno-infantile, lotta all'Aids, tubercolosi e malaria, formazione)
- 3 centri di riabilitazione motoria
- 4 scuole infermieri
- 3 Università (in Uganda, Mozambico ed Etiopia)

2. RACCOLTA E ANALISI DI REQUISITI E DATI

2.1 Raccolta e analisi requisiti

Dalla lista di requisiti stilata dopo il colloquio è emerso come, attraverso l'utilizzo di funzioni di un software GIS definite ad hoc e di un database accessibile via web, si sarebbe potuta migliorare la gestione di situazioni rilevanti, facilitando la pianificazione e il lavoro sul campo di medici e volontari. L'attività è stata focalizzata in una regione geografica piuttosto circoscritta, il Sud Sudan, ma è facilmente estendibile ad altre regioni oppure a regioni più estese.

Figura 2 - Il Sud Sudan

La prima richiesta inoltrata dal CUAMM è stata quella di produrre mappe tematiche del Sud Sudan e specificatamente della regione di Yirol, al fine di evidenziare la dislocazione e l'operatività delle strutture sanitarie presenti nel territorio in relazione ai villaggi da esse serviti. Queste mappe sono state utilizzate a sussidio di una spedizione nelle zone interessate.

Successivamente è emersa l'esigenza di strutturare l'organizzazione dei dati in possesso e di futura raccolta riguardanti:

- il tema delle nascite, sia presso i villaggi che presso i centri sanitari, con particolare interesse per le informazioni relative al neonato e alla madre, per localizzare le situazioni di criticità e avere la possibilità di produrre statistiche precise, utili per la programmazione di eventuali interventi migliorativi;
- l'anagrafica degli operatori sanitari (specializzati e non), in modo da gestire al meglio le risorse umane disponibili sul territorio;
- il sistema di vaccinazione, archiviando informazioni sui tipi di vaccinazioni somministrate, sui villaggi o i centri sanitari in cui si effettuano, sul numero di soggetti vaccinati, ecc..

Tale requisito è stato soddisfatto con la progettazione e la realizzazione di un database e della relativa interfaccia web che ne permette l'amministrazione anche da remoto.

Infine, per favorire la pianificazione della distribuzione di vaccini e/o reti anti zanzare presso una serie di località presso cui l'ONG opera, è stata proposta una soluzione al problema della scelta del percorso stradale più conveniente con la realizzazione di un'estensione per un'applicazione GIS. Questa implementa un algoritmo che risolve, nella pratica, la situazione conosciuta in Ricerca Operativa con il nome di *Travelling Salesman Problem*, cioè data una serie di località, trova il percorso di lunghezza minima necessario per la loro visita, una volta sola, facendo ritorno al luogo da cui si era partiti.

2.2 Raccolta e analisi dati

Durante l'incontro il responsabile del Cuamm, oltre ai requisiti, ha fornito alcuni dati, sotto forma di *file shape*, contenenti informazioni (geografiche e non solo) riguardanti il

Sud Sudan, su cui è stato impostato l'intero lavoro. Del pacchetto, che comprendeva svariate voci, sono state selezionate le occorrenze più utili allo scopo, e cioè:

- *Roads* (le principali strade)
- *Population by county* (geografia delle contee e statistica della popolazione)
- *Towns* (i centri urbani)
- *Settlements* (i villaggi)
- *Health facilities* (le strutture sanitarie)

2.3 Il *file shape*

Il *file shape* è un diffuso formato vettoriale per GIS, sviluppato da ESRI[2] e che di fatto è diventato uno standard per il dato spaziale vettoriale non topologico (i.e. non può memorizzare informazioni quali l'adiacenza, la connessione, la prossimità, la coincidenza, etc.).

Ad ogni *file shape* (con estensione *.shp*), che contiene le geometrie vere e proprie, è associato un file con estensione *.shx* (che contiene l'indice delle geometrie) e uno con estensione *.dbf* (che rappresenta il database degli attributi delle geometrie), mentre opzionalmente possono essere associati file con estensione *.sbn* e *.sbx* (che sono gli indici spaziali), *.prj* (che contiene informazioni sul sistema di coordinate utilizzato e sulla proiezione) e *.shp.xml* (che è il metadato geospaziale in formato XML).

Ogni *file shape* gestisce *features* di un solo tipo di geometria, cioè in un *file shape* si possono trovare o solo punti, o solo linee, o solo poligoni; i tipi primitivi di *features* puntuali disponibili sono *points* (che possono rappresentare ad esempio pozzi o città) e *multipoints* (che possono rappresentare ad esempio arcipelaghi), quelli di *features* lineari sono *lines* e *polylines* (che possono rappresentare ad esempio fiumi o strade),

[2] http://www.esri.com

quelli di area sono *polygons* (che possono rappresentare ad esempio singole isole o stati).

Ciascuna *feature*, assieme alla sua geometria e ad eventuali attributi, costituisce l'entità che si intende rappresentare. La totalità delle entità di ugual natura memorizzate nel *file shape* costituisce un *layer* sulla mappa; tramite appositi software è possibile visualizzare i vari *layers* in sovrapposizione e l'interazione tra *layers* diversi (i.e. unione, intersezione, etc.).

3. SCELTE TECNOLOGICHE

3.1 OpenJump

La scelta del software GIS da utilizzare è ricaduta su OpenJUMP, un desktop GIS open source derivato da JUMP, sviluppato da Vivid Solution e Refraction Research Inc. e basato sul linguaggio Java (JUMP è infatti acronimo di *Java Unified Mapping Platform*). Il progetto OpenJUMP è nato diversi anni or sono con l'intento di riunificare alcuni progetti, tutti derivanti dal progetto JUMP, ampliarne le funzionalità, diffonderne e documentarne l'uso.

Figura 3 – Il logo di OpenJUMP

Questa applicazione gestisce sia file vettoriali che, grazie ad alcune estensioni, file *raster* e database (PostGIS, Oracle, ArcSDE), ha una capacità di editing e disegno potente e al tempo stesso molto intuitiva: gli elementi possono essere scalati, ruotati, spostati, etc., inoltre la sezione strumenti offre una vasta gamma di tools di editing e di analisi spaziale.

La creazione delle mappe del Sud Sudan e di Yirol è stata effettuata sfruttando tali strumenti.

La caratteristica fondamentale che ha indirizzato la scelta su di esso è stata, tuttavia, la sua struttura modulare, la quale permette facilmente di estendere le funzioni di base tramite la scrittura di *plugins* (in Java) che si integrano perfettamente con il nucleo del programma. Un *plugin* implementa verosimilmente un algoritmo che va ad operare sui dati forniti in ingresso, ad esempio tramite un *file shape*, lavorando sulla parte delle

geometrie e/o su quella degli attributi. In OpenJUMP lo sviluppo e ed il test di *plugins* avvengono praticamente in parallelo, in quanto subito dopo aver scritto il codice è possibile immediatamente visualizzarne il risultato.

Il problema della scelta e visualizzazione del percorso stradale ottimo per la distribuzione di vaccini e/o reti anti zanzare presso una serie di località è stato affrontato e risolto proprio con la scrittura di un *plugin* per OpenJUMP.

3.2 SQL: PostgreSQL e PostGIS

Figura 4 - Il logo di PostgreSQL e PostGIS

La parte di inserimento/archiviazione/modifica/cancellazione online delle informazioni (di tipo geografico e non) è stata ovviamente affrontata con l'impiego di un database gestito in SQL.

SQL (acronimo di *Structured Query Language*) a dispetto del nome non è soltanto un Query Language ma include anche i linguaggi tipici per le funzioni di progettazione, gestione e amministrazione del database, come:

- Creazione e modifica di schemi di database (DDL)

- Lettura, modifica e gestione di dati memorizzati in un RDBMS (DML)
- Creazione e gestione di strumenti di controllo e accesso ai dati (DCL)

La scelta tecnologica si è orientata, ancora una volta, verso il mondo open source, con l'adozione di un DBMS come PostgreSQL, un object-relational DBMS che, com'è facile intuire dal nome, usa il linguaggio SQL per eseguire queries sui dati. Come tools grafici di gestione sono state usate le applicazioni pgAdmin (applicazione multipiattaforma scritta in C++ che consente di amministrare in modo semplificato, per mezzo di un'interfaccia grafica, database di PostgreSQL) e phpPgAdmin (applicazione PHP libera che consente di amministrare in modo semplificato, con un'interfaccia grafica basata sul web, database di PostgreSQL). L'interazione tra PostgreSQL e OpenJUMP (o, più in generale, qualunque altro software GIS) è resa possibile tramite PostGIS, estensione spaziale di PostgreSQL, un geodatabase che attua il sistema di gestione dati sui quali è basato un GIS.

3.3 Java

Figura 5 - Il logo di Java

Lo sviluppo del *plugin* per OpenJUMP è stato basato su Java, un linguaggio di programmazione orientato agli oggetti, indipendente dalla piattaforma, con una già vastissima base di utenza e che permette di sfruttare strumenti e librerie open source.

Una di queste librerie è la Java Topology Suite (JTS): sviluppata nella piattaforma Java JDK 1.4, è una libreria software open source che fornisce un modello a oggetti per le geometrie Euclidee planari e lineari, assieme ad un set di funzioni geometriche

fondamentali. E' stata concepita soprattutto per essere impiegata come nucleo in software geomatici basati su vettori, come i GIS, ma può essere utilizzata anche come libreria generica che fornisce algoritmi in geometria computazionale. Le classi geometriche supportano la gestione di *points*, *linestrings*, *polygons* e *collections*.

Per la stesura del codice, infine, è stato scelto Eclipse, un ambiente di sviluppo integrato (IDE) open source multilinguaggio e multipiattaforma, scritto in Java, che offre al programmatore comode funzioni di aiuto quali il completamento automatico e il suggerimento dei tipi di parametri dei metodi. Inoltre attua una perfetta interazione con OpenJUMP anche in modalità debug, consentendo di lanciare varie istanze del programma direttamente dal suo interno e di modificare "al volo" il codice scritto, favorendone il test in tempo reale.

3.4 PHP

Il sito web per la gestione online del database è stato creato in PHP (acronimo di *PHP: Hypertext Preprocessor*), un linguaggio di programmazione interpretato originariamente concepito per la programmazione web, cioè la realizzazione di pagine web dinamiche.

Figura 6 - Il logo di PHP

La decisione di adottare questa tecnologia è stata dettata dalla semplicità con cui PHP si integra con PostgreSQL e con cui permette la gestione delle più comuni operazioni su database. A differenza di linguaggi come HTML, infatti, con cui si possono creare pagine web di tipo statico, l'elaborazione di codice PHP sul server produce essa stessa, di volta in volta, il codice HTML da inviare al browser dell'utente che sta navigando: la pagina viene "creata" ogni volta ex novo e può essere differente a

seconda delle condizioni presenti in quel preciso momento (ad esempio, le pagine che realizzano la ricerca vengono "generate" solo dopo che l'utente ha inserito i parametri desiderati).

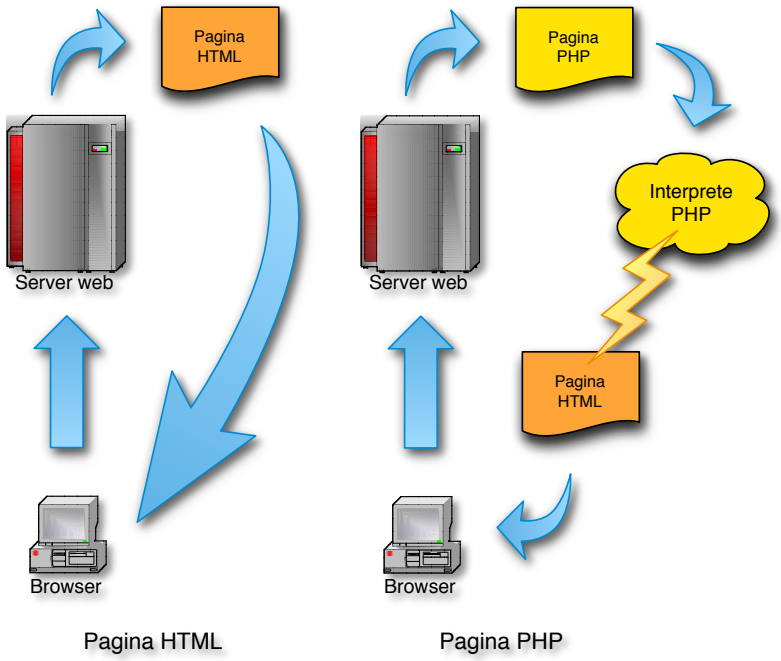

Figura 7 - Produzione di una pagina PHP

Come si può vedere dalla *figura 7*, nel caso di pagina statica scritta in HTML, il server, dopo aver ricevuto la richiesta, invia la pagina al browser, il quale seguendo le istruzioni HTML la ricostruisce e la visualizza sullo schermo del computer; se invece la pagina è scritta in PHP, il server, dopo aver ricevuto la richiesta, invia la pagina al preprocessore PHP, il quale interpreta le istruzioni PHP per creare una nuova pagina HTML da inviare al browser che ha effettuato la richiesta. Il browser riceve quindi, in ogni caso, una pagina HTML.

3.5 Software FOSS

Con il termine FOSS, acronimo di *Free and Open Source Software*, si identificano contemporaneamente il software libero e quello open source. Di fatto, la differenza che

separa le due categorie è minima: entrambe comprendono al loro interno software libero, il cui codice sorgente è pubblico. Differiscono invece dal punto di vista etico e morale: l'open source è una metodologia di sviluppo, il software libero è più inquadrato come movimento di carattere sociale.

Vale la pena sottolineare, ancora una volta, come la scelta della tecnologia software adottata si sia sempre rivolta verso il mondo FOSS: a cominciare dal GIS OpenJUMP, proseguendo con i linguaggi di programmazione Java e PHP, ed il relativo editor usato per scrivere il codice Eclipse, concludendo con il DBMS PostgreSQL. La loro adozione è stata motivata dal fatto che questi strumenti, seppure offerti con licenza per software libero, e quindi reperibili gratuitamente nel web, sarebbero riusciti a soddisfare pienamente le esigenze progettuali; inoltre il supporto online è ampio, sono prodotti conosciuti e largamente diffusi, si integrano l'un l'altro con facilità e hanno di fatto annullato i costi per il software impiegato.

4. REALIZZAZIONE MAPPE TEMATICHE SUD SUDAN E YIROL

Il primo lavoro compiuto è stato l'allestimento di mappe tematiche riguardanti il Sud Sudan ed in particolare la regione di Yirol, generate e gestite con il software open source OpenJUMP, esportate e consegnate poi in file .pdf. Le mappe sono state richieste dai responsabili del Cuamm in vista di una spedizione in tali zone, al fine di disporre di informazioni immediate e facilmente consultabili riguardo la situazione demografica generale del posto e quella dei centri sanitari.

Per quanto concerne il Sud Sudan, esse trattano la descrizione di diverse caratteristiche demografiche, come la suddivisione del territorio in macro-regioni (dette *counties*), la distribuzione della popolazione all'interno di queste macro-regioni, la sua densità e la sua divisione per sesso ed età, con particolare attenzione alle statistiche della popolazione femminile; per Yirol, invece, illustrano la dislocazione delle strutture sanitarie rispetto ai villaggi, evidenziando quelli che distano più di 10 chilometri da un *health center* e che quindi risultano difficilmente raggiungibili. Infine è stato generato un ulteriore set di mappe che considera la condizione dell'*health center*, in cui si differenziano quelli effettivamente funzionanti da quelli parzialmente funzionanti o in disuso, in modo da capire quale potrebbe essere l'effetto di una riattivazione della struttura sanitaria dismessa in termini di copertura di potenziali utenti.

Tecnicamente le mappe descrittive (quelle del Sud Sudan) sono state ottenute organizzando con OpenJUMP la visualizzazione dei *file shape*, forniti dal Cuamm e contenenti le informazioni di interesse, in modo da far risaltare di volta in volta i caratteri significativi tramite etichette o colorazioni tematiche differenti (vedi *figure 8*, *9* e *10*). Per le mappe di Yirol invece si è sfruttata la combinazione delle funzioni "*buffer* circolare" e "intersezione geometrica" di *layers*, entrambe offerte da OpenJUMP: si è individuata una zona circolare (*buffer*) del raggio di 10 chilometri attorno ad ogni *health*

center e, dopo aver decretato quali fossero i villaggi all'interno di questa zona con l'intersezione tra il nuovo *layer* del *buffer* e quello dei villaggi, si è creato un *layer* popolato dai soli villaggi che non fossero all'interno di queste zone di vicinanza, visualizzando il risultato complementare dell'intersezione (vedi *figure 11* e *12*). Infine, l'ultimo set di mappe è stato ottenuto in maniera simile a quello appena descritto con la differenza che in questo si distinguono le strutture sanitarie definite "funzionanti" da quelle "non funzionanti", sulla base di un attributo fornito nelle tabelle dei file shape degli health centers. Sono di conseguenza stati creati due tipi di *buffer*: uno attorno alle strutture funzionanti e un altro attorno a quelle non funzionanti (vedi *figura 13*).

Nella visualizzazione finale risulta notevole come alcuni villaggi, che prima erano "coperti" e quindi non apparivano nella visualizzazione, ora invece si trovano nel raggio di un *health center* che non è in grado di assicurare servizio medico. Interpretando queste ultime mappe si possono, ad esempio, avanzare dei ragionamenti su quali *health centers* sarebbe più opportuno riattivare, in modo da servire il maggior numero di villaggi possibile.

Un approccio di questo genere, sviluppato successivamente attraverso un'analisi approfondita, è stato descritto, ad esempio, nell'articolo pubblicato sull'International Journal of Gynecology and Obstetrics "*Using a GIS to model interventions to strengthen the emergency referral system for maternal and newborn health in Ethiopia*" (Bailey & al, 2011). La conclusione significativa a cui si giunge nello studio citato è che: "Il rilevamento e la modellizzazione tramite GIS permettono analisi spazio-temporali fondamentali per studiare l'accesso della popolazione ai servizi sanitari e il sistema di smistamento dei pazienti. La disposizione di veicoli e sistemi di comunicazione e il miglioramento dei centri sanitari a ospedali di raccolta di primo livello sono strategie a breve e medio termine che possono aumentare rapidamente l'accesso ai servizi di soccorso".

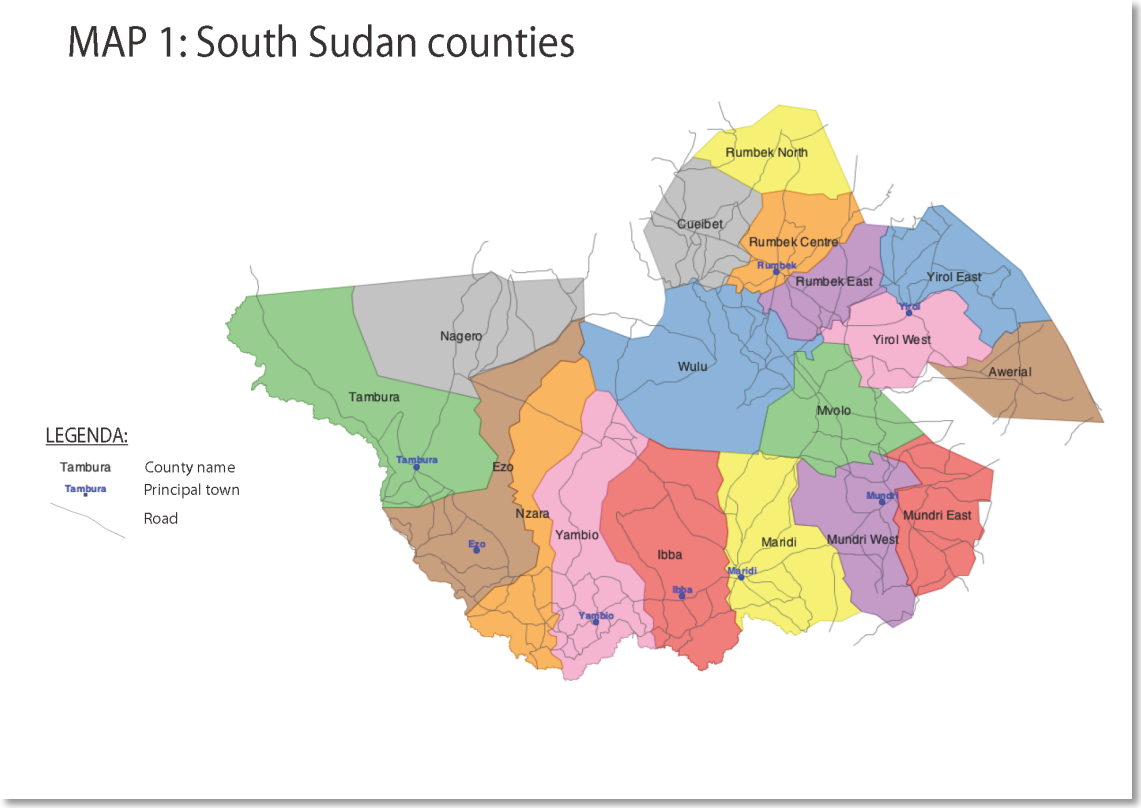

Figura 8 - Mappa delle contee del Sud Sudan

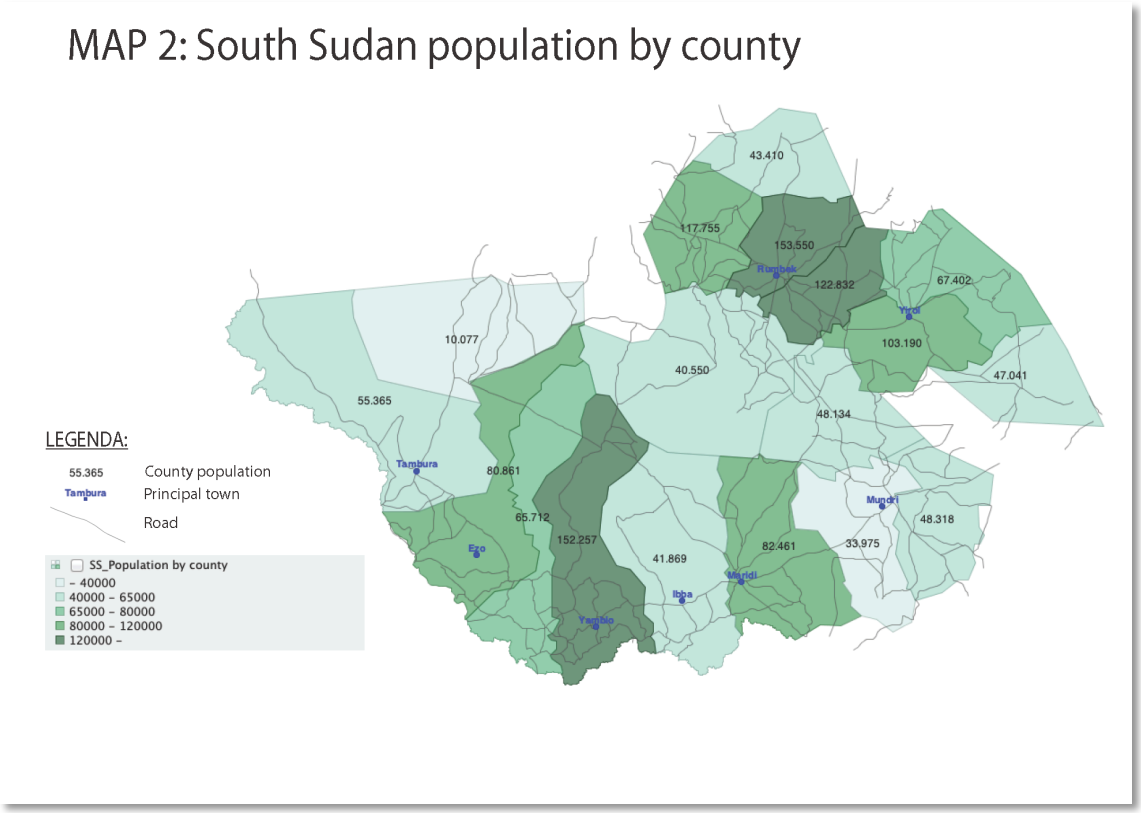

Figura 9 - Mappa della popolazione per contea

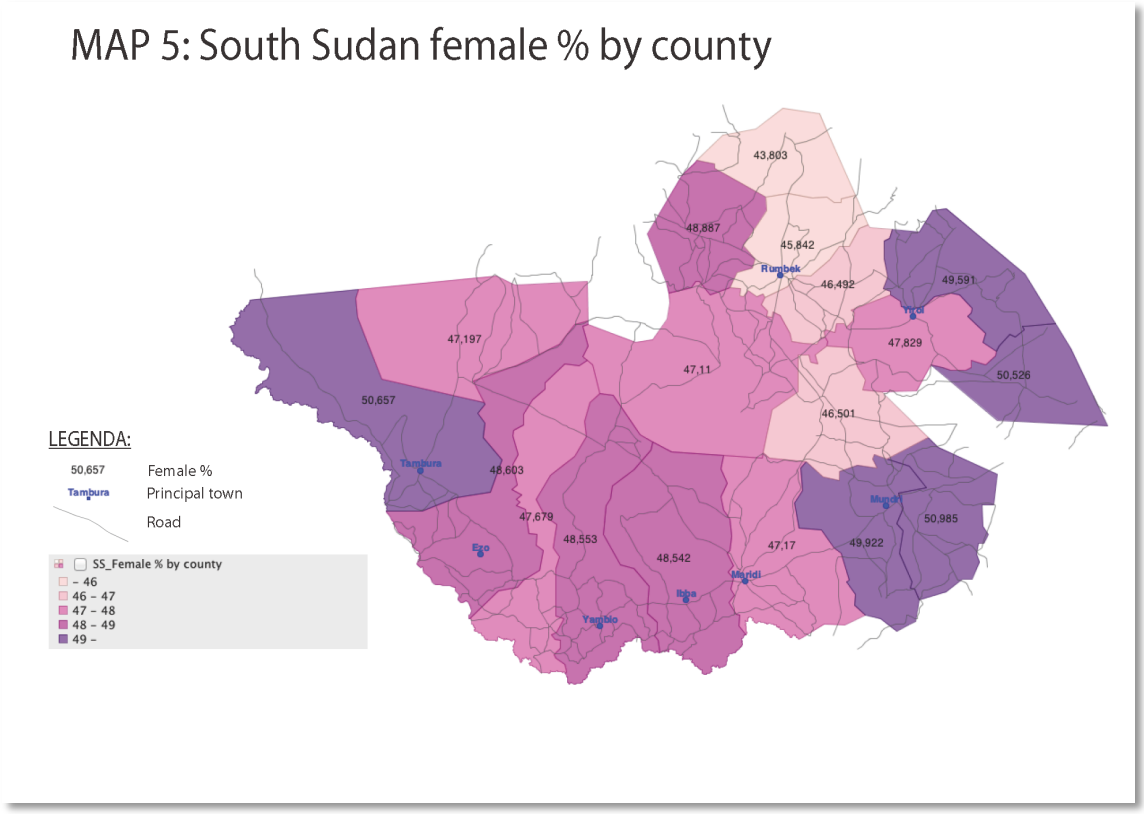

Figura 10 - Mappa della % della popolazione femminile per contea

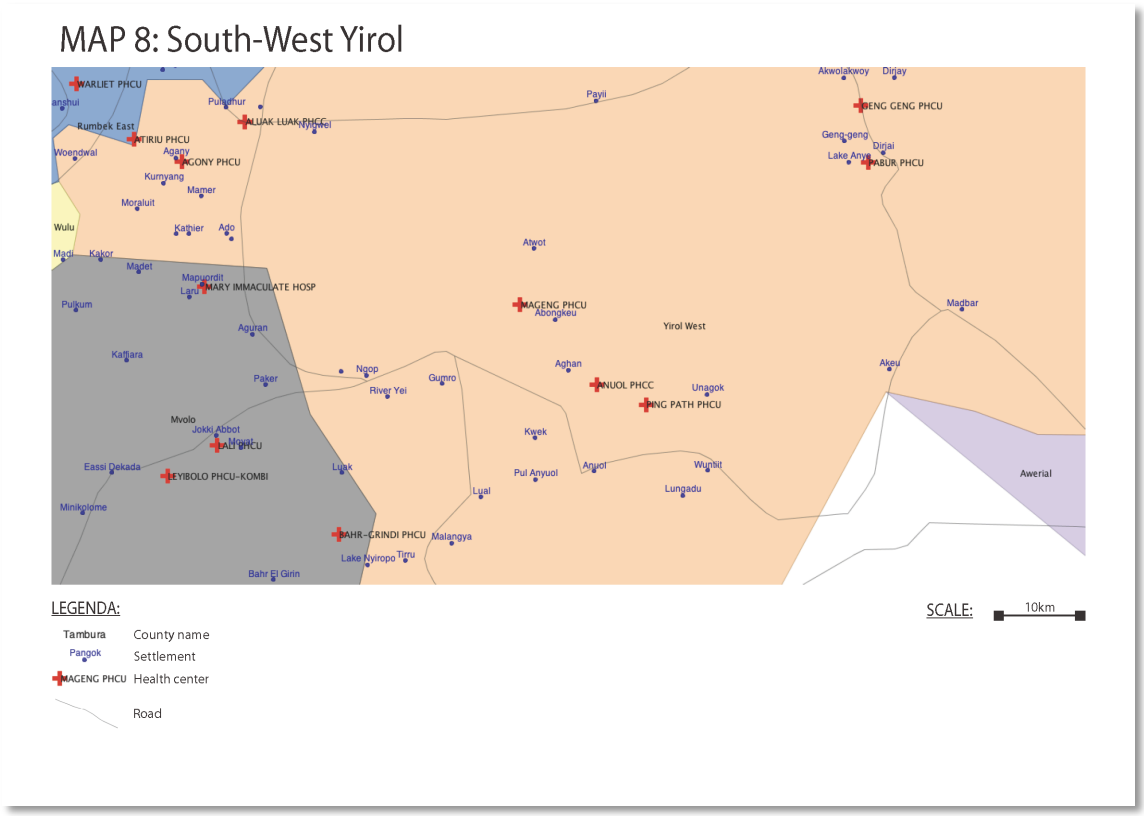

Figura 11 - Mappa degli HC nella zona S-O di Yirol

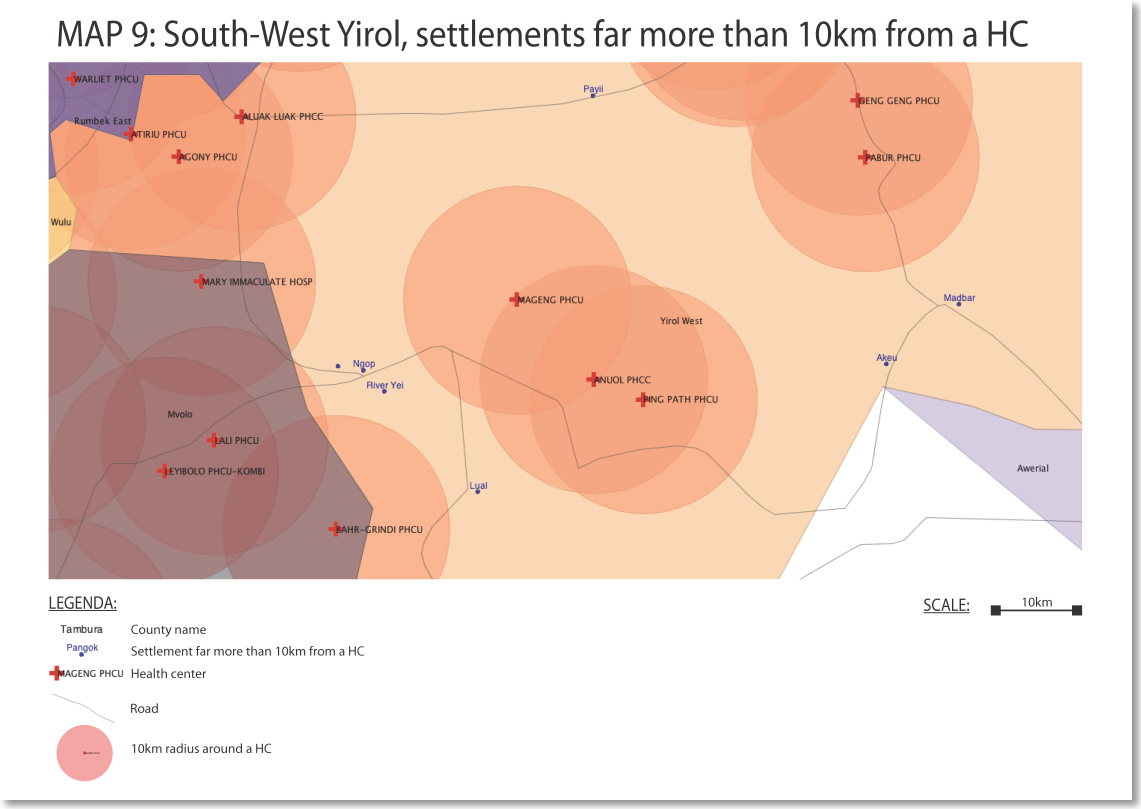

Figura 12 - Mappa dei villaggi lontani più di 10km da un HC, zona S-O di Yirol

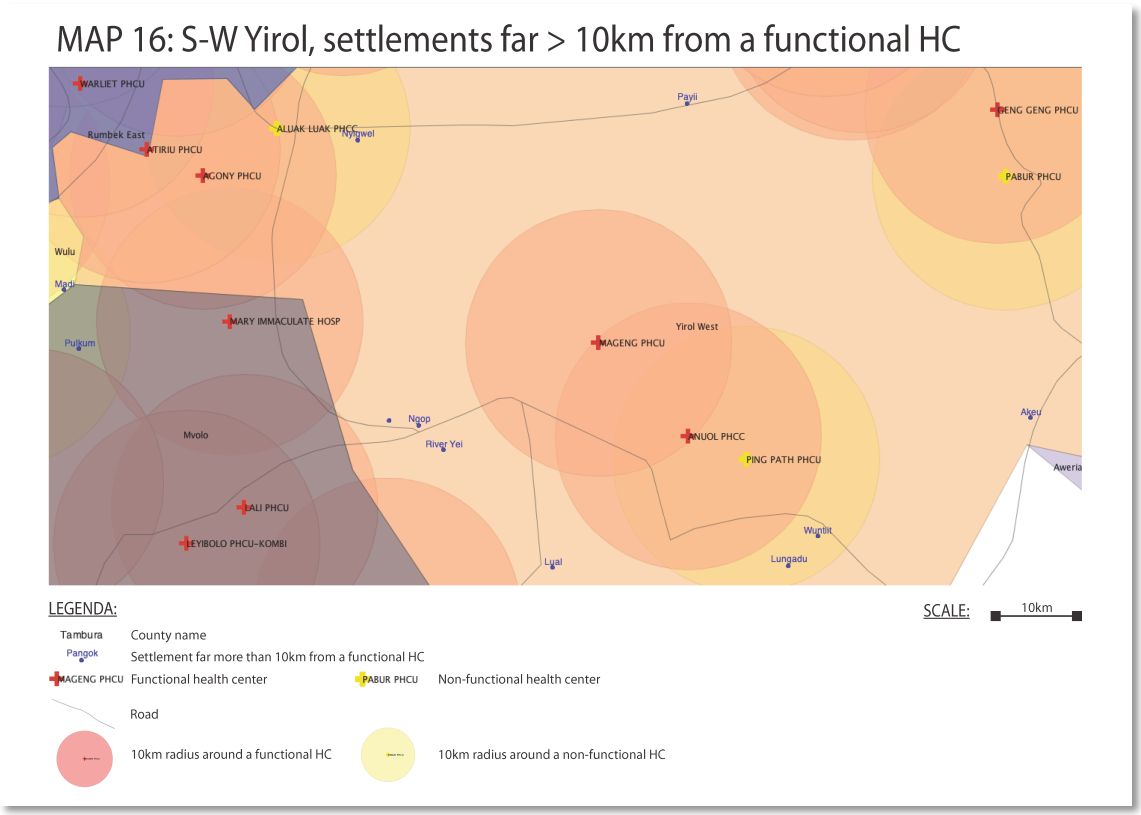

Figura 13 - Mappa dei villaggi lontani più di 10km da un HC funzionante, zona S-O di Yirol

5. REALIZZAZIONE DATABASE E WEBSITE

Un'ulteriore necessità del Cuamm consisteva nel gestire l'anagrafica delle nascite e delle vaccinazioni nelle zone in cui presta operato. A questo scopo è stato creato un database, scegliendo di renderlo accessibile online in modo da potervisi connettere, tramite il web, da una qualsiasi postazione remota. Per prima cosa si è scelto un servizio di *web hosting* che risultasse compatibile con gli strumenti utilizzati, cioè PHP e PostgreSQL: alwaysdata (*www.alwaysdata.com*). Il servizio è gratuito nella sua offerta base, che comprende 10MB di spazio su disco, 64MB di memoria RAM e 1GB di traffico mensile. A queste condizioni è chiaro come sia possibile impostare solo una fase iniziale di test e non un reale utilizzo a regime; ciononostante la struttura del database resta valida in previsione di un upgrade ad offerte che mettano a disposizione un maggior spazio per l'archiviazione oppure di una migrazione verso una piattaforma differente, ferma restando la capacità di quest'ultima di interfacciarsi con la tecnologia definita.

5.1 Progettazione concettuale db: schema E-R

In fase di progettazione concettuale si è cercato di pensare ad una struttura che risultasse semplice ma al tempo stesso completa.

Le funzioni del database dovevano essere quelle sopra citate: registrare l'anagrafica delle neomadri (ID madre, nome, cognome, data di nascita, luogo di nascita, numero di figli, data di morte, eventuali altri dati…), dei neonati (ID neonato, nome, cognome, data di nascita, luogo di nascita, sesso ed esito parto) e delle visite effettuate da questi, registrandone la data e l'esito (come attributo semplice o, in caso, come una scansione del referto medico), così come degli operatori sanitari (ID operatore, nome, cognome, data di nascita, luogo di nascita, qualifica, mansioni) di ruolo negli *health centers*. Il database doveva inoltre tener traccia delle vaccinazioni effettuate sia presso i villaggi

che presso gli *health centers*, registrando il tipo di vaccinazione, la data di somministrazione e il totale delle persone vaccinate oltre che, ovviamente, dove e da chi è stata fatta. La parte geografica del database è rappresentata dalle informazioni ricavate direttamente dai *file shape* forniti inizialmente. Come accennato prima, questo tipo di formato non contiene solamente dati geografici, ma anche tabelle di attributi. In questo modo si sono create due tabelle, Villaggio e HC, contenenti (oltre ai dati geografici) indicazioni dettagliate sui villaggi (PCode, nome, tipo, latitudine, longitudine, popolazione, State, County, Payam di appartenenza e la data in cui sono state distribuite le reti antizanzare) e sugli *health centers* (nome, ID interno, tipo, latitudine, longitudine, se è funzionante o meno, State, County e Payam di appartenenza).

Dall'analisi condotta è stato prodotto lo schema E-R.

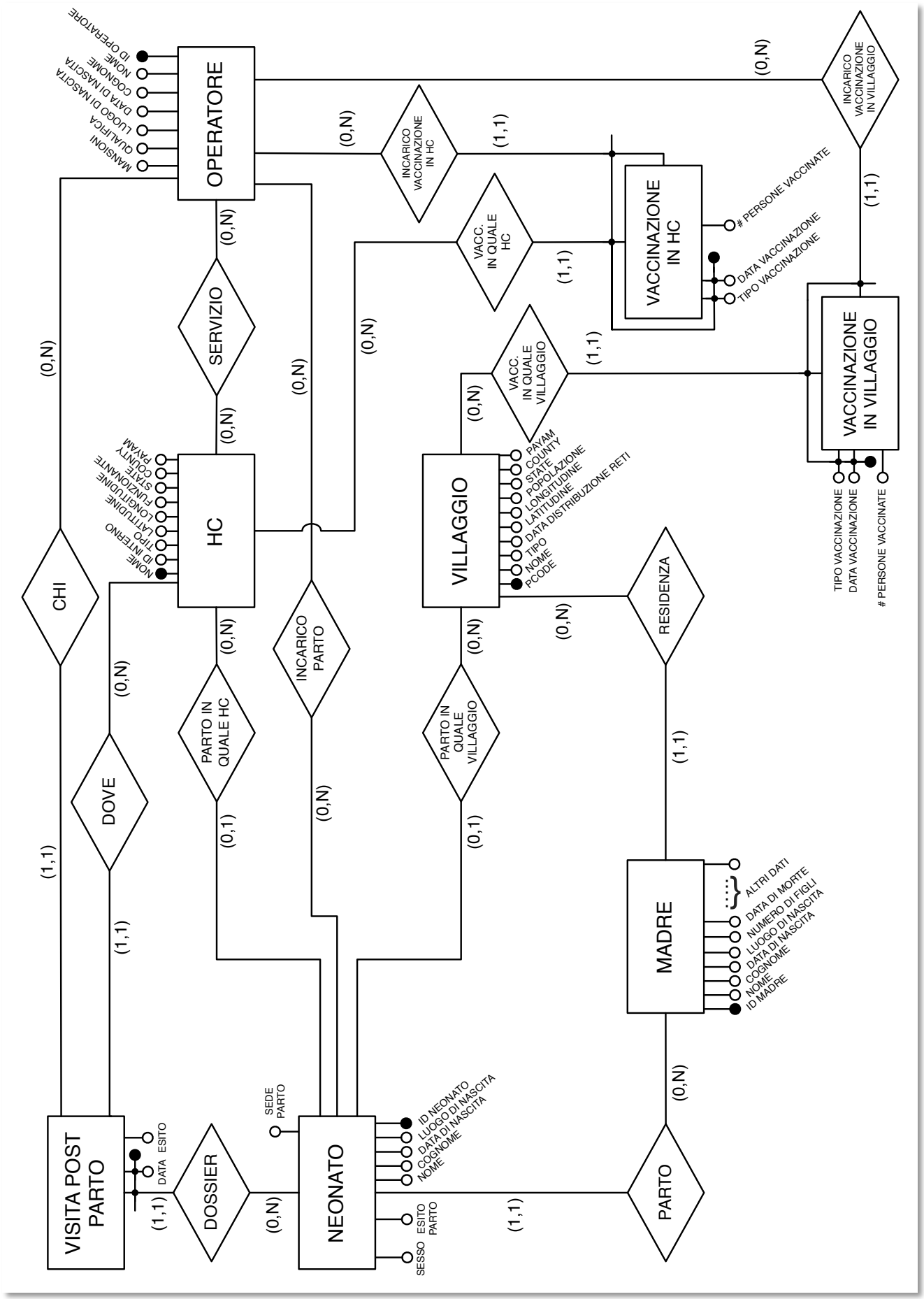

Figura 14 - Schema E-R

5.2 Progettazione logica db: schema logico

Lo schema E-R ottenuto è servito per compilare il relativo schema logico.

5.3 Realizzazione

La realizzazione del database è stata realizzata direttamente online con l'ausilio di phpPgAdmin, un'applicazione PHP open source che consente di amministrare in modo semplificato database di PostgreSQL tramite un qualsiasi browser, messa a disposizione da alwaysdata.

Per accedere al database e alle sue funzioni (inserimento, modifica, cancellazione, ricerca…) è stato creato un semplice sito web in PHP che, una volta autenticati per

mezzo di username e password, permette di visualizzare le tabelle e di operare su di esse. Il sito web è raggiungibile all'indirizzo http://lucadiielsi.alwaysdata.net.

Figura 15 - Homepage del sito web

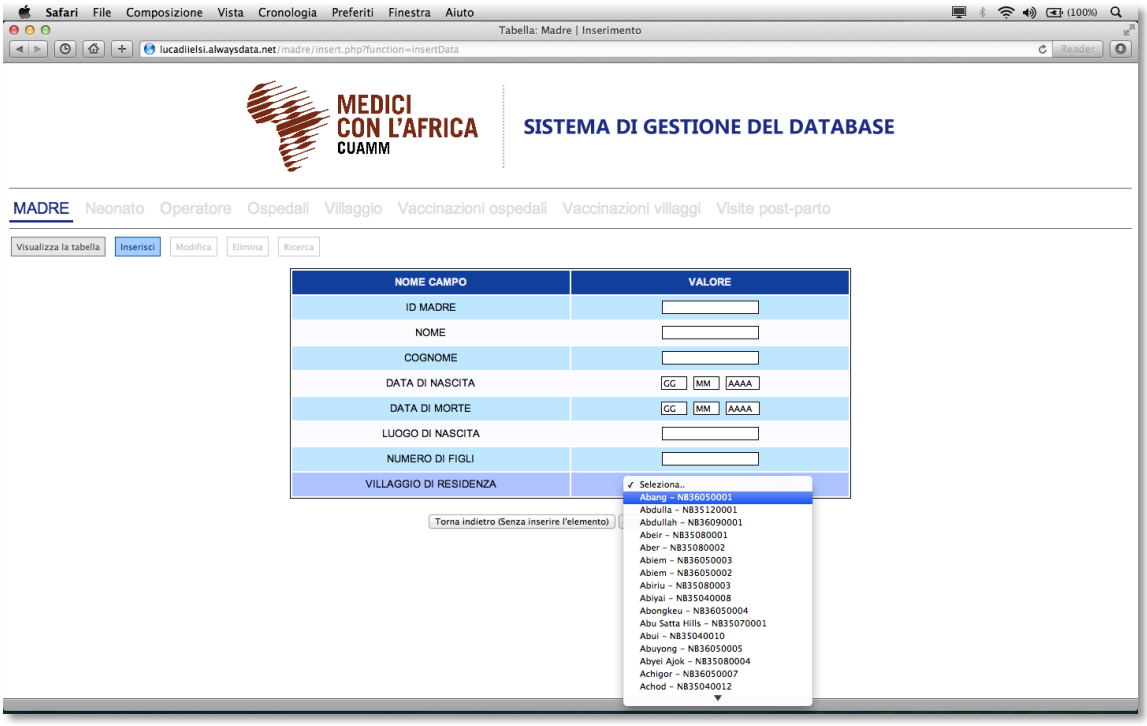

Figura 16 - Tabella Madre, pagina di inserimento dati

6. REALIZZAZIONE APPLICAZIONE PER IL PERCORSO OTTIMO

6.1 Il *Travelling Salesman Problem*

Il problema del commesso viaggiatore (da cui l'acronimo inglese TSP: *Travelling Salesman Problem*) consiste nel trovare il percorso minimo che passi una e una sola volta per tutte le città di un elenco dato e ritorni al punto di partenza.

In teoria dei grafi, la formulazione risulta: dato un grafo completo e pesato, trovare il cammino di costo minore visitando tutti i nodi una sola volta e tornando al nodo di partenza. La rete di città può essere rappresentata come un grafo in cui le città sono i nodi, le strade sono gli archi e, come pesi sugli archi, si può attribuire la distanza euclidea, la distanza lungo una strada, il tempo di percorrenza (facendo attenzione che non sempre il tempo di percorrenza da A a B corrisponde a quello da B ad A, si pensi ad esempio ad una strada che presenti una certa pendenza: percorrerla in un salita richiederà mediamente più tempo che percorrerla in discesa), ecc..

Nel caso trattato, si è assunto di considerare il tragitto minimo come il più corto in termini di lunghezza. Per semplificare, si è assunto anche che non esistano "sensi unici", ossia che se da A è possibile raggiungere B, allora anche da B è possibile raggiungere A, percorrendo la medesima distanza. Quest'ultima assunzione permette di definire il problema come simmetrico.

Nonostante queste semplificazioni, e nonostante il TSP sia uno dei problemi di matematica computazionale più studiati, ancora oggi non se ne conosce un metodo risolutivo in grado di fornire una soluzione ottima: infatti da oltre cinquant'anni gli studi effettuati hanno portato allo sviluppo di vari metodi di soluzione in diversi campi dell'ottimizzazione matematica. Per avere un'idea della complessità di calcolo in caso si volesse tentare un approccio risolutivo di tipo *brute force*, bisognerebbe considerare che

il numero totale dei differenti percorsi possibili attraverso le *n* città si calcola così: data una città di partenza, sono disponibili *n-1* scelte per la seconda città, *n-2* per la terza, ecc.. Il totale delle possibili scelte tra cui cercare il percorso migliore è dunque *(n-1)!*, e data la simmetria del problema, questo numero va diviso a metà. Perciò, date *n* città, i percorsi che le collegano sono $\frac{(n-1)!}{2}$, cioè *O(n!)*, l'ordine di grandezza degli algoritmi *brute force* che cercano la soluzione di problemi NP-completi.

Appare evidente come non sia applicabile tale metodo di risoluzione, e sia necessario applicare tecniche euristiche che non forniscono la soluzione ottima, ma una soluzione approssimativamente molto vicina ad essa.

6.2 *Plugin* per trovare il percorso ottimo

Il *plugin* realizzato si integra con il software GIS open source OpenJUMP ed è stato scritto in Java.

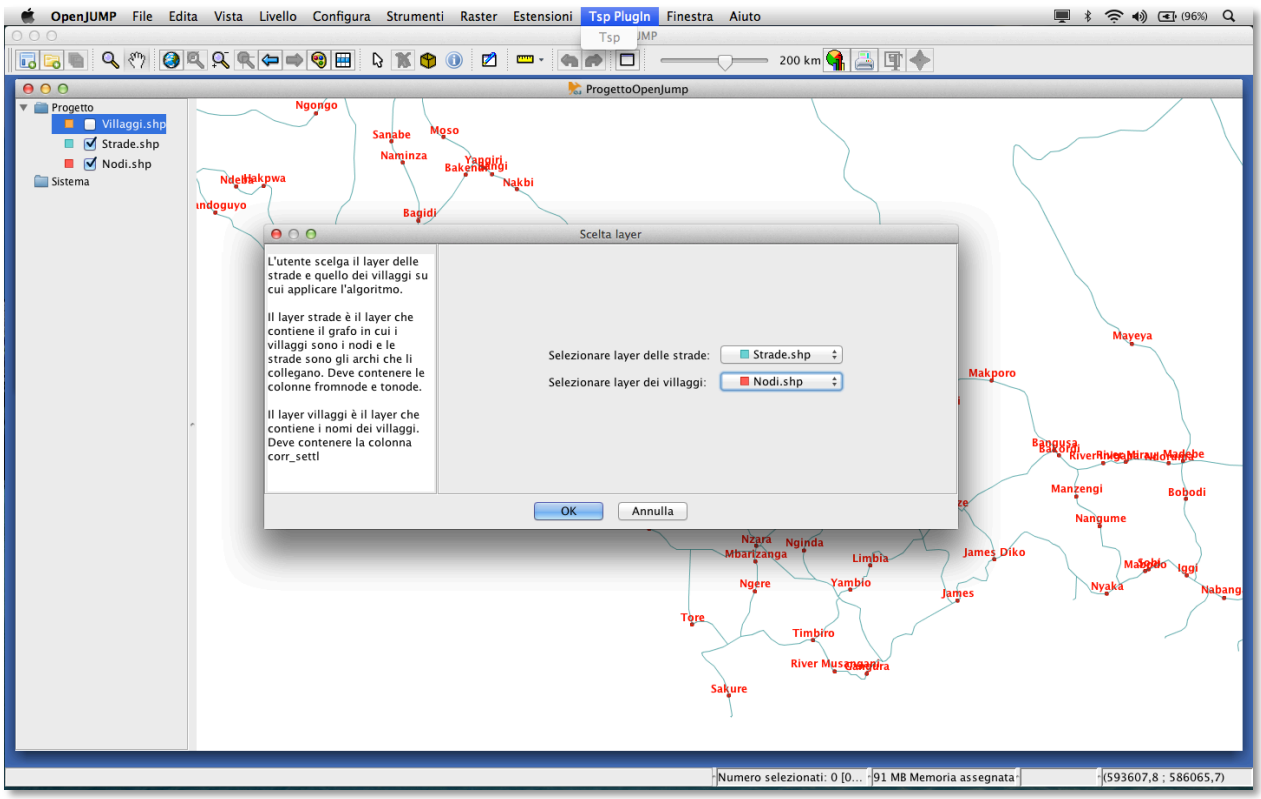

Figura 17 - Schermata di scelta dei *layers*

All'avvio è dotato di un'interfaccia grafica piuttosto semplice e intuitiva con cui l'utente è chiamato a scegliere in prima battuta un *layer* "strade" e un *layer* "villaggi" (vedi *figura 17*) e poi, da una lista ricavata dal *layer* "villaggi" appena selezionato, i villaggi che desidera visitare (vedi *figura 18*). Questa lista è ordinata alfabeticamente e del resto non è interessante l'ordine con cui l'utente sceglie i villaggi da visitare, giacché il risultato è un circuito chiuso, dove non ha importanza alcuna da quale nodo si parta.

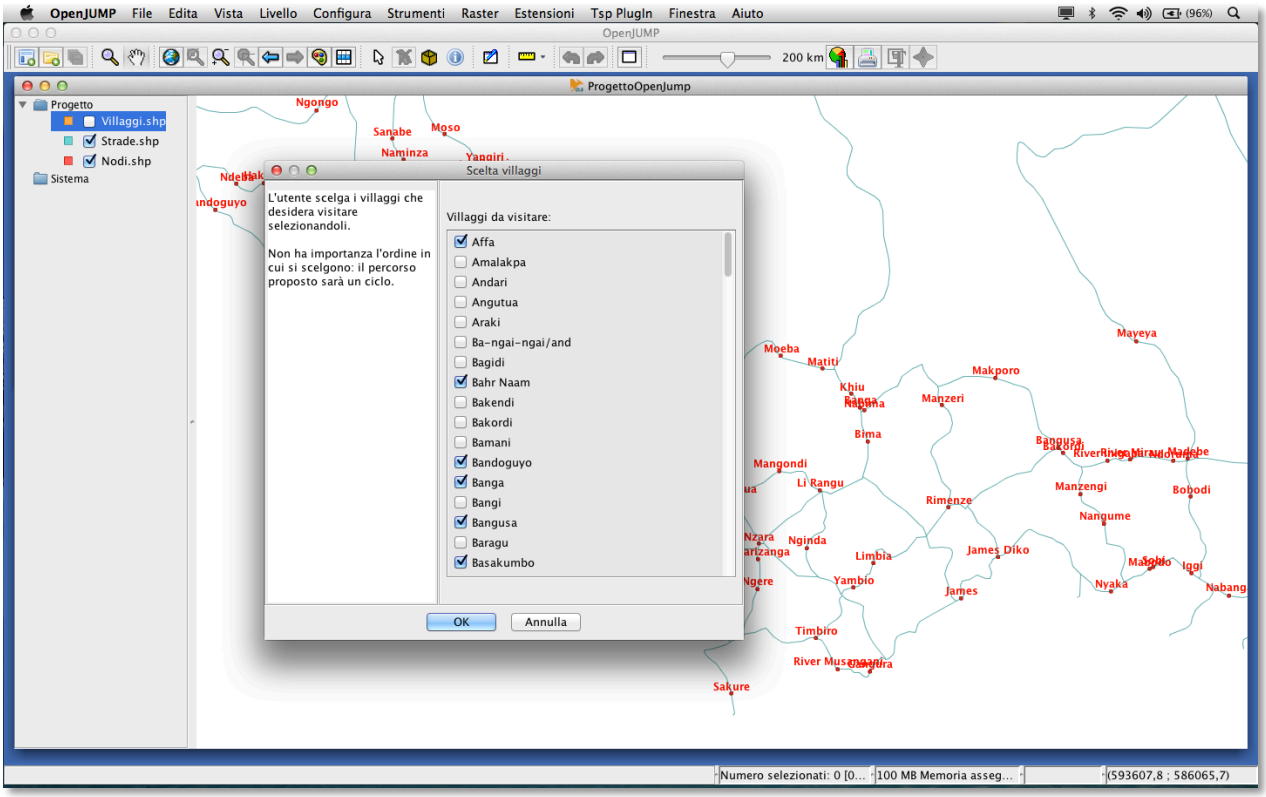

Figura 18 – Schermata di scelta dei villaggi

Durante la fase di scelta, uno scenario piuttosto frequente in situazioni di questo genere potrebbe essere quello in cui l'utente desidera visitare un insieme di villaggi dislocati all'interno di una certa area geografica. Per fare ciò, è sufficiente creare in OpenJUMP un nuovo *layer* di nodi (sottoinsieme del *layer* di nodi originale) costruito selezionando i villaggi nella zona di interesse con lo strumento di selezione rettangolare del mouse e applicando la funzione "Replica elementi…" attivando l'opzione "Replica in un nuovo

livello". A questo punto si sceglie questo nuovo *layer* come *layer* "villaggi", selezionando tutti i villaggi visualizzati. E' possibile adoperare il medesimo procedimento qualora si desiderasse effettuare la selezione dei villaggi direttamente dalla mappa, anche senza rimanere all'interno di una zona rettangolare ma individuando villaggi sparsi.

L'algoritmo utilizzato si serve della soluzione (il cui codice sorgente è reperibile nel sito web all'indirizzo http://www.rene-grothmann.de/java/TSP/index.html) che cerca il percorso più breve tramite la tecnica di minimizzazione locale chiamata discesa del gradiente. Questo inizia trovando un cammino casuale scelto tra tutti i possibili e prosegue cercandone uno più corto tra quelli nelle vicinanze, tramite miglioramenti successivi, definiti scambiando gli elementi del cammino. Infine, quando non sono più possibili ulteriori iterazioni, arriva a trovare un ottimo locale.

6.3 Input dei dati

L'algoritmo scelto per risolvere il TSP richiede in ingresso il grafo pesato rappresentante la rete di villaggi (nodi) e strade (archi), e la corrispondente matrice di adiacenza o, per meglio dire, matrice dei pesi. In tale matrice, quadrata e di ordine $n \times n$ (con n numero dei villaggi da visitare), ogni elemento (i, j) è il peso associato all'arco (i, j), congiungente i nodi i e j. In questo modo è possibile individuare sia la disposizione degli archi tra i nodi, sia i valori del loro peso; qualora l'arco (i, j) non fosse presente (e quindi i villaggi i e j non fossero uniti direttamente da una strada) si attribuisce alla coppia (i, j) un peso infinito ∞.

Nel caso trattato, ossia quello in cui la realtà è rappresentabile tramite un grafo non orientato, la matrice dei pesi risulta simmetrica, poiché $(i, j) = (j, i)$, con medesimo peso. Il grafo e la matrice dei pesi sono stati ricavati dai *file shape* forniti. Per prima cosa è stato creato il grafo: si è modificato il *file shape* delle strade, dove inizialmente i nodi

rappresentavano solamente gli incroci, “tagliando” i segmenti in prossimità dei villaggi e inserendovi un nuovo nodo che si è battezzato con il nome del villaggio (vedi *figura 19*). Sono stati mappati in questo modo 128 villaggi campione, corrispondenti ad altrettanti nodi del grafo. Questo numero costituisce solo una piccola parte della totalità dei villaggi presenti nel *file shape* originale, tuttavia di certo sufficiente per testare la validità del *plugin*.

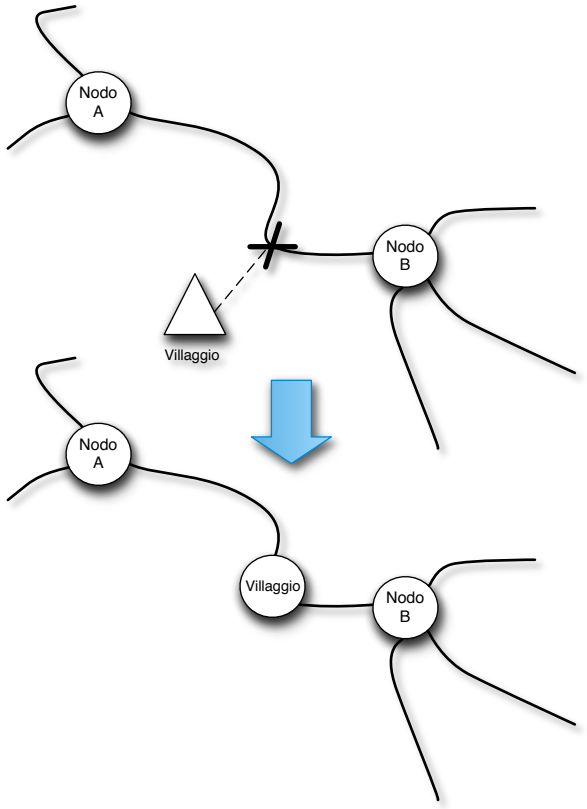

Figura 19 - Meccanismo di creazione del grafo

Successivamente si è calcolata la matrice dei pesi, limitatamente ai villaggi da visitare scelti dall’utente, applicando l’algoritmo di Dijkstra per calcolare il percorso minimo, e la relativa lunghezza, per ogni coppia di villaggi. Questo algoritmo, che prende il nome dall’informatico olandese Edseger Dijkstra (1930-2002), il quale lo formulò nel 1956, permette appunto di trovare i cammini minimi in un grafo ciclico con pesi non negativi sugli archi; in particolare può essere utilizzato parzialmente per trovare il cammino minimo che unisce due nodi del grafo ovvero totalmente, per trovare quelli che

uniscono un nodo d'origine a tutti gli altri nodi o, iterando il procedimento, per trovare tutti i cammini minimi da ogni nodo a ogni altro nodo, come nel caso in questione.

6.4 Output dei dati

Il risultato dell'esecuzione del *plugin* è rappresentato dalla visualizzazione in OpenJUMP di due nuovi *layers*: il primo è costituito dai nodi scelti dall'utente, su cui si è fatto agire l'algoritmo per risolvere il TSP, ed ha solo titolo informativo, mentre il secondo – e più significativo – ricalca il *layer* strade, evidenziando in particolare il percorso ottimo suggerito dall'applicazione dell'algoritmo. Oltre a visualizzarlo, l'algoritmo salva tale percorso ottimo sotto forma di stringa di testo in cui elenca l'ordine di visita calcolato, nodo per nodo. Nell'implementazione scelta, questa informazione viene fornita all'utente tramite la finestra di output di OpenJUMP, contemporaneamente alla visualizzazione del percorso sulla mappa (vedi *figura 20*).

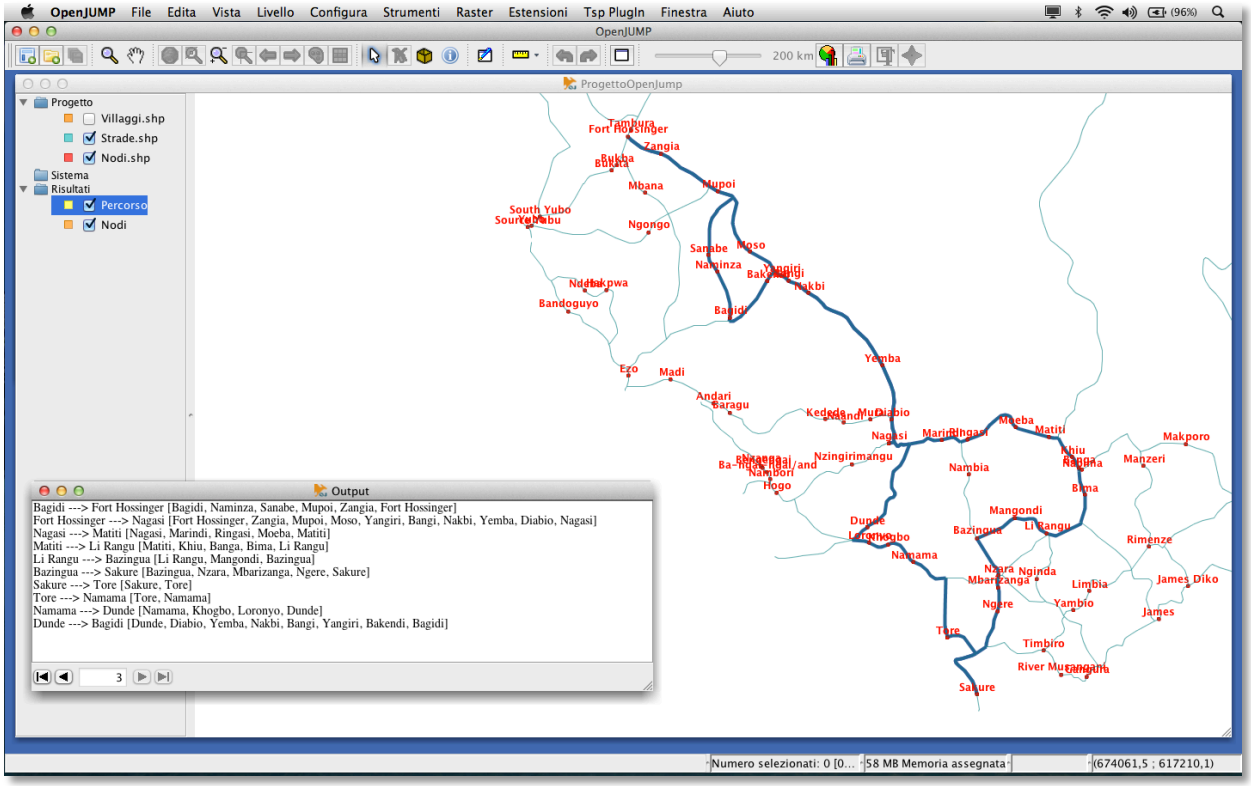

Figura 20 - L'output dell'applicazione

7. POSSIBILI SVILUPPI E VALUTAZIONI

Per quanto riguarda il primo lavoro, ossia la produzione delle mappe, non sono individuabili sviluppi rilevanti, poiché queste sono state create per soddisfare un'esigenza occasionale, di utilità per lo più pragmatica. Oltre a quelle proposte, le mappe tematiche che si possono ricavare dai dati a disposizione sono molteplici, e di volta in volta è possibile porre l'accento su aspetti specifici, per mettere in risalto particolari d'interesse.

Nel merito del database, invece, gli sviluppi possibili sono essenzialmente due:

- Come già accennato in precedenza, la piattaforma su cui è stato avviato il database è una piattaforma di prova. Si renderà necessario quindi effettuare quanto prima una migrazione verso un servizio di *web hosting* che metta a disposizione una maggior quantità di spazio su disco. La condizione necessaria è che questo nuovo servizio supporti la tecnologia adottata, cioè l'utilizzo di PostgreSQL come RDBMS e PHP come linguaggio per il sito web.

- Una funzionalità interessante di OpenJUMP consiste nella capacità di interrogare, tramite queries SQL, database spaziali. Potrebbe essere utile, dopo aver approfondito l'argomento in collaborazione con i responsabili del Cuamm, e dopo aver trovato un'intesa riguardo le informazioni a cui questi ultimi potrebbero essere interessati, sviluppare un set prestabilito di queries che vadano ad operare direttamente sul database online e forniscano i risultati tramite una visualizzazione in OpenJUMP (ad esempio: visualizzare in OpenJUMP solo gli ospedali in cui si siano somministrate più di 300 vaccinazioni nell'ultimo mese; oppure: ricavare un *layer* in OpenJUMP in cui compaiano soltanto i villaggi visitati prima di una certa data, e che quindi probabilmente richiedono una

nuova visita nel futuro prossimo). Questo lavoro diventerebbe oltremodo conveniente qualora fosse possibile individuare delle interrogazioni il cui impiego fosse in effetti molto frequente. Altrimenti si potrebbe pensare di sviluppare un'applicazione che permetta la costruzione dinamica di queries, tramite una finestra con cui l'utente possa interagire. In ogni caso rimane possibile effettuare interrogazioni estemporanee direttamente da OpenJUMP, tramite la funzione "Esegui interrogazione su datastore" (vedi *figura 21*).

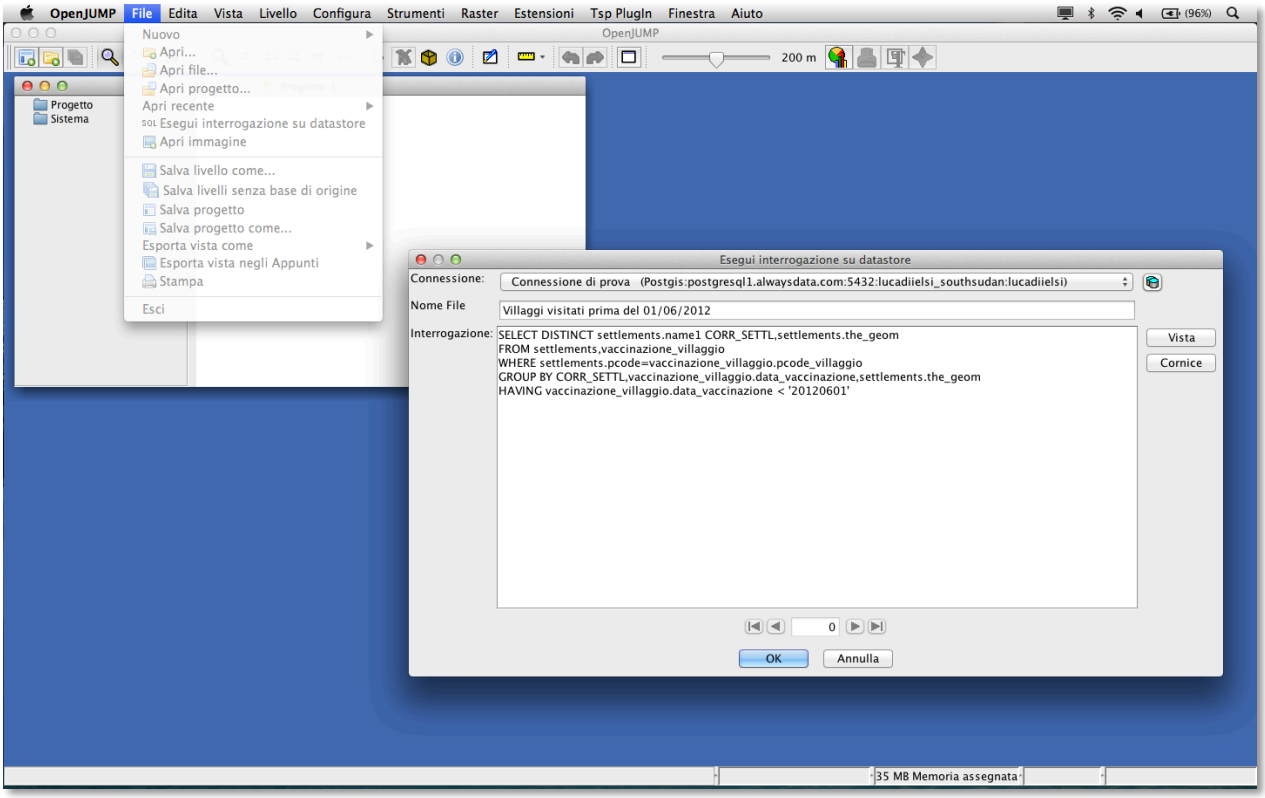

Figura 21 - Query su datastore

A proposito del *plugin* per la ricerca del percorso ottimo, il punto cruciale che andrà affrontato in futuro sarà trovare un metodo per l'automatizzazione della creazione dei *layers* "strade" e "villaggi". Una possibile soluzione potrebbe consistere nel completare a mano il grafo delle strade aggiungendo dei segmenti di connessione tra i villaggi e le principali strade già esistenti. Questi connettori rappresentano le vie di accesso, magari secondarie, che anche nella realtà collegano le strade principali ai villaggi. Non si

tratterebbe quindi di un'astrazione o di un artificio, bensì di un miglioramento qualitativo nella rappresentazione della rete stradale. Fatto questo, si ricaverebbe un grafo connesso in cui sarebbe sufficiente evidenziare i nodi che rappresentano i villaggi, filtrando e scartando quelli che rappresentano i semplici incroci.

APPENDICE A. MANUALE TspPlugIn

Introduzione

Il TspPlugIn si impiega per ottenere, dato un grafo completo e pesato, il cammino di peso minimo che visiti tutti i nodi una sola volta e che torni al nodo di partenza.

Installazione

Per poter utilizzare il *plugin* è sufficiente copiare il file TspPlugIn.jar nella cartella lib/ext, dove è stato installato OpenJUMP. Al successivo avvio del programma, verrà caricato automaticamente e comparirà tra le scelte disponibili nella barra dei menu situata nella parte superiore dello schermo, sotto il nome "TspPlugIn - TSP" .

Input dei dati

Per poter funzionare correttamente, il TspPlugIn richiede in ingresso due *layers*, che l'utente è chiamato a selezionare.

Il primo è il *layer* delle strade: tale *layer* deve rappresentare il grafo della rete stradale. Ogni strada che congiunga due nodi (siano essi villaggi o incroci) deve avere come attributi "fromnode" e "tonode", ossia gli estremi che congiunge. La lunghezza è calcolata dall'algoritmo in fase di esecuzione, e perciò non è un attributo necessario.

Il secondo è il *layer* dei villaggi (o dei nodi notevoli): può essere ricavato direttamente dal *layer* delle strade, selezionando solo i vertici che rappresentino in effetti un villaggio. Dall'attributo "corr_settl", che necessariamente deve essere presente in questo *layer*, l'applicazione ricava i nomi dei villaggi che successivamente propone all'utente per la visita.

Output dei dati

L'output dell'applicazione è rappresentato dalla visualizzazione sulla mappa del percorso ottimo calcolato, tramite la creazione di un nuovo *layer* di strade nominato "Percorso". Oltre a questo *layer* ne viene visualizzato un secondo, "Nodi", dove vengono visualizzati ed elencati i nodi visitati. Infine l'utente può visualizzare testualmente i percorsi intermedi da tappa a tappa, tramite la finestra di output di OpenJUMP.

Gestione degli errori

L'applicazione gestisce le situazioni d'errore in modo piuttosto semplice, avvisando l'utente con un messaggio che riporta il tipo d'errore. Se l'utente dovesse imbattersi in queste situazioni, sarebbe sufficiente riavviare nuovamente il *plugin* provvedendo a fornire i dati corretti.
Gli errori gestiti sono di due tipi:

- Uno dei due *layers* forniti in ingresso non possiede gli attributi richiesti. L'utente è invitato a riavviare il *plugin* fornendo in ingresso il *layer* corretto (*figura 22*)

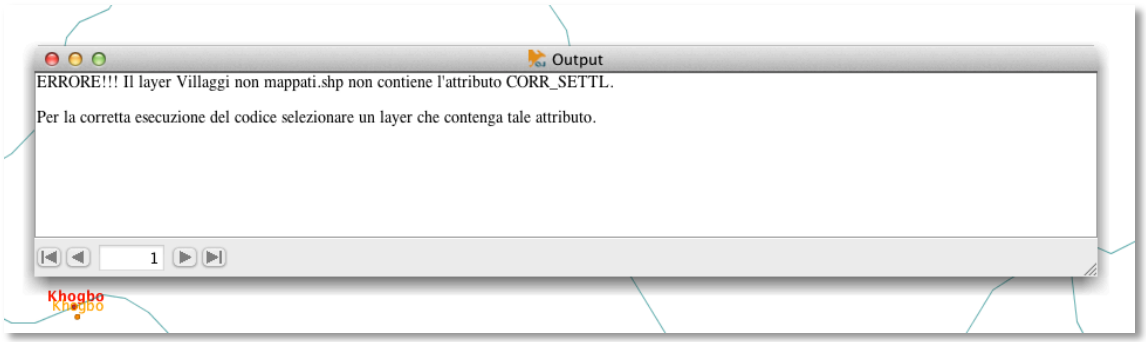

Figura 22 - Messaggio d'errore

- Non viene selezionato alcun villaggio su cui applicare l'algoritmo. L'utente è invitato a riavviare il *plugin* scegliendo almeno un villaggio.

BIBLIOGRAFIA

http://www.mediciconlafrica.org - Website del CUAMM

http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf - Descrizione tecnica del *file shape* di ESRI

http://www.openjump.org - Website di OpenJUMP

http://www.postgresql.org - Website di PostgreSQL

http://postgis.refractions.net - Website di PostGIS

Cay Horstmann, Concetti di informatica e fondamenti di JAVA 5a ed., Apogeo, 2010

http://www.oracle.com/technetwork/java/javase/overview/index.html - Website di Oracle in riferimento a Java

http://www.php.net - Website di PHP

http://www.eclipse.org - Website di Eclipse

P.E. Bailey & al. (2011, ottobre). Using a GIS to model interventions to strengthen the emergency referral system for maternal and newborn health in Ethiopia. *Internation Journal of Gynecology and Obstetrics*, p. 10

P.Atzeni, S.Ceri, S.Paraboschi, R.Torlone, Basi di dati - Modelli e linguaggi di interrogazione 3a ed., McGraw Hill, 2009

R.A. Elmasri, S.B. Navathe, Sistemi di basi di dati - Fondamenti 4a ed., Pearson, 2004

http://en.wikipedia.org/wiki/Travelling_salesman_problem - Il TSP da Wikipedia

http://www.vividsolutions.com/jump/bin/JUMP%20Developer%20Guide.pdf - JUMP Developer's Guide

http://sourceforge.net/apps/mediawiki/jump-pilot/index.php?title=How_to_use_and_make_own_Plugins - Guida alla creazione di *plugins* per OpenJUMP