ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
SEDE DI CESENA
FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Magistrale in Scienze e Tecnologie Informatiche

PROGETTO E REALIZZAZIONE DI UN FRONT-END CARTOGRAFICO WEB E MOBILE BASATO SU HTML5 PER SERVIZI DI LOGISTICA DISTRIBUTIVA

Tesi di Laurea in
Algoritmi per Applicazioni Aziendali

Relatore:
Prof. Vittorio Maniezzo

Presentato da:
Alessandro Valenti

Sessione III
Anno Accademico 2011-2012

A Francesca che mi ha dato la determinazione,
la forza e il supporto per intraprendere e terminare
questo percorso di studi.

Grazie

Indice

Introduzione

Questo progetto nasce con l'intento di fornire un applicativo web basato su GIS (Geographic Information System) per la gestione di servizi di logistica distributiva. Questo applicativo deve essere fruibile da diversi dispositivi per diverse tipologie di utilizzatori. Nasce quindi il progetto "GeoWeb5", "Geo" in quanto per visualizzare i percorsi di logistica distributiva ci si appoggerà sulla tecnologia GIS, "Web" in quanto deve essere fruibile attraverso Internet, "5" in quanto si basa sul nuovo standard Html5.

GeoWeb5 deve essere quindi un applicativo web basato su standard che permetta la visualizzazione tramite un normale PC o tramite un dispositivo mobile, quest'esigenza per rispondere a diverse esigenze funzionali; dall'ufficio un utente dell'amministrazione a seguito di imprevisti, deve poter alterare le soluzioni di logistica distributiva, soluzioni precedentemente calcolate da algoritmi basandosi su dati ottimi; lo spedizioniere dal proprio mezzo tramite dispositivo mobile deve poter conoscere le proprie missioni ed eventualmente visualizzare in tempo reale le modifiche apportate dall'utente dell'amministrazione.

Per realizzare questo progetto si sono dovute utilizzare tecnologie di tipo web oriented, di framework geospaziali, di architetture e di infrastrutture che permettano di legare queste entità per fornire un pacchetto completo, altamente integrato ed estensibile.

La prima parte di questo documento si concentrerà sullo stato attuale della cartografia digitale e dei sevizi GIS, si esporranno i problemi pratici della georefenziazione, della gestione dei dati e metadati geospaziali; inoltre si vedranno alcuni progetti open source e si vedrà come hanno approcciato questi problemi, parleremo quindi di:

- Tile Map Service;
- PostgreSQL e Postgis;
- GeoServer;

Si entrerà nel dettaglio di GIS come "Geographic Information System, un sistema adatto per catturare, immagazzinare, manipolare, analizzare, gestire e rappresentare tutti i tipi di dati geografici. L'acronimo GIS è spesso usato per significare la scienza o gli studi sulle informazioni geografiche (dette anche geospaziali)."

Il servizio di logistica distributiva utilizza queste coordinate geografiche come input e come output, come input per conoscere le distanze tra i vari punti e come output ci fornisce un set di oggetti geometrici che rappresentano le soluzioni. Entrando un attimo nel dettaglio del progetto si introdurrà il concetto di warehouse rappresentato come un punto avente coordinate geospaziali; un set di punti che rappresenteranno l'entità clienti anch'essi rappresentati a loro volta da coordinate geospaziali ed infine un set di soluzioni memorizzate come un insieme di punti uniti da linee (MULTILINE).

Precedentemente citati PostgreSQL e Postgis, database relazionali utilizzati per la gestione della persistenza, sarà il database di frontiera condiviso dall'applicativo aziendale (gestione clienti) e dal software di calcolo di logica distributiva; oltre naturalmente utilizzato dall'applicativo di front-end deputato alla visualizzazione delle soluzioni.

Per non legarsi ad una specifica tecnologia come PostgreSQL e Postgis si è utilizzato GeoServer (progetto open source) e allo standard WFS (Web Feature Service) in esso implementato per interrogare ed alterare lo strato di persistenza dati.

Risolto l'aspetto riguardante i dati da visualizzare ci si è concentrati sulla cartografia, cioè la mappa, in cui i dati devono essere posizionati per una corretta User Interface; si parlerà quindi dello standard "Tile Map Service (TMS)" il quale fornisce una gerarchia di organizzazione dei tile (frammenti di mappa) che dovranno essere ricomposti per creare la mappa di sfondo alle soluzioni; si parlerà quindi di OpenStreetMap (progetto open source) come fornitore di questi tile e saranno inoltre introdotte offerte più o meno commerciali messe a disposizione dai principali vendor informatici come Google, Microsoft ed ESRI.

A questo punto si dispone di tutti i tasselli per realizzare il progetto, si hanno a disposizione i dati, gli strumenti per la rappresentazione, resta solo da identificare le tecnologie con cui si realizzerà l'applicativo e le modalità di realizzazione.

La prima parte di una buona implementazione consiste nell'identificare una buona architettura, bilanciata tra computazione lato server e computazione lato client, infatti, si è adottato un approccio basato su un'architettura multi layer SOA (Service Oriented Architecture), in particolare un'architettura three tier.

Presentation Layer
Htlm5, CSS, JavaScript, Ajax

Middle Tier
ASP.NET MVC4
Services Web (SOA)
Api Web

Data Layer
GeoServer
PostgreSQL Postgis
OpenStreetMap
Tile Provider

Questo modello architetturale rappresenta anche l'indice di due importanti capitoli "Tecnologie Utilizzate" e "Progetto e Realizzazione".

In "Tecnologie Utilizzate" verrà percorso questo stack dall'alto verso il basso mostrando lo stato dell'arte della programmazione client side, quindi, della programmazione nel browser.

Si parlerà di:

- Html5 e DOM (Document Object Model), ovvero, del codice che sarà interpretato staticamente dal browser e fornito dal Middle Tier;
- CSS3 (Cascading Style Sheets), ovvero, le tecniche di impaginazione della pagina web e non solo, si introdurrà "Media Query" per la gestione dei dispositivi mobili, si vedrà l'importante introduzione dei selettori e di molti altri aspetti;
- JavaScript e JQuery come motori applicativi della parte di progetto lato client;

Si parlerà di Middle Tier, cioè della parte applicativa lato server, il fornitore dell'Html, il fornitore di servizi ed Api che si occuperanno di interagire con il dominio dei dati, in particolare, ci si soffermerà sulla soluzione Microsoft per la realizzazione queste parti.

Sarà presentato:

- ASP.NET MVC4
    - MVC (Pattern Model View Controller);
    - Razor View Engine;
    - Api Controller;
- IIS (Internet Information Services) ossia il server Web, e non solo, che ospiterà l'applicativo;

Il capitolo terminerà approfondendo le tecnologie di persistenza con particolare riguardo GeoServer che così bene si adatta all'architettura SOA, mettendo a disposizione un servizio SOAP (Simple Object Access Protocol) chiamato WFS (Web Feature Service).

Si è parlato di architettura SOA, di architettura multi tier, di pattern. Tutti questi argomenti saranno trattati nel dettaglio nel capitolo "Progetto e Realizzazione", in particolare questo capitolo sarà formato da quattro macro aree:

- Progettazione, in cui si parlerà del progetto e degli scopi, approfondiremo l'architettura e i design pattern utilizzatati;
- Installazione e Configurazioni, dove si prepareranno gli ambienti e servirà come introduzione al capitolo successivo;
- Realizzazione, dove si parlerà dell'implementazione di GeoWeb5 seguendo il modello three tier visto precedentemente:
    - Data Layer, le tecniche utilizzate dall'applicativo per l'accesso ai dati;
    - Middle Tier, le funzionalità messe a disposizione dalla parte server;
    - Presentation Layer, la programmazione client side e le tecniche di Web Design lato interfaccia;

Nel corso di questa tesi verranno trattati argomenti tecnologici, architetturali, infrastrutturali che permettano la realizzazione di questo progetto. Si entrerà nei dettagli applicativi, saranno mostrarne le possibilità ampliative grazie ad una progettazione "astratta", quindi, ad un disaccoppiamento tra interfacce e classi "concrete"; possibilità ampliative dovute anche ad una forte separazione dei domini applicativi che permettono una crescita parallela dell'applicativo.

Infine nel capitolo conclusioni si ripercorreranno gli aspetti funzionali e progettuali più significativi.

Cartografia Digitale

Il moderno termine scientifico geomatica, deriva dal prefisso “geo” (terra) e dal suffisso “matica” (informatica), ed indica l’applicazione estesa dell’informatica alle discipline del rilevamento. Tale termine è stato coniato negli anni ’80 per descrivere un approccio interdisciplinare, che va dal rilievo all’analisi e gestione del territorio attraverso l’acquisizione, l’elaborazione, l’archiviazione, la rappresentazione e la gestione dei dati spazialmente riferiti (georeferenziazione).

Le tecniche e le discipline che la costituiscono sono:

- La geodesia (studio della forma e delle dimensioni della terra);
- La topografia (insieme delle procedure del rilevamento diretto del territorio);
- La cartografia (rappresentazione della superficie terrestre secondo regole prefissate);
- La fotogrammetria (analisi metrica degli oggetti effettuata mediante immagini fotografiche);
- GPS (Global Positioning System);
- Telerilevamento (acquisizione digitale a distanza di dati riguardanti il territorio e l’ambiente);
- L’informatica (insieme di programmi che consentono la realizzazione di Sistemi Informativi Territoriali (SIT) o Geografici (GIS).

Il Geomatico sarà il moderno topografo, un nuovo professionista che si occupa del rilevamento e del trattamento dei dati, per la rappresentazione del territorio.
In molti paesi europei questa figura già esiste: Chartered Surveyor (paesi anglosassoni), Geomètre Expert (paesi francesi), Vermessungsingenieur (paesi tedeschi), Geodetisch Ingenieur (paesi bassi). In Italia per il momento manca tale figura, ma si va verso una specializzazione e chiarificazione di tale soggetto sempre maggiore all’interno del sistema delle mansioni ingegneristiche. A favore dello sviluppo di questo nuovo tipo di preparazione scientifica, matematica, fisica, informatica e di formazione multidisciplinare e quindi di nuovi sbocchi professionali, ci sono Stato, Regioni, Province, Comuni e imprese private. Questa nuova figura

alla cui formazione stanno contribuendo quasi tutti i corsi di laurea del settore, si serve di diverse discipline, tutte condensate in sistemi che per la loro natura sono stati denominati Sistemi Informativi Geografici (GIS - Geographic Information System). Questi sistemi si basano sui sistemi di disegno computerizzato (CAD) e i data base relazionali (DBMS). Il primo sistema permette il disegno computerizzato delle entità geografiche il secondo l'immagazzinamento dei dati e delle informazioni legate a tali entità. I sistemi informativi geografici ampliano il concetto di territorio, diventa un oggetto composito che presenta, a seconda del modo in cui viene guardato, facce (si potrebbe usare il termine tecnico di coperture) diverse. Ogni copertura è una classe di oggetti diversi: si potrebbe vedere un livello di rappresentazione dedicato alla morfologia generale, un altro riguardante l'uso del suolo, su un altro il tracciato per mezzo del quale raggiungere un punto, un livello dedicato ad informazioni di carattere marittimo, ad esempio presenza di secche, o di correnti troppo forti. In termini ingegneristici le diverse coperture sono: strade, pozzi, sorgenti, edifici, tipo di flora, rappresentazione di zone a rischio geologico, e così via. In questo modo emergono diversi aspetti e facce del territorio, potendone facilmente evidenziare i cambiamenti nel tempo, ed ecco che un GIS non è un puro e semplice strumento per la gestione di un'informazione statica bi o tridimensionale, bensì una struttura che rappresenta una realtà dinamica.

Stato dell'arte

Gli ambiti applicativi dei GIS sono i più diversi, coinvolgendo tutte le possibili attività che hanno implicazione diretta o indiretta con il territorio. Tra gli esempi di applicazioni GIS troviamo l'utilizzo in agricoltura in cui vengono usati per: stimare le colture stagionali, generando le carte tematiche digitali della copertura agricola dalle immagini da satellite, elaborando statistiche spaziali in funzione delle aree amministrative, e restituendo i dati di previsione ai fini di indirizzare le politiche dei prezzi dei prodotti della terra; pianificare gli interventi di risanamento analizzando la capacità reale e l'uso del suolo con tecniche di sovrapposizione automatica (overlay); gestire i finanziamenti agli operatori del settore verificando, con ricerche geostatistiche, dove e perché questi sono necessari. Nella protezione civile i GIS vengono usati per: generare coperture di erosione potenziale attraverso la sovrapposizione (overlay mapping) di dati territoriali relativi agli aspetti geologici, topografici, di uso del suolo, idrografici, e così via; studiare le vulnerabilità degli acquiferi sotterranei attraverso l'uso di modelli geografici che integrano i dati relativi alle falde sotterranee, con gli aspetti geologici, idrogeologici e morfologici, di uso del suolo; viene utilizzato per simulazione di eventi catastrofici analizzando le aree abitate interessate e prevedendo come, dove e quando evacuare le zone interessate e intervenendo per arginare i danni. Nei vari settori ambientali i GIS vengono usati per: pianificare e definire confini di aree di interesse naturalistico da porre sotto protezione attraverso l'analisi integrata delle varie componenti ambientali e antropiche; monitorare geograficamente attraverso carte di isovalore l'inquinamento dell'aria, delle risorse idriche, del mare e della terra, analizzando le fonti potenziali e pianificando gli interventi; controllare l'evoluzione della pressione antropica sui territori di particolare valore ambientale. Nelle amministrazioni comunali i GIS vengono usati per: realizzare e gestire i "Piani", le varianti ed i percorsi attuativi attraverso l'analisi della situazione di fatto e delle necessità prioritarie; gestire i tributi in funzione della localizzazione e della tipologia delle proprietà.

Nelle aziende che gestiscono reti tecnologiche i GIS vengono usati per: inventariare e gestire le reti sul territorio; supportare le attività di manutenzione segnalando alle squadre di intervento la precisa dislocazione dei tratti di rete ed i componenti

presenti; simulare guasti, attraverso opportuni modelli, e programmare gli interventi necessari; pianificare l'incremento della rete nelle zone di espansione. Nel settore delle analisi socio-demografiche e di mercato i GIS vengono usati per: analizzare la distribuzione geografica dei dati statistici rilevati attraverso le indagini censuarie e individuarne le interdipendenze con opportuni modelli geografici; effettuare analisi di geo-marketing utili al controllo delle aree geografiche in cui si distribuisce l'utenza reale e potenziale, dell'influenza territoriale della concorrenza, dello studio della localizzazione ottimale di un nuovo servizio.

Nel settore dei trasporti i GIS sono utilizzati per: controllare le flotte di mezzi sul territorio visualizzando in ogni istante la localizzazione geografica di ogni veicolo; analizzare i percorsi ottimali in funzione di varie impedenze quali la distanza, il traffico, la pendenza, la tortuosità, i limiti di velocità imposti, ecc.; gestire gli interventi di manutenzione ordinaria e le concessioni stradali. Nel settore della progettazione di opere e infrastrutture i GIS sono utilizzati per: progettare la localizzazione ottimale di grandi opere in funzione di varie informazioni territoriali quali la distribuzione demografica, l'uso del suolo, le caratteristiche geomorfologiche, le distanze dai potenziali utenti, le interconnessioni con altre opere e infrastrutture, ecc.; valutare l'impatto ambientale attraverso tecniche di sovrapposizione automatica dei vari tematismi ambientali con l'opera progettata e l'analisi geografica delle incompatibilità. Nel settore delle telecomunicazioni sono utilizzati per: studiare la copertura territoriale delle antenne di trasmissione in funzione delle caratteristiche di propagazione delle onde radio, delle impedenze dovute alla morfologia del terreno ed alla vegetazione, ed alla concentrazione della popolazione.

Le applicazioni finora descritte, costituiscono un elenco tutt' altro che esaustivo che ha unicamente lo scopo di mostrare in quanti campi i GIS sono oggi operativi e suggerire quanti altri settori potrebbero essere in futuro coinvolti da questa nuova tecnologia.

La raccolta dei dati e del successivo post-processamento mediante applicazioni di tipo GIS mobile, la memorizzazione delle informazioni riguardanti i percorsi saranno organizzate in una struttura di dati interrogabile, cioè il database, i dati territoriali e alfanumerici saranno memorizzati e gestiti dal database di tipo relazionale, in esso i dati sono organizzati in una serie di tabelle, dove le righe sono i record e le colonne

sono i campi relativi ai singoli dati; le tabelle possono essere legate tra loro con campi di legami comuni.

Il problema della georeferenziazione

Le caratteristiche fondamentali delle carte o dei sistemi informativi geografici e territoriali sono il contenuto, l'attualità e la precisione. Il contenuto dipende in larga misura dalla scala e dallo scopo per cui si redige il lavoro cartografico, l'attualità dipende dalla data del rilevamento o dell'inserimento delle informazioni nel sistema, la precisione e l'accuratezza invece ci dicono quanto bene sono georeferenziati o georeferenziabili i particolari oggetti in un predefinito sistema di riferimento. I termini di precisione ed accuratezza non debbono essere confusi, nel caso di un set di misure ripetute il primo ci dà un'idea di quanto siano coerenti tra loro le misure, il secondo invece ci informa di quanto si è vicini al "valore vero".
Per georeferenziazione di informazioni territoriali si intende quel complesso di attività che consentono di stabilire una serie di corrispondenze biunivoche tra:

- Un'informazione territoriale, intesa come da inserire in un sistema informativo di definita risoluzione;
- Il fenomeno territoriale attraverso cui si manifesta e/o si materializza l'informazione;
- L'epoca del rilevamento del fenomeno;
- La stima della posizione spaziale che compete al fenomeno, definita da una sequenza di coordinate di affidabilità nota, in un assegnato sistema di riferimento.

Nel passato tale problematica riguardava scienziati e costruttori di carte, e l'utente si limitava all'utilizzo di un posizionamento relativo in ambito puramente locale; attualmente invece quello della georeferenziazione è di fatto uno dei primi e principali problemi che devono affrontare, e risolvere in maniera rigorosa, tutti quelli che traggono gli indiscutibili benefici dall'accesso alle informazioni territoriali raccolte nelle banche dati. Per confrontare e utilizzare informazioni diverse provenienti da più soggetti, è necessaria una base geometrica comune, estesa a tutto

il territorio interessato; tale omogeneità può essere realizzata unicamente adottando un unico sistema di coordinate, e riferire quindi a tale sistema tutti gli oggetti presenti sul territorio. Si va dunque da un contesto locale ad una georeferenziazione globale, e a spingere in tale direzione è la continua crescita dei processi di globalizzazione, e soprattutto la disponibilità tecnologica che rende possibile un tipo di posizionamento non più legato alla distanza ed alla intervisibilità.

La gestione dei dati e la costruzione dell'informazione

Un sistema informativo geografico è costituito da un database di dati georiferiti. Tali dati devono essere raccolti, organizzati e conservati e i sistemi informatici garantiscono che ciò venga eseguito in modo che i dati vengano conservati in modo permanente in dispositivi per la loro memorizzazione, aggiornati per riflettere rapidamente le loro variazioni e resi accessibili alle interrogazioni degli utenti, e distribuirli in modo capillare sul territorio. Elemento fondamentale dunque, è la disponibilità di informazioni e la capacità di gestirle in modo efficiente. Ogni organizzazione è dotata di un sistema informativo che organizza e gestisce le informazioni necessarie per pianificare le attività e raggiungere gli scopi prefissati. Il sistema informativo può essere visto come un insieme di informazioni, che vengono rappresentate mediante i dati. In tale ambito è necessario distinguere i due termini il cui significato porta il dato ad esser qualcosa di immediatamente presente alla conoscenza prima di ogni elaborazione e l'informazione ad esser notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni. Insomma i dati da soli non hanno alcun significato, ma una volta contestualizzati costituiscono informazioni. Chiarita tale differenza si capisce cosa si intende con il termine di base di dati (database): una collezione di dati utilizzati per rappresentare le informazioni di interesse per un sistema informativo. La gestione dei dati è affidata ad un sistema software in grado di lavorare con collezioni di dati che siano gradi, condivise e persistenti, assicurando la loro affidabilità e privatezza. Tale sistema prende il nome di database management system (DBMS). Una base dati è una collezione di dati gestita da un DBMS.

Integrazione del database e sviluppo in ambiente GIS

La progettazione della struttura dati esula, in linea teorica, dalla scelta del particolare software col quale viene implementata. In genere si progetta la base dati nei suoi aspetti concettuale e logico, e successivamente le si dà una connotazione di tipo fisico quando viene caricata in un sistema di gestione di database (DBMS). E' in questo momento che si testa la solidità della struttura informativa, mediante uno strumento molto semplice: le query. Si pongono le domande per le quali la struttura informativa ha ragione di esistere, in genere derivate da un'analisi preliminare dei requisiti applicativi, e se si ottengono le giuste risposte, si può dire che il modello informativo è stato progettato correttamente. Il passo successivo è costituito dalla connessione del database al sistema GIS ovvero:

> "Geographic Information System è un sistema adatto per catturare, immagazzinare, manipolare, analizzare, gestire e rappresentare tutti i tipi di dati geografici. L'acronimo GIS è spesso usato per significare la scienza o gli studi sulle informazioni geografiche (dette anche geospaziali)."

A tale scopo si utilizza un driver di connessione; grazie a questa interfaccia di programmazione delle applicazioni è possibile accedere mediante linguaggio SQL, dalle applicazioni ai dati di diversi sistemi di gestione di database. A questo punto il sistema GIS, caricate le componenti (geo)grafiche, è pronto per essere utilizzato per ogni tipo di elaborazione. Come si può vedere, la creazione della struttura dati che girerà in un GIS, viaggia su due livelli, da una parte quello che riguarda la strutturazione concettuale e logica della base dati, dall'altra si assiste ad un completamento, anche se sarebbe il caso di intenderlo come un'alternativa di progettazione, che si basa prevalentemente sulla strutturazione degli elementi della base dati a livello di software. Ad esempio la costruzione di un elemento lineare a livello concettuale richiede molte informazioni (nodi terminali, verso, e tutte le altre caratteristiche topologiche intrinseche dell'elemento). D'altra parte, l'utilizzo di un software GIS, semplifica le cose in quanto un elemento lineare, una volta digitalizzato ha in se quelle caratteristiche (come il verso) che altrimenti bisogna sempre specificare. Alla luce di questa considerazione, è importante avere sempre coscienza del significato e della collocazione delle operazioni eseguite, all'interno

della fase concettuale o applicativa. Sono due campi chiamati a coesistere nella progettazione della base dati; ma bisogna gestire l'ampiezza d'azione di ognuno con criterio; la relativa semplicità proposta dai dei sistemi GIS commerciali, si scontra spesso con la diversa implementazione delle relazioni, a parità di modello topologico, dei vari applicativi. Questo infatti, può portare a problemi di trasportabilità dei dati da un sistema all'altro. In conclusione, agendo a livello concettuale si elabora un prodotto universalmente riconoscibile dai diversi applicativi GIS, ma ciò richiede un lavoro di progettazione preliminare maggiore. E' comunque sempre possibile, utilizzare il software, per implementare il modello topologico della base dei dati.

GIS

GIS ci offre un nuovo modo di guardare il mondo che ci circonda.

Un sistema informativo geografico (GIS) integra hardware, software e dati per l'acquisizione, la gestione, l'analisi e la visualizzazione di tutte le forme di informazioni di tipo geografico. GIS ci permette di vedere, capire, interrogare, interpretare e visualizzare i dati in molti modi che rivelano rapporti, modelli e tendenze in forma di mappe, globi, report e grafici. Un GIS consente di rispondere alle domande e risolvere i problemi, cercando in i dati in un modo che è rapidamente compreso e facilmente condivise. La tecnologia GIS può essere integrato in un quadro del sistema delle informazioni aziendali.

Con il GIS è possibile:

- Trovare luoghi che hanno le caratteristiche che stai cercando;
- Trovare luoghi che soddisfano determinati criteri e prendere provvedimenti, potremmo desiderare di trovare i codici di avviamento postale di molte giovani famiglie con un reddito relativamente elevato. I funzionari della sanità pubblica potrebbero voler mappare il numero di medici per 1.000 persone in ogni tratto censimento per individuare quali aree siano adeguatamente servite, e quelle che non lo sono;
- Generare una mappa di densità consente di misurare il numero di funzioni che utilizzano una unità uniforme in modo da poter vedere chiaramente la distribuzione. Ciò è particolarmente utile quando si mappatura delle aree, come ad esempio tratti censiti o contee, che variano notevolmente di dimensioni. Nelle mappe che mostrano il numero di persone per censimento delle vie, delle vie più grandi possono avere più persone di quelle più piccole. Ma alcuni tratti più piccoli potrebbero avere più persone per miglio quadrato, una densità maggiore;
- Monitorare ciò che sta accadendo e di adottare misure specifiche per la mappatura cosa c'è dentro una determinata area. Ad esempio, un procuratore distrettuale avrebbe monitorato arresti per droga per scoprire se

un arresto si trova a 1.000 metri di una scuola, in tal caso, si applicano sanzioni più severe;

- Aiutare a scoprire cosa sta accadendo all'interno di una distanza impostata di una funzionalità di mappatura cosa c'è nei dintorni;
- Aiutare a decidere una linea di condotta, o per valutare i risultati di un'azione o di una politica. Con la mappatura dove e come le cose si muovono in un periodo di tempo, è possibile ottenere informazioni sul modo in cui si comportano. Ad esempio, un meteorologo potrebbe studiare i percorsi degli uragani di prevedere dove e quando potrebbero verificarsi in futuro;

La geografia è la scienza del nostro mondo. Accoppiato con GIS, la geografia ci aiuta a comprendere meglio la terra e applicare le conoscenze geografiche a una serie di attività umane. Il risultato è la nascita di "The Geographic Approach", un nuovo modo di pensare, un approccio al problem solving che integra informazioni geografiche in modo di intendere e gestire il nostro pianeta. Questo approccio ci permette di creare conoscenze geografiche, misurando la terra, l'organizzazione di questi dati, e analisi e modellazione dei vari processi e le loro relazioni. L'approccio geografico ci permette anche di applicare questa conoscenza per il nostro modo di progettare, pianificare, e cambiare il nostro mondo.

"The Geographic Approach", consiste, in diverse fasi:

1. "AskApproaching" un problema geograficamente inquadrare comporta la questione da un punto di vista "location based". Qual è il problema che si sta tentando di risolvere o analizzare, e dove si trova? Essere il più specifici possibile sulla questione si sta cercando di rispondere vi aiuterà con le fasi successive. Quando si è di fronte a decidere come strutturare l'analisi, che i metodi analitici da utilizzare, e come presentare la risultati al target di riferimento;
2. Acquisizione, dopo aver definito con chiarezza il problema, è necessario determinare i dati necessari per completare l'analisi e l'accertamento di cui i dati possono essere trovati o generati. Il tipo di dati e la portata geografica del progetto contribuirà a indirizzare i vostri metodi di raccolta dei dati e condurre l'analisi. Se il metodo di analisi richiede informazioni dettagliate

e/o ad alto livello, può essere necessario creare o calcolare i nuovi dati. Creazione di nuovi dati può significare semplicemente calcolare i nuovi valori nella tabella di dati o di ottenere nuovi livelli di mappa o gli attributi, ma possono anche richiedere geoprocessing;

3. "ExamineYou", non si sa con certezza se i dati acquisiti sono appropriati per il vostro studio a fondo fino a che non si esamina. Questo include l'ispezione visiva, così come indagare come i dati sono organizzati (lo schema), quanto bene i dati corrisponde a altri insiemi di dati e le regole del mondo fisico (la sua topologia), e dalla storia del luogo in cui i dati provenienti;
4. Analizzare i dati vengono elaborati e analizzati in base al metodo di esame o di analisi che si sceglie, che dipende dai risultati che si spera di raggiungere. Guardando i risultati possono aiutare a decidere se le informazioni sono valide o utile, o se è necessario eseguire nuovamente l'analisi di parametri diversi o anche un metodo diverso. Strumenti di modellazione GIS rendono relativamente facile fare questi cambiamenti;
5. I risultati "ActThe" e la presentazione dell'analisi sono parti importanti di questo approccio. I risultati possono essere condivisi attraverso le relazioni, mappe, tabelle e grafici e consegnati in forma cartacea o digitale su una rete o su Internet. È necessario decidere il modo migliore per presentare la vostra analisi. È possibile confrontare i risultati di diverse analisi e vedere quale metodo presenta le informazioni in modo più accurato. E è possibile personalizzare i risultati per diverse tipologie di pubblico. Ad esempio, un pubblico potrebbe richiedere un rapporto convenzionale che sintetizza le analisi e trasmette raccomandazioni o alternative comparabili. Un altro pubblico, potrebbe essere necessario un formato interattivo che permette loro di chiedere che cosa-se domande o l'esercizio di ulteriori analisi.

Servizi

La cartografia di gitale negli ultimi anni ha ottenuto un successo tale da muovere l'intera comunità informatica e le principali software house ad implementare sistemi software e proprie basi dati come veri e propri framework per lo sviluppo delle applicazioni basate sulla geolocalizzazione visti precedentemente.

Open Street Map

OpenStreetMap è libero, fornisce una mappa modificabile di tutto il mondo. A differenza di set di dati proprietari come Google Map Maker, la licenza di OpenStreetMap permette il libero accesso al dataset mappa completa. Questa enorme quantità di dati possono essere scaricati per intero, ma è disponibile anche in forme immediatamente utili come mappe e servizi commerciali. Il modo principale che gli utenti partecipano OpenStreetMap è modificando la mappa. Con un account utente gratuito, è possibile apportare miglioramenti alla mappa che risolvere i problemi e aggiungere i dati per tutti. Alcuni utenti prendono unità GPS a passeggio, unità o gite in bicicletta per registrare le tracce che possono poi essere importati in OpenStreetMap. Altri aiutano ad identificare strade tramite analisi sulle caratteristiche di immagini satellitari. Il finanziamento di OpenStreetMap e le infrastrutture è supportato dal non-profit Fondazione OpenStreetMap. Anche se non mantiene la proprietà del set di dati o il controllo del progetto, la Fondazione favorisce la crescita, lo sviluppo e la distribuzione di dati geospaziali liberi per chiunque di utilizzare e condividere. OpenStreetMap è non solo i dati aperti, ma è costruito su software open source. L'interfaccia del software di sviluppo web, la mappatura del motore, API, editor, e molti altri componenti della mappa sono resi possibili grazie al lavoro dei volontari.

Google Maps

Google Maps (già Google Local) è un servizio web mapping di applicazioni e la tecnologia fornita da Google, che alimenta molti servizi "map based", tra cui il sito

web di Google Maps, Google Ride Finder, Google Transit e mappe incorporate su di terze parti siti web tramite l'API di Google Maps. Offre mappe stradali, un route planner per viaggiare a piedi, in auto, in bicicletta o con i mezzi pubblici e un localizzatore per le imprese urbane in numerosi Paesi del mondo.

Esri

Esri è il fornitore leader a livello mondiale di sistemi software GIS e applicazioni per la gestione di basi di dati geolocalizzate. La compagnia ha un milione di utenti e più di 350,000 organizzazioni, di cui molte agenzie federali statunitensi e agenzie di mappatura statunitense, e tutti i dipartimenti della salute e dei trasporti dei 50 staticamente degli Stati Uniti, aziende forestali e governi locali, scuole e università, organizzazioni non governativa. Esri ha sede a Redlands, in California. La Società fu Fondata venire Environmental Systems Research Institute (Istituto di Ricerca per Sistemi Ambientali) Nel 1969 i Prodotti Esri (in particolare ArcGis Desktop) aveva il 40,7% del mercato globale. Nel 2002 ha approssimativamente il 30% del mercato globale GIS.

Microsoft Bing

Fondamentalmente è la risposta di Microsoft a Google Maps, fornendo un framework altamente integrato, un insieme di Api REST e un insieme di controlli utilizzabili nella piattaforma .Net

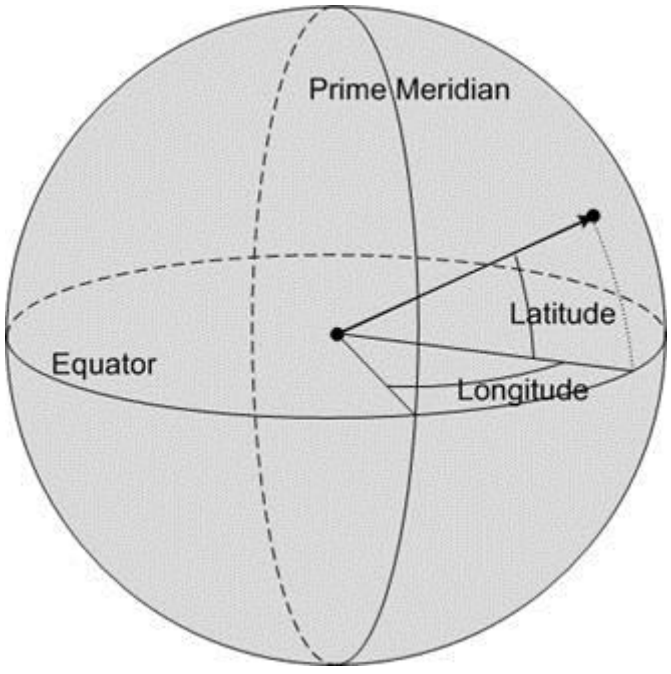
Tile Map Service

Purtroppo non esiste uniformità sulla gestione dei tile, però possiamo identificare TMS come uno standard per l'organizzazione dei tile partendo dalle coordinate x, y, z.

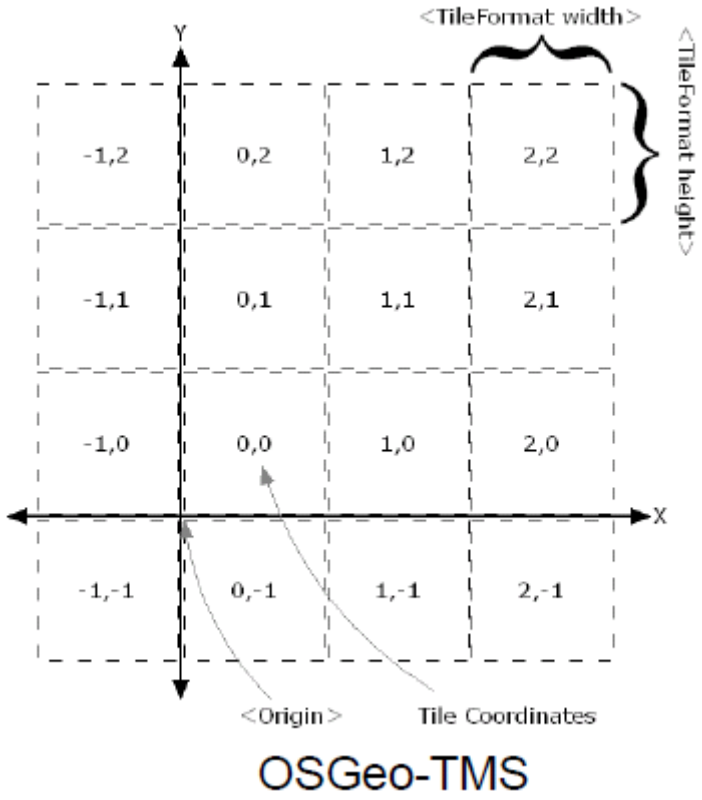
Nel sistema di identificazione tile x, y, z:

- "z" è il livello di zoom della griglia
- "x" ed "y" sono la colonna e il numero di riga in cui questo tile deve essere posto, misurato da un origine in alto a sinistra della mappa.

Le specifiche Tile Map Service (TMS) sono definite dalla Open Source Geospatial Foundation ed è uno standard per servire i tile della mappa. Secondo le specifiche TMS, "Sull'asse x l'aumento della numerazione del tile corrisponde all'aumento della coordinata longitudine del sistema di riferimento spaziale, e la coordinata y l'aumento della numerazione dei tile aumenta con la coordinata latitudine del sistema di riferimento spaziale". In altre parole, i tile (0, 0), a qualsiasi livello di zoom deve essere sempre posizionato in basso a sinistra della mappa, non in alto a sinistra.

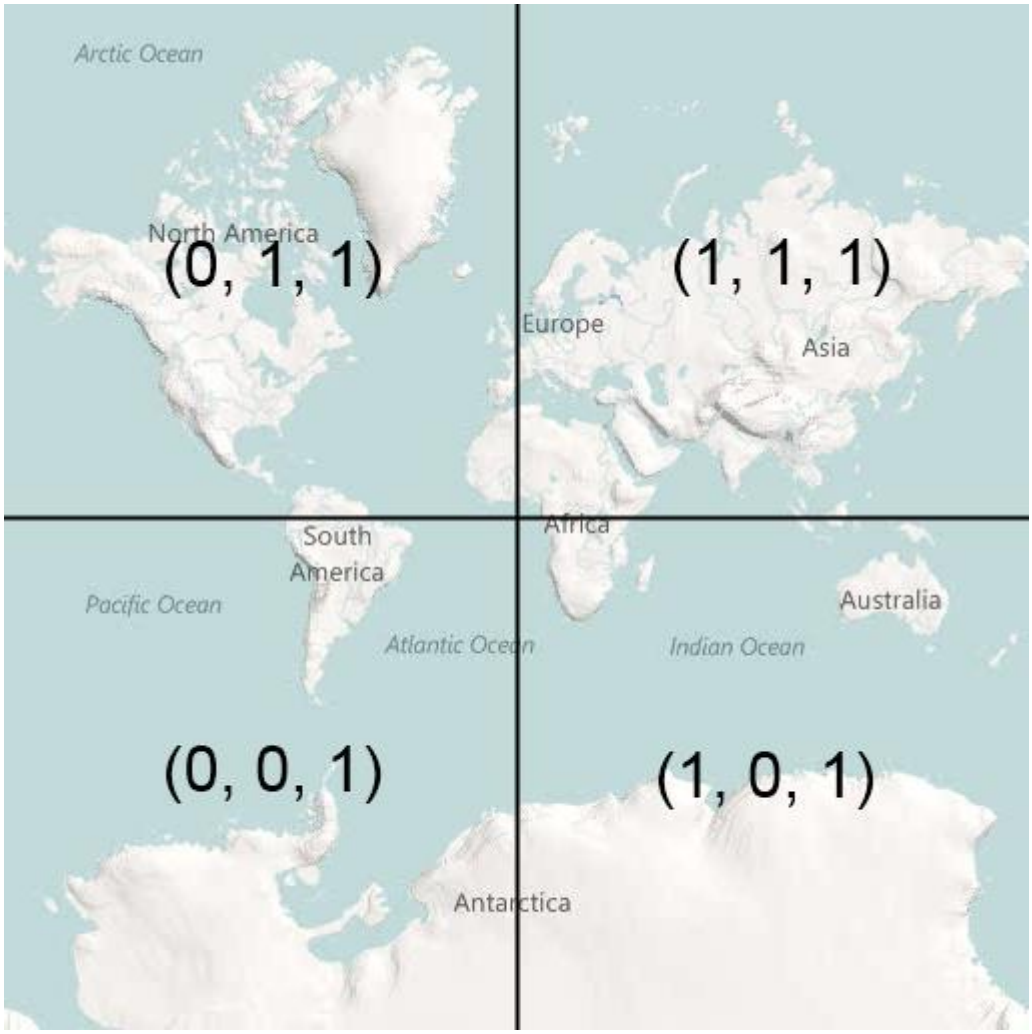


Questo sistema è utilizzato da Google Maps, Open Street Maps, ESRI, e molti altri. In realtà, questa numerazione tile suona molto simile (ed è spesso erroneamente descritto come) ad un indice di TMS.



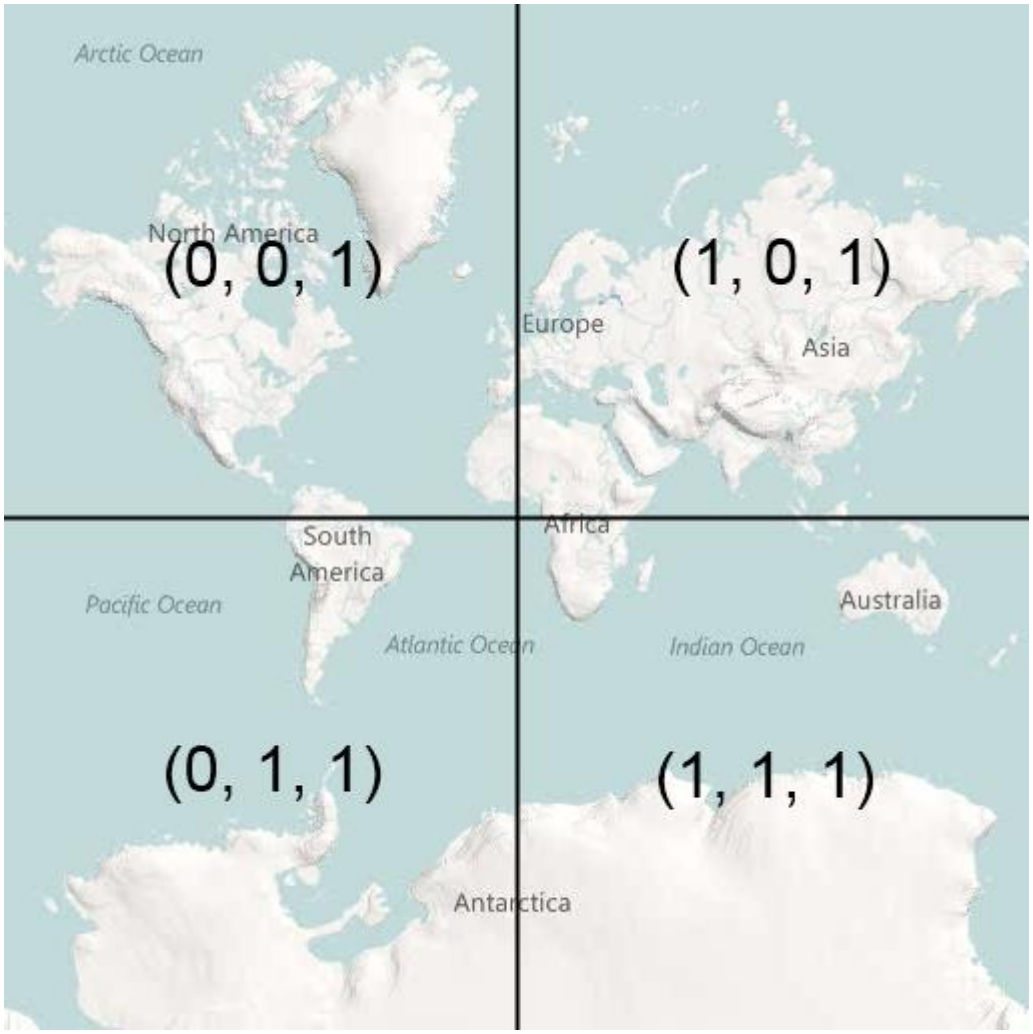
OSGeo-TMS

Utilizzando il sistema TMS, gli indici di tile a livello di zoom 1 diventano come segue:



Il problema è che, dal momento che questi sistemi sono identici in quasi tutti gli aspetti, si tende dare per scontato che il sistema utilizzato da Google / Bing / OSM ed altri siano lo standard e dei sistemi basati su tile, ma purtroppo poi ci si accorge che si riferiscono erroneamente al formato TMS. Ci si può imbattere in un pezzo di software o di un servizio che emette veramente tile numerati secondo lo standard TMS e, se non si corregge l'origine tile, la mappa va a finire a testa in giù sul lato sbagliato della il mondo.

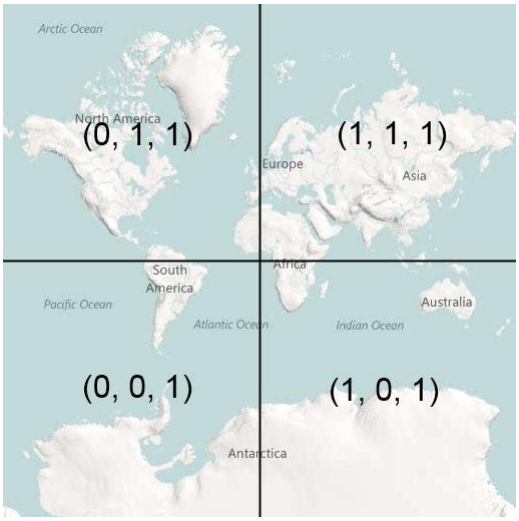
Se a livello di zoom 1, è possibile descrivere i quattro tile necessari per fare una mappa completa in Bing con x, y, z le coordinate nel modo seguente:


Arctic Ocean
North America
(0, 0, 1)
(1, 0, 1)
Europe
Asia
Africa
South
America
Pacific Ocean
Atlantic Ocean
Indian Ocean
Australia
(0, 1, 1)
(1, 1, 1)
Antarctica

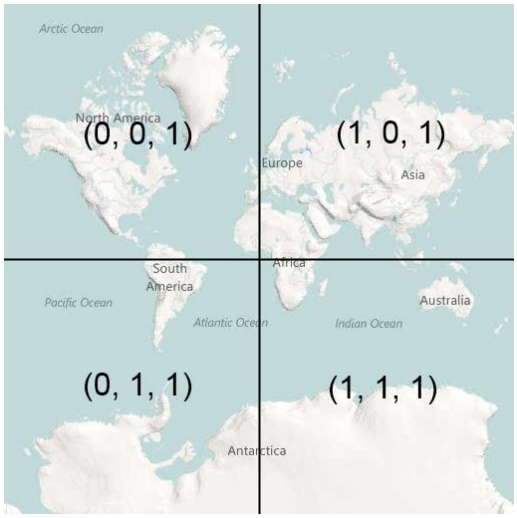
Per fortuna la correzione è molto semplice da fare. In ogni dato livello di zoom della mappa, lo zoom, è possibile invertire l'indice y di una tile da TMS a Google/Bing come segue:

```
var   ymax  = 1 << zoom;
var   y     = ymax - y - 1;
```

TMS – OpenStreetMap
(x, y, z)



Bing, Google
(x, y, z)



Tecnologie Utilizzate

Questo progetto ha utilizzato le ultime tecnologie fornite da Microsoft per la realizzazione di vere e proprie applicazioni web. In particolare il progetto è suddiviso a layer ognuno dedicato a svolgere funzioni particolari. Il progetto si è concentrato soprattutto nel "Presentation Layer" aderendo al nuovo standard HTML5 e CSS3. Lato server il progetto è stato implementato utilizzando il Microsoft .Net Framework 4 e come linguaggio di sviluppo Visual C#. A livello dati l'applicativo è in grado di accedere direttamente a un database relazionale con estensioni GIS (nel nostro caso Postgis) oppure utilizzando lo standard WFS esposto da GeoServer.

Client Side - Presentation Layer

HyperText Markup Language (HTML5)

HTML5 sarà il nuovo standard per l'HTML (HyperText Markup Language). Le versioni precedenti di HTML risalgono all'anno 1999. Ad oggi i principali browser sono già in grado di interpretare i nuovi elementi e le nuove Api definiti dallo standard. HTML5 nasce dalla collabrazione tra World Wide Web Consortium (W3C) ed il Web Hypertext Application Technology Working Group (WHATWG). WHATWG lavorò principalmente sulle Web Form e le Applicazioni Web e W3C lavorò con XHTML 2.0. Nel 2006 decisero di collaborare e creare la nuova versione di HTML. I principali obiettivi di questo accordo era uniformare lo sviluppo Web e che le nuove funzioni fossero basate su HTML, CSS, DOM e JavaScript riducendo il più possibile il bisogno di componenti esterni come il plug-in Flash. Tra gli altri obiettivi c'era una migliore gestione degli errori, una migliore gestione dei markup e soprattutto rendere HTML5 indipendente dal dispositivo, quest'ultima caratteristica ha reso possibile la realizzazione di questo progetto.
Le principali innovazioni utilizzate introdotte da HTML5 e utilizzate nel progetto sono il tag `canvas` e il tag `navigator`; sono stati rispettivamente utilizzati per il rendering bidimensionale della mappa e dei percorsi di ottimizzazione, mentre tramite il tag `navigator` è possibile localizzare sulla mappa la posizione corrente dell'utente. L'oggetto `canvas` espone un pannello e le primitive per poterci disegnare all'interno, addirittura, questo pannello rende disponibile anche la grafica tridimensionale. L'oggetto `navigator` invece comunica e localizza con l'eventuale GPS del dispositivo mobile o tramite Api di geolocalizzazione legate all'indirizzo IP del dispositivo, le coordinate comunicate da questa Api saranno poi interpretate e rimesse in scala sul `canvas` del progetto mostrando oltre ai percorsi di ottimizzazione anche la posizione corrente dell'utente.

Cascading Style Sheets (CSS3) - Responsive Web Design (RWD)

Una pagina web è formata fondamentalmente da due elementi: i contenuti veri e propri e la formattazione, ovvero l'aspetto con cui i contenuti sono mostrati all'utente. I browser che interpretano il codice (X)HTML mostrano all'utente formattazioni predefinite per ogni tag che incontrano. Tuttavia questa formattazione è completamente sotto il controllo dell'utente, che può modificarla tramite le impostazioni del browser. Per permettere agli autori di definire l'aspetto delle loro pagine, dal 1993 in poi Netscape Navigator ed Internet Explorer, i due browser che si disputavano gli utenti nella nota guerra dei browser, presentarono tag proprietari, ovvero non aderenti agli standard e non compatibili con i browser concorrenti. Questi tag proprietari di formattazione erano l'unico modo per gli autori di definire la formattazione e così il loro uso è diventato massiccio. Tuttavia presentano diversi problemi tra cui la lunghezza di questi tag che possono inficiare sulle performance di download e interpretazione della pagina. Il problema sempre più rilevante è stato la mancanza di compatibilità con i nuovi computer palmari e gli smartphone. In quegli anni le pagine sono progettate per schermi con risoluzione minima 800x600 pixel. I palmari, che hanno una risoluzione inferiore ed una forma dello schermo ben diversa dal rapporto 4:3 dei monitor per computer, si trovano quindi impossibilitati a visualizzare correttamente la pagina e l'utente dovrà tentare di "decodificarla", operazione spesso molto scomoda. Per tentare di risolvere questa situazione, nel 1996 il W3C emanò le specifiche CSS 1. I CSS 1 erano un interessante sistema per separare contenuto da formattazione. La base di questo linguaggio, infatti, consisteva nel fatto che il contenuto sarebbe stato sempre definito dal codice (X)HTML, mentre la formattazione si sarebbe trasferita su un codice completamente separato, il CSS appunto. I richiami tra i due codici venivano effettuati tramite due particolari attributi: class e ID. Queste specifiche purtroppo non prendevano però in considerazione il problema di formati e risoluzioni video diversi, dato che nel 1996 i PDA (i palmari) erano scarsamente diffusi. Per includere le nuove funzionalità e rendere i CSS un linguaggio ben supportato, nel 1998 il W3C emanò le specifiche CSS 2 e nel 2004 le specifiche CSS 2.1. I CSS 2 sono la naturale evoluzione dei CSS 1 ed offrono potenti soluzioni per risolvere il problema dei PDA, con la possibilità di creare fogli di stile separati per i dispositivi portatili. Anche il secondo problema

è ormai pienamente risolvibile, scrivendo una pagina (X)HTML esclusivamente indirizzata alla struttura e ai contenuti e manovrandola poi esclusivamente con i CSS per impaginarla. Con la comparsa di Internet Explorer 5, di Firefox e di Opera 7, i CSS 2 hanno potuto avvalersi di browser in grado di interpretarli e sono quindi entrati a far parte del codice di molti siti web.

Una volta uscite le specifiche CSS3 che introducono diverse nuove istruzioni grafiche e, soprattutto, una parte fondamentale per la realizzazione del progetto, grazie a questi nuovi standard diventa possibile il Responsive Web Design (RWD) utilizzato per indicare una particolare tecnica di Web Design per la realizzazione di siti web "responsive", cioè , in modo che le pagine adattino automaticamente al layout; la capacità di fornire una visualizzazione ottimale in funzione dell'ambiente nei quali vengono visualizzati (pc su desktop con diverse risoluzioni, tablet, smartphone, cellulari di vecchia generazione, web tv) riducendo al minimo all'utente la necessità di ridimensionamento e scorrimento, in particolare quello orizzontale. Un sito progettato per il RWD utilizza CSS3 Media Queries, un'estensione della regola `@media rule` per adattare il layout all'ambiente nel quale viene visualizzato.

Le Media Queries consentono alla pagina di usare diversi stili CSS sulla base delle caratteristiche del device sul quale vengono visualizzati, principalmente sulla larghezza del viewport, ovvero della finestra del browser nella quale viene visualizzata la pagina. Come risultato, gli utenti che utilizzano diverse periferiche e browser per la visualizzazione hanno accesso ad un singolo sorgente i cui contenuti vengono disposti in modo tale da essere facilmente consultabili, e si possa navigare senza dover fare troppe operazioni di ridimensionamento, scorrimento e spostamento delle pagine.

Javascript - JQuery

JavaScript è un linguaggio di scripting orientato agli oggetti comunemente usato nei siti web. E' stato utilizzato come linguaggio di riferimento Client Side dell'intero progetto e tramite il Framework JQuery rende l'applicativo cross-browser e cross-device, inoltre permette la minimizzazione del roundtrip dei dati ed evitando i

postback dando all'utente finale una sensazione di fluidità e di responsività che permetterà alle applicazioni Web di sostituire le classiche applicazioni Stand-Alone. Fu originariamente sviluppato da Brendan Eich della Netscape Communications con il nome di Mocha e successivamente di LiveScript, ma in seguito è stato rinominato "JavaScript" ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java. JavaScript è stato standardizzato per la prima volta tra il 1997 e il 1999 dalla ECMA con il nome ECMAScript. L'ultimo standard, di marzo 2011, è ECMA-262 Edition 5.1. Il cambio di nome da LiveScript a JavaScript si ebbe più o meno nel periodo in cui Netscape stava includendo il supporto per la tecnologia Java nel suo browser Netscape Navigator. La scelta del nome si rivelò fonte di grande confusione. Non c'è una vera relazione tra Java e JavaScript; le loro somiglianze sono soprattutto nella sintassi (derivata in entrambi i casi dal linguaggio C); le loro semantiche sono piuttosto diverse, e in particolare i loro object model non hanno relazione e sono ampiamente incompatibili. Dato il successo di JavaScript come linguaggio per arricchire le pagine web, Microsoft sviluppò un linguaggio compatibile, conosciuto come JScript. La necessità di specifiche comuni fu alla base dello standard ECMA 262 per ECMAScript, di cui sono state pubblicate tre edizioni da quando il lavoro iniziò, nel novembre 1996. La caratteristica principale di JavaScript è quella di essere un linguaggio interpretato: il codice non viene compilato, ma interpretato (in JavaScript lato client, l'interprete è incluso nel browser che si sta utilizzando). La sintassi è relativamente simile a quella del C, del C++ e del Java. Il linguaggio definisce le funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo del paradigma object oriented. JavaScript è un linguaggio debolmente tipizzato. JavaScript è un linguaggio debolmente orientato agli oggetti. Il codice viene eseguito direttamente sul client e non sul server. Il vantaggio di questo approccio è che, anche con la presenza di script particolarmente complessi, il server non viene sovraccaricato a causa delle richieste dei clients. Di contro, nel caso di script che presentino un sorgente particolarmente grande, il tempo per lo scaricamento può diventare abbastanza lungo. Un altro svantaggio è il seguente: ogni informazione che presuppone un accesso a dati memorizzati in un database remoto deve essere rimandata ad un linguaggio che effettui esplicitamente la transazione, per poi

restituire i risultati ad una o più variabili JavaScript; operazioni del genere richiedono il caricamento della pagina stessa. Con l'avvento di AJAX tutti questi limiti sono stati superati. A differenza di altri linguaggi, quali il C o il Java, che permettono la scrittura di programmi completamente stand-alone, JavaScript viene utilizzato soprattutto in quanto linguaggio di scripting, integrato, quindi, all'interno di un altro programma. L'esempio tipico (e, forse, il più noto) di programma ospite per uno script JavaScript è quello del browser. Un browser tipicamente incorpora un interprete JavaScript; quando viene visitata una pagina web che contiene il codice di uno script JavaScript, quest'ultimo viene portato in memoria ed eseguito dall'interprete contenuto nel browser. Le interfacce che consentono a JavaScript di rapportarsi con un browser sono chiamate DOM (Document Object Model). Molti siti web usano la tecnologia JavaScript lato client per creare potenti applicazioni web dinamiche. Un uso principale del Javascript in ambito Web è la scrittura di piccole funzioni integrate nelle pagine HTML che interagiscono con il DOM del browser per compiere determinate azioni non possibili con il solo HTML statico: controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, ecc. Sfortunatamente, gli standard DOM imposti dal W3C non sempre vengono rispettati dai vari browser; per questo motivo si sono sviluppati diversi framework con l'obiettivo di rendere il JavaScript cross-browser. JQuery è una libreria di funzioni (framework) Javascript, cross-browser per le applicazioni web, che si propone come obiettivo quello di semplificare la programmazione lato client delle pagine HTML. Pubblicato, per la prima volta il 22 agosto 2005 da John Resig, ha raggiunto la versione 1 (stabile) il 26 agosto dell'anno successivo. Tramite l'uso della libreria JQuery è possibile, con poche righe di codice, effettuare svariate operazioni, come ad esempio ottenere l'altezza di un elemento, o farlo scomparire con effetto dissolvenza. Anche la gestione degli eventi è completamente standardizzata, automatica; stessa cosa per quanto riguarda l'utilizzo di AJAX, in quanto sono presenti alcune funzioni molto utili e veloci che si occupano di istanziare i giusti oggetti ed effettuare la connessione e l'invio dei dati. Il framework fornisce metodi e funzioni per gestire al meglio aspetti grafici e strutturali come posizione di elementi, effetto di click su immagini, manipolazione del Document Object Model e quant'altro ancora, mantenendo la compatibilità tra browser diversi e standardizzando gli oggetti messi a disposizione dall'interprete Javascript del browser. Il core di JQuery fornisce per ottenere elementi tramite un selettore per ottenere un elemento referenziandolo come parametro oppure per

creare ex novo un elemento partendo da codice HTML grezzo. Il framework inoltre riconosce gli oggetti di tipo event e provvede a modificare le loro proprietà rendendoli uniformi, semplificando la loro gestione, la loro propagazione, e fornendo un'utile modalità per impedire al browser di continuare. L'assegnazione di eventi quali click, load, mouseover è gestita in maniera efficace e non invadente. È possibile definire facilmente effetti personalizzati specificando la proprietà css da manipolare, come è pure possibile specificare la durata dell'effetto e una funzione di callback da eseguire dopo l'animazione. La gestione delle chiamate asincrone è davvero semplificata, e sono fornite le funzioni per caricare contenuti dinamicamente, per eseguire richieste asincrone per l'interazione con Javascript e funzione per caricare un oggetto JSON unzione per caricare un file Javascript remoto ed eseguirlo automaticamente. Anche gli eventi AJAX sono comodamente gestiti, per il completamento, gli errori e l'invio.

Conclusioni

All'interno del progetto sono state utilizzate tutte queste tecnologie e nel successivo capitolo "Progettazione e Realizzazione" si presenterà in pratica come queste parti hanno interagito tra loro per creare un'applicazione web cross-browser e cross-device.

Middle Tier - Business Logic

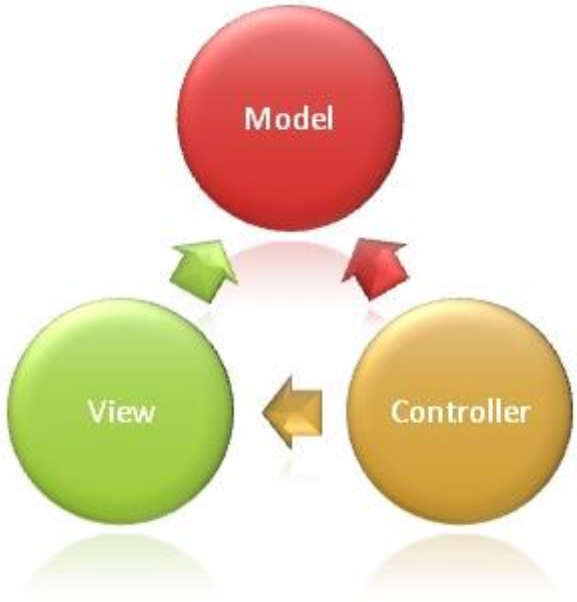
In questo capitolo saranno esposte le tecnologie lato server che hanno permesso la realizzazione del progetto, in particolare, i pattern utilizzati per la realizzazione del progetto; progetto nel quale ci si è focalizzati sull'usabilità, le performance e l'estendibilità, fattori di successo per qualsiasi applicativo. Per "Business Logic" nel contesto di questo progetto, si intende, creare un framework cartografico per la rappresentazione di percorsi di ottimizzazione; l'applicativo deve dare la possibilità di alterare le soluzioni calcolate lato server, la possibilità di usufruire di questi risultati tramite un dispositivo mobile; per questi motivi la parte server dell'applicazione è stata differenziata dal punto funzionale ma non dal punto di vista strutturale, cioè, il rendering grafico è comune; differiscono le funzionalità tra i diversi dispositivi.

Microsoft ASP.NET MVC4

L'architettura Model-View-Controller (MVC) separa l'applicazione in tre principali component: il modello, la vista ed il controller. L'ASP.NET MVC framework è leggero e veloce (rispetto alle precedenti WebForms ASPX), fornisce un presentation layer testabile ed è completamente integrato alle funzioni rese disponibili da IIS e dall'intero framework .Net. Le precedenti WebForms erano basate sul concetto di postback e di viewstate, le quali permettevano di simulare la gestione ad eventi di una normale applicazione Windows, rendevano quindi la programmazione web stateless una sorta di programmazione statefull in cui lo stato veniva memorizzato nel viewstate e ritrasmesso ad ogni postback simulando così lo stato dell'applicazione. Questo approccio appesantiva l'applicativo e la rete trasmettendo l'intero stato tra il client e il server; è quindi un approccio più pesante e ha il difetto di scarsa testabilità (unitaria, funzionale) oltre che separare poco la parte prettamente di web design dalla parte di logica server; questa promisquità rende anche difficile la scrittura del JavaScript in quanto con l'approccio WebForms anche il JavaScript viene gestito e rilegato a migliorare l'esperienza utente per minimizzare i postback che sono un vero e proprio reload della pagina con un effetto di sfarfallio che rende l'esperienza della navigazione

meno piacevole. L'utilizzo del pattern MVC ha eliminato il viewstate rendendo le pagine più snelle e conferendo a JavaScript e JQuery una parte importante nella computazione dell'applicativo, delegando al client (browser) parte della computazione e soprattutto suddividendo gli eventi prima gestiti solo lato server anche al client. Naturalmente entrambi gli approcci hanno loro vantaggi e svantaggi, questi approcci non sono esclusivi e possono essere combinati in quanto entrambi basato sul Microsoft .Net Framework.

Tornando all'MVC framework e parlando delle tre principali componenti: il controller incaricato di rispondere ad un determinato url (detta anche rotta), un modello dei dati calcolati lato server e una vista incaricato di scrivere il codice HTML che il browser interpreterà .



L'oggetto Models è la parte di applicazione che implementa la logica del dominio; generalmente si occupa di ricevere e memorizzare le informazioni nello storage, prepara i dati che successivamente il web designer trasformerà in HTML.
La Views è il componente che crea l'interfaccia utente (UI). Viene create partendo dal Model Object renderizzando le strutture dati messi a disposizione dal modello trasformandoli in controlli UI (tag) interpretabili da un browser, ad esempio una lista di stringhe può essere renderizzata in modi diversi a seconda del dispositivo o a seconda dell'utilità all'interno del contesto, cioè, utilizzando un drop-down list (<SELECT><OPTION>) se da questa lista deve essere selezionato un elemento o in una text-box (<SPAN>) se si vuole solo visualizzare lo stato. Il Controller è il

componente che risponde all'interazione utente, lavora con il modello e infine seleziona la Views per renderizzare l'interfaccia grafica. L'MVC pattern aiuta a creare applicazioni che separano i differenti aspetti delle applicazione e provvede a tenere debolmente accoppiati questi elementi. Il pattern specifica dove ogni parte della logica deve essere messa nell'applicazione: la UI logica nella vista, la gestione degli input nel controller e la business logic nel model. Questa separazione aiuta a gestire la complessità durante la creazione dell'applicazione in quanto permette di mantenere il focus su un aspetto di implementazione alla volta. L'altro aspetto fondamentale in questa separazione di responsabilità sta nel fatto che è possibile rendere la business logic testabile; al contrario nell'approccio WebForms risultava complesso in quanto testare una pagina significava istanziare l'intera pagina, tutti i controlli figli, le classi da cui dipendeva; per tutti questi motivi risultava difficile scrivere test che avessero scope su una specifica parte dell'applicazione, scrivere test risultava molto dispendioso e spesso scarsamente efficaci a causa di troppe dipendenze, ora, utilizzando questo approccio di sviluppo è possibile testare piccole parti della logica, proprio partendo dai controller si può individuare quali parti del modello testare. L'altro notevole vantaggio ad avere un debole accoppiamento tra le tre parti dell'applicativo consiste nella possibilità di promuovere il parallelismo di sviluppo, i web designer possono scrivere l'interfaccia su oggetti di modello finti, parallelizzare quindi lo sviluppo web dallo sviluppo funzionale.
Questa separazione inoltre permette per gli sviluppatori lato server un approccio migliore per il test-driven development (TDD), cioè, avendo separato completamente la logica dell'applicativo dalla logica UI di approcciare la business logic partendo dai test funzionali, all'inizio rossi per poi, implementando la logica di business, fare sì che tutti i test funzionali e unitari precedentemente scritti diventino verdi, questo garantisce un alto coverage unitari e funzionali. MVC inoltre mette a disposizione un potente URL mapping componente che consente di creare applicazioni che hanno URL comprensibile e consultabile. Gli URL non necessitano di includere il file con estensioni, e sono progettati per supportare modelli di URL che funzionano bene per l'ottimizzazione dei motori di ricerca (SEO) e per le chiamate Representational State Transfer (REST). L'evoluzione non ha generato incompatibilità con il passato, infatti sono comunque state rese disponibili tutte le funzioni core del fremework e anche l'utilizzo del markup esistenti pagina ASP.NET (file aspx), controllo utente (file ascx), e la pagina master (file master) oltre ai file di markup come modelli di vista. È possibile utilizzare le funzionalità ASP.NET

esistenti con il framework ASP.NET MVC, come ad esempio le pagine master nidificate, in linea espressioni (<%= %>), i controlli server, modelli dichiarativi, di associazione dati, localizzazione, e così via. ASP.NET MVC consente di utilizzare funzioni core come precedentemente detto come autenticazione basata su Form e l'autenticazione di Windows, l'autorizzazione degli URL, l'appartenenza e ruoli, la produzione e la memorizzazione nella cache dei dati, la gestione dello stato della sessione e profilo, il sanity check, il sistema di configurazione, e l'architettura del provider.

Internet Information Service 7

Internet Information Services (IIS) 7 è un'infrastruttura server che consente ai siti di utilizzare i protocolli HTTP e HTTPS ed è estensibile per l'utilizzo di altri protocolli. Un motore di server Web che può essere personalizzato con l'aggiunta o la rimozione di moduli integrati di elaborazione delle richieste condotte da IIS e ASP.NET. IIS contiene diversi componenti che svolgono funzioni importanti per l'applicazione e ruoli del server Web in Windows Server 2008 (IIS 7.0) e Windows Server 2008 R2 (IIS 7.5). Ogni componente ha delle responsabilità, come ad esempio l'ascolto per le richieste effettuate al server, gestione dei processi, e la lettura di file di configurazione. Questi componenti includono handler di protocollo, come HTTP.sys e servizi, come il servizio Pubblicazione sul Web (servizio WWW) e Windows Process Activation Service (WAS). Ascoltatori protocollo ricevere richieste specifiche del protocollo, inviarli a IIS per l'elaborazione, e poi tornare risposte richiedenti. Ad esempio, quando un client richiede una pagina del browser Web da Internet, il listener HTTP, HTTP.sys, raccoglie la richiesta e la invia a IIS per l'elaborazione. Una volta che IIS elabora la richiesta, HTTP.sys restituisce una risposta al browser del client. Per impostazione predefinita, IIS fornisce HTTP.sys come l'ascoltatore protocollo che ascolta le richieste HTTP e HTTPS. HTTP.sys è stato introdotto in IIS 6.0 come listener di protocollo per le richieste HTTP. HTTP.sys rimane il listener HTTP in IIS 7, ma include il supporto per Secure Sockets Layer (SSL). Per supportare i servizi e le applicazioni che utilizzano protocolli diversi da HTTP e HTTPS, è possibile ad esempio utilizzare tecnologie come Windows Communication Foundation (WCF). WCF dispone di più listener che forniscono le funzionalità sia di un listener di protocollo e un adattatore

ascoltatore. Il listener HTTP è parte del sottosistema di rete dei sistemi operativi Windows, ed è implementato come un driver in modalità kernel dispositivo chiamato stack del protocollo HTTP (HTTP.sys). HTTP.sys ascolta le richieste HTTP provenienti dalla rete, passa le richieste su IIS per l'elaborazione, e quindi restituisce le risposte elaborate ai browser client. In IIS 6.0, HTTP.sys era sostituito dalle API di Windows Sockets (Winsock), che era un componente in modalità utente utilizzato anche da versioni precedenti di IIS per ricevere le richieste HTTP e inviare risposte HTTP. IIS 7 avvalendosi del modulo HTTP.sys per le richieste HTTP offre quindi la gestione delle richieste in modalità kernel cache. Le richieste di risposte memorizzate nella cache vengono serviti senza dover passare alla modalità utente. Le richieste causano meno over head in cambio di contesto perché il kernel inoltra le richieste direttamente al processo di lavoro corretto. Se nessun processo di lavoro è disponibile ad accettare una richiesta in modalità kernel la coda di richieste contiene la richiesta fino a quando un processo di lavoro lo raccoglie filtrando le richieste di pre-elaborazione e di sicurezza. In IIS 7 e superiore, funzionalità precedentemente gestite dal Servizio Web Publishing (servizio WWW) da solo è ora divisa tra due servizi: Servizio WWW e Windows Process Activation Service (WAS). Questi due servizi sono eseguiti come LocalSystem nel processo Svchost.exe stessa, e condividono gli stessi file binari. Il servizio WWW legge le informazioni di configurazione dalla metabase di IIS e utilizza tali informazioni per configurare e aggiornare il listener http, inoltre, il servizio WWW è monitor e gestisce i processi di lavoro che elaborano le richieste HTTP. Il servizio WWW monitora le prestazioni e fornisce contatori delle prestazioni per i siti web e per la cache di IIS. Il servizio WWW gestisce pool di applicazioni e processi di lavoro, quali l'avvio, l'arresto e riciclo dei processi di lavoro. Inoltre, il servizio Web monitora lo stato dei processi di lavoro. In IIS 7 il servizio Windows Process Activation Service (WAS) gestisce pool di configurazione delle applicazioni e processi di lavoro al posto del servizio WWW. Questo consente di utilizzare la stessa configurazione e modello di processo per HTTP. All'avvio, WAS legge alcune informazioni dal file ApplicationHost.config, e passa le informazioni agli adattatori listener sul server. Adattatori listener sono componenti che stabiliscono la comunicazione tra WS e ascoltatori di protocollo. Una volta adattatori listener ricevere informazioni di configurazione, si configurano i loro ascoltatori di protocollo relativi e preparare gli ascoltatori per ascoltare le richieste. Nel caso di WCF, un adattatore listener include la funzionalità di un ascoltatore protocollo. Ad esempio

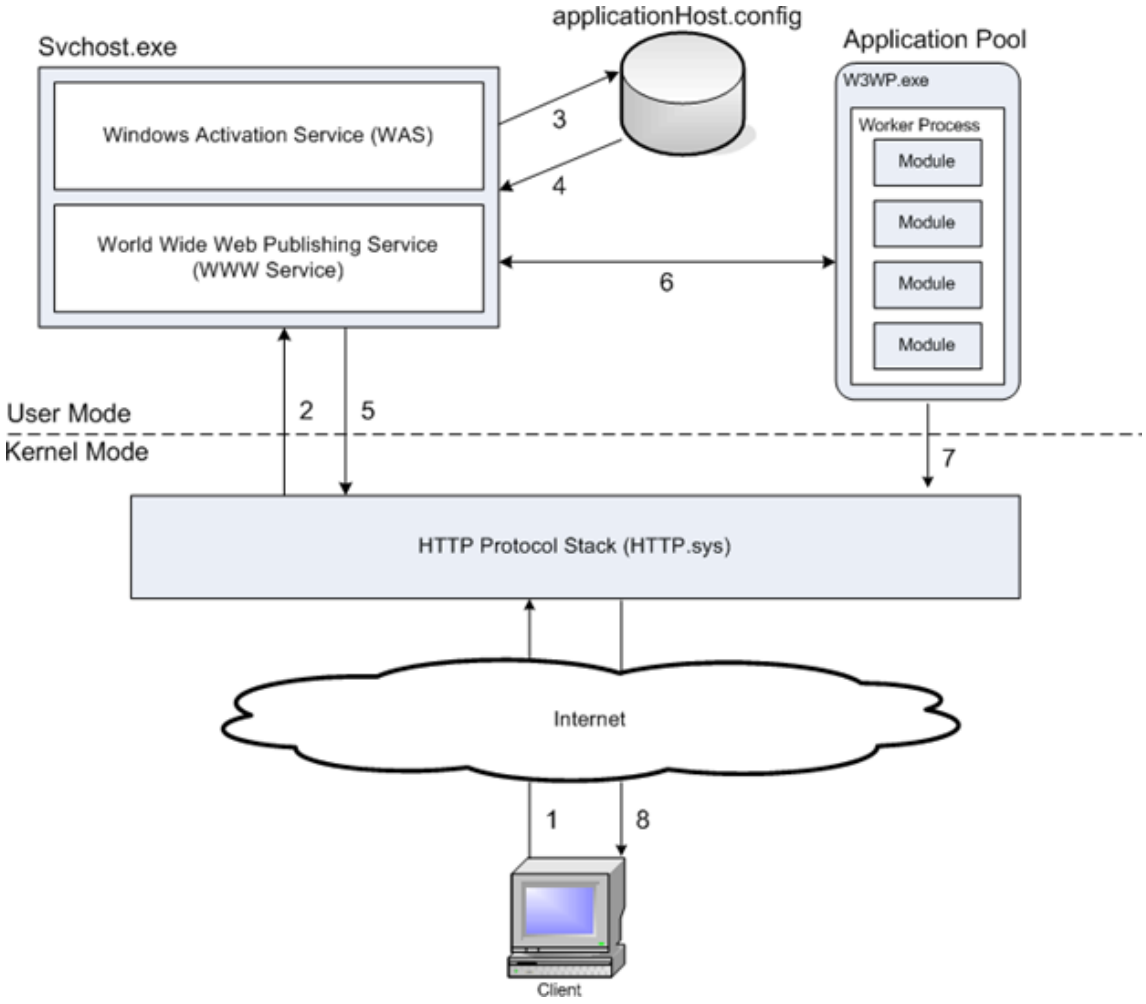
un adattatore listener WCF, come NetTcpActivator, è configurato in base alle informazioni da WAS. Una volta NetTcpActivator è configurato, ascolta le richieste che utilizzano il protocollo net.tcp. I tipo di informazioni che sono state legge da configurazione: informazioni di configurazione globale, le informazioni sulla configurazione del protocollo per i protocolli HTTP e non, configurazione di applicazioni, la configurazione del sito, configurazione dell'applicazione. Se ApplicationHost.config cambia WAS riceve una notifica e aggiorna gli adattatori ascoltatore con le nuove informazioni. IIS fornisce una nuova architettura che è diverso rispetto alle versioni precedenti di IIS. Invece di tenere la maggior parte delle funzionalità all'interno del server stesso, IIS includono un motore di server Web in cui è possibile aggiungere o rimuovere i componenti, chiamati moduli, a seconda delle esigenze. I moduli sono caratteristiche individuali che il server utilizza per elaborare le richieste. Per esempio, IIS utilizza moduli di autenticazione per autenticare le credenziali del client, e moduli per la gestione della cache attività della cache. La nuova architettura offre diversi vantaggi rispetto alle versioni precedenti di IIS. È possibile controllare quali moduli si desidera sul server, è possibile personalizzare un server per un ruolo specifico nel proprio ambiente, è possibile utilizzare i moduli personalizzati per sostituire i moduli esistenti o introdurre nuovi caratteristiche. La nuova architettura migliora anche la sicurezza e semplifica l'amministrazione. Rimuovendo moduli non necessari, è possibile ridurre la superficie di attacco del server e di occupazione di memoria, che è la quantità di memoria che utilizza processi di lavoro del server sulla macchina. È anche possibile eliminare la necessità di gestire le funzioni non necessarie per i siti e le applicazioni. In IIS 7 ci sono diversi moduli per eseguire attività specifiche per HyperText Transfer Protocol (HTTP) nella richiesta pipeline di elaborazione. Moduli HTTP includono moduli per rispondere alle informazioni e richieste inviate nelle intestazioni dei client, per tornare errori HTTP, per reindirizzare le richieste, e altro ancora. Altri moduli in IIS deputati a eseguire operazioni relative alla sicurezza nella richiesta pipeline di elaborazione separati per ciascuno degli schemi di autenticazione, che consentono di selezionare i moduli per i tipi di autenticazione che si desidera sul vostro server. Alcuni esempi, il nome del modulo descrive già il suo utilizzo: AnonymousAuthenticationModule, BasicAuthenticationModule, CertificateMappingAuthenticationModule, DigestAuthenticationModule, IISCertificateMappingAuthenticationModule, RequestFilteringModule,

UrlAuthorizationModule, WindowsAuthenticationModule, IpRestrictionModule. I moduli di content includono moduli per elaborare le richieste di file statici, per restituire una pagina di default quando un client non specifica una risorsa in una richiesta, per elencare il contenuto di una directory, e altro ancora. Due moduli in IIS sono deputati eseguire la compressione nella richiesta pipeline di elaborazione, nell'elaborato si è sperimentato il loro utilizzo, poi in un secondo momento sono stati rimossi in quanto il beneficio nell'utilizzo non giustificava l'adozione, in particolare i moduli di compressione ottengono in loro maggior beneficio su stream testuali, nell'elaborato invece utilizzando tile di immagini già compressi (png, jpg) il beneficio risultava minimo. Altri moduli in IIS sono deputati ad eseguire operazioni relative alla memorizzazione nella cache nella richiesta pipeline di elaborazione. La memorizzazione nella cache migliora le prestazioni dei siti Web e applicazioni Web per la memorizzazione delle informazioni elaborate, come ad esempio le pagine Web, in memoria sul server e quindi riutilizzare le informazioni in richieste successive per la stessa risorsa, anche questi moduli non sono stati utilizzati in quanto si è preferito utilizzare i moduli managed messi a disposizione dal framework .Net. Una importantissima coppia di moduli di IIS gestiscono l'integrazione di moduli di codice gestito (managed modules) nella richiesta di elaborazione pipeline di IIS; grazie a questa integrazione oltre ai moduli nativi, IIS consente di utilizzare i moduli di codice gestito per estendere le funzionalità di IIS. In particolare prima è stato citato FormsAuthentication che supporta l'autenticazione utilizzando l'autenticazione Forms come su Aspx. Ad esempio System.Web.Caching.OutputCacheModule è stato utilizzato per la gestione managed della cache. L'architettura di elaborazione delle richieste è costituito da un elenco ordinato di moduli nativi e gestiti che eseguono compiti specifici in risposta alle richieste. Questo design offre diversi vantaggi rispetto alle versioni precedenti di IIS. In primo luogo, tutti i tipi di file è possibile utilizzare le funzioni che erano inizialmente disponibili solo per codice gestito. Ad esempio, è ora possibile utilizzare ASP.NET autenticazione basata su form e Uniform Resource Locator (URL) l'autorizzazione per i file statici, Active Server Pages (ASP), i file e tutti gli altri tipi di file nei siti e applicazioni. In secondo luogo, questa soluzione elimina la duplicazione di funzioni diverse in IIS e ASP.NET. Per esempio, quando un client richiede un file gestito, il server richiama il modulo di autenticazione appropriato nella pipeline integrata per autenticare il client. Nelle versioni precedenti di IIS, questa stessa richiesta potrebbe passare attraverso un processo di

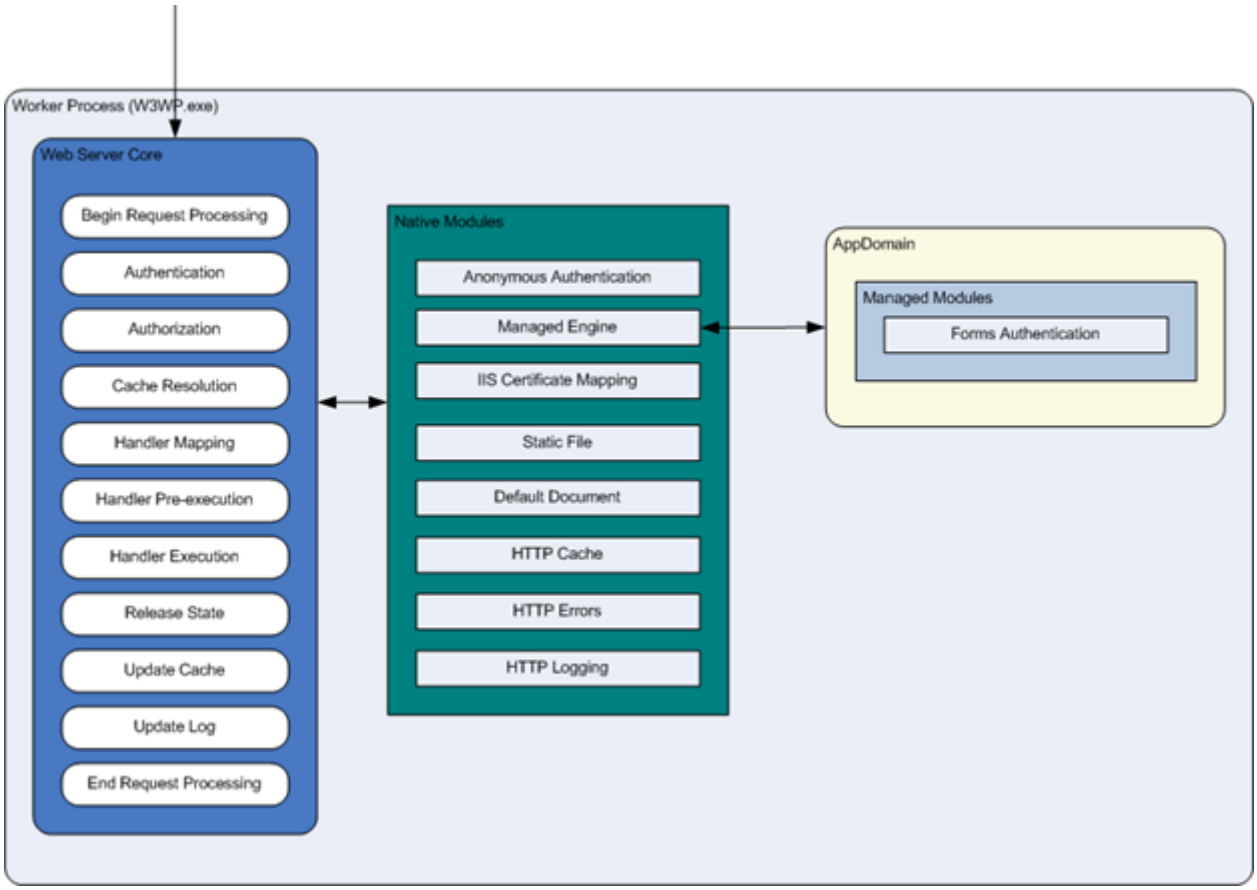
autenticazione sia la pipeline di IIS e nella pipeline ASP.NET. Infine è possibile gestire tutti i moduli in un unico luogo, invece di gestire alcune funzionalità di IIS e alcuni nella configurazione di ASP.NET. Questo semplifica la gestione di siti e applicazioni sul server. I pool di applicazioni, vere e proprie applicazioni separate per i limiti del processo per evitare una richiesta di influenzare un'altra applicazione sul server, inoltre, è ora possibile specificare un'impostazione che determina la modalità di elaborare le richieste che coinvolgono le risorse gestite: la modalità integrata o in modalità classica. Quando un pool di applicazioni è in modalità integrata, è possibile usufruire del sistema integrato di elaborazione delle richieste architettura di IIS e ASP.NET. Quando un processo di lavoro in un pool di applicazioni riceve una richiesta, la richiesta passa attraverso un elenco ordinato di eventi. Ogni evento chiama i moduli necessari nativi e gestiti per elaborare porzioni di richiesta e di generare la risposta. Ci sono diversi vantaggi per pool di applicazioni in esecuzione in modalità integrata. In primo luogo i modelli di elaborazione delle richieste di IIS e ASP.NET sono integrati in un modello di processo unificato. Questo modello elimina i passaggi che sono stati precedentemente duplicati in IIS e ASP.NET, ad esempio l'autenticazione. Inoltre, la modalità integrata consente la disponibilità delle funzioni gestite a tutti i tipi di contenuto. Quando un pool di applicazioni sia in modalità classica, IIS 7 gestisce le richieste allo stesso modo come in IIS 6.0 modalità di isolamento processo di lavoro. Richieste ASP.NET prima passare attraverso fasi di lavorazione nativi in IIS e sono quindi indirizzati a Aspnet˙isapi.dll per l'elaborazione di codice gestito nel runtime gestito. Infine, la richiesta viene rimandato IIS per inviare la risposta. Questa separazione dei modelli di elaborazione delle richieste di IIS e ASP.NET risulta una duplicazione di alcune fasi di lavorazione, come l'autenticazione e l'autorizzazione. Inoltre, le caratteristiche di codice gestito, ad esempio l'autenticazione Forms, sono disponibili solo per le applicazioni ASP.NET o applicazioni per cui si dispone di script mappato tutte le richieste vengono gestite da aspnet˙isapi.dll. Le seguenti immagini forniscono una panoramica di una richiesta HTTP.

Il seguente elenco descrive l'elaborazione delle richieste di flusso che è mostrato in figura:

1. Un browser client avvia una richiesta HTTP per una risorsa sul server Web, HTTP.sys intercetta la richiesta;
2. HTTP.sys ha il compito di ottenere informazioni dal archivio di configurazione;
3. WAS reperisce le informazioni dall'archivio di configurazione, ApplicationHost.config;
4. Il servizio WWW riceve le informazioni di configurazione, come pool di applicazioni e di configurazione del sito;
5. Il servizio WWW utilizza le informazioni di configurazione per configurare HTTP.sys;
6. WAS avvia un processo di lavoro del pool di applicazioni per cui è stata presentata la richiesta;
7. Il processo di lavoro elabora la richiesta e restituisce una risposta al HTTP.sys;
8. Il client riceve una risposta;



Nel dettaglio di un worker process, una richiesta HTTP passa attraverso una serie di fasi ordinate, eventi chiamati, in Core Web Server. In ogni caso, un modulo nativo elabora parte della richiesta, come ad esempio l'autenticazione dell'utente o l'aggiunta di informazioni nel registro eventi. Se una richiesta richiede un modulo gestito, il modulo nativo ManagedEngine crea un dominio di applicazione, in cui il modulo di gestione in grado di eseguire le elaborazioni necessarie, come ad esempio l'autenticazione di un utente con autenticazione basata su form. Quando la richiesta passa attraverso tutti gli eventi del Web Server Core, la risposta viene restituita al HTTP.sys. Nella figura di seguito, mostra una richiesta HTTP di entrare nel processo di lavoro.



In particolare GeoWeb5 essendo full managed, integrato nel framework .Net esegue ordinatamente il flusso passando dai Web Server Core, utilizzando strettamente i moduli nativi utili per la creazione dell'AppDomain managed. All'interno di questo dominio utilizziamo direttamente i moduli managed messi a disposizione dal framework, si vedrà un semplice esempio dell'utilizzo del modulo di Cache nel capitolo Progetto e Realizzazione.

Server Side - Data Layer

In questo paragrafo si parlerà della persistenza e dell'accesso ai dati di business, che nel progetto sono rappresentati da un database relazionale al cui interno sono memorizzate le informazioni dei clienti nella tabella "`romagna_customers`" e le soluzioni al nostro problema di ottimizzazione memorizzati nella tabella "`sol`". La particolarità di queste entità tabellari consiste nella loro struttura, infatti, entrambe le tabelle contengono particolari tipi dato che sono una estensione dei normali tipi dato di un database relazionale. Per questo motivo parliamo di Postgis e non di PostgreSQL, Postgis è l'estensione a PostgreSQL che ci permette di avere una serie di estensioni per il GIS e di utilizzarle nelle tabelle precedentemente citate, in particolare, nella tabella dei clienti ogni cliente possiede le sue coordinate geografiche; le soluzioni sono una struttura complessa definita solo in Postgis, stiamo parlando del tipo Geometry che può avere diverse declinazioni, Point per i clienti e MultiLineString per le soluzioni. Inoltre, queste estensioni forniscono StoredProcedure, Function e altre funzioni dedicate al tema GIS, queste estensioni permettono ad esempio l'integrazione con GeoServer. Anche Oracle nella sua versione Enterprise fornisce un Add-In per queste funzionalità mentre SqlServer sembra non disporre ancora di una soluzione integrata ma punti ad utilizzare un progetto esterno ArcGis.

Postgis Server

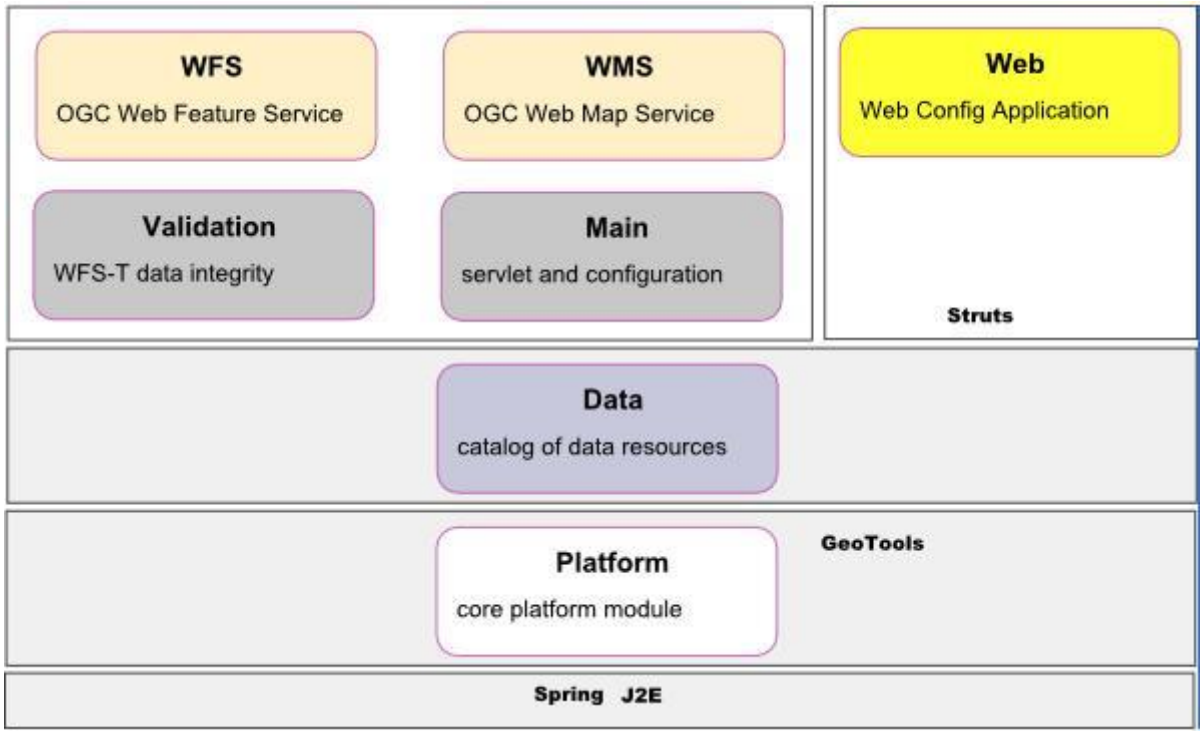
Essendo un'estensione a PostgreSQL preserva tutte le caratteristiche di un normale database relazionale, autenticazione, autorizzazione e modalità di accesso. In particolare nel progetto sono state utilizzate nel DataLayerProvider normali istruzioni SQL arricchite dalle estensioni GIS, se ne potrà vedere ampliamente l'utilizzo nel prossimo capitolo "Progettazione e Realizzazione".
Essendo comunque PostgreSQL l'engine nel progetto sono stati utilizzati in normali driver ADO.NET per la connessione, l'interrogazione e l'aggiornamento alla base dati, in particolare Postgres fornisce come driver nativo sul Framework 4:

```
Npgsql2.0.12.0-bin-ms.net4.0\Npgsql2.0.12.0-bin-ms.net4.0\Npgsql.dll
```

Geo Server

GeoServer è un server open source scritto in Java che permette agli utenti di condividere e modificare i dati geo spaziali. Progettato per l'interoperabilità, che pubblica i dati da qualsiasi fonte importante di dati spaziali utilizzando standard aperti. Essendo un progetto guidato dalla comunità, GeoServer è sviluppato, testato e supportato da un gruppo eterogeneo di individui e organizzazioni di tutto il mondo. GeoServer è l'implementazione di riferimento di Open Geospatial Consortium (OGC), servizio Web Feature Service (WFS) e Web Coverage Service (WCS), così come un alto rendimento certificato e conforme Web Map Service (WMS). GeoServer è stato avviato nel 2001 da The Open Planning Project (TOPP), un'organizzazione non-profit incubatore tecnologico con sede a New York. TOPP è stata la creazione di una serie di strumenti per consentire democrazia aperta e per contribuire a rendere il governo più trasparente. Il primo di questi è stato GeoServer, che è uscito di un riconoscimento che una suite di strumenti per consentire il coinvolgimento dei cittadini nel governo e nella pianificazione urbana sarebbe notevolmente rafforzata dalla possibilità di condividere i dati territoriali. I fondatori GeoServer immaginato un Web geo spaziale, analogo al World Wide Web. Con il World Wide Web, è possibile cercare e scaricare il testo. Con il Web geo spaziale, si può cercare e scaricare i dati territoriali. I provider di dati sarebbe in grado di pubblicare i propri dati direttamente a questo sito, e gli utenti possono accedere direttamente, in contrasto con i metodi indiretti e ingombranti ora di condividere i dati che esistono oggi. Coloro che sono coinvolti con GeoServer fondato il progetto GeoTools, un sistema open source GIS Toolkit Java. Attraverso GeoTools, il supporto per Shapefile, database Oracle, ArcSDE integrazione, e molto di più è stato aggiunto. Intorno allo stesso tempo, come GeoServer è stata fondata, il Consorzio OpenGIS (ora Open Geospatial Consortium), stava lavorando sullo standard Web Service Feature. Si specifica un protocollo per rendere i dati territoriali direttamente sul web, utilizzando GML (Geographic Markup Language), un formato di dati interoperabile. È stato anche creato, un protocollo per la creazione e la visualizzazione di immagini di mappe create dai dati spaziali. Altri progetti si interconnessi. La ricerca Rifrazioni creato Postgis, un database gratuito e aperto spaziale, che ha permesso GeoServer per la connessione a un database gratuito. Inoltre, MetaCarta creato OpenLayers, un open source basata

su browser programma di utilità di visualizzazione mappa. Insieme, questi strumenti sono tutti hanno migliorato le funzionalità di GeoServer. GeoServer possono ora inviare i dati a molti altri visualizzatori di dati territoriali, come ad esempio Google Earth, un popolare mondo virtuale tridimensionale. Inoltre, GeoServer sta attualmente lavorando direttamente con Google al fine di consentire i dati GeoServer sia consultabile su Google Maps. Ben presto la ricerca di dati spaziali sarà facile come una ricerca su Google per una pagina web. Così GeoServer sta continuando la sua missione di rendere i dati spaziali più accessibile a tutti.



Per queste caratteristiche di estendibilità , standardizzazione, si è deciso di utilizzarlo come metodo alternativo all'accesso direttamente ai dati.

Progetto e Realizzazione

In questo capitolo si vedrà l'applicazione pratica all'interno del progetto delle tecnologie precedentemente citate, in particolare si ripercorrerà lo stack multi tier partendo dal Data Layer risalendo fino al Presentation Layer. Nella prima parte del capitolo si spiegheranno le motivazioni architetturali e i design pattern utilizzati, successivamente vedremo l'installazione e la configurazione per poi entrare definitivamente nell'implementazione.

Progettazione

Scopo del progetto

Lo scopo di questo progetto è quello di creare un layer di front-end cartografico interattivo per la gestione di percorsi; ottenuti come risultato da un processo di ottimizzazione, in particolare, il progetto deve essere in grado di mostrare questi percorsi tramite un browser collegato ad internet.
Questo progetto ha due scopi finali, tramite un normale browser desktop l'applicativo deve essere in grado di alterare le soluzioni precedentemente calcolate, cioè, deve essere in grado di eliminare, aggiungere e spostare punti da una soluzione. Inoltre l'applicativo deve poter visualizzare gli stessi risultati anche su dispositivi mobili, cioè , deve modificare il suo aspetto in autonomia riconoscendo il dispositivo che gli sta richiedendo informazioni.
Le due visualizzazioni avranno funzionalità diverse, in particolare la versione mobile è pensata per la sola visualizzazione, sono quindi state rimosse le funzionalità amministrative delle soluzioni, in un generico user case si potrebbe dire che la versione desktop è stata progettata per l'utilizzo da parte di persone che organizzano le consegne, mentre la versione mobile è utilizzata dai vettori incaricati di eseguire le consegne.

Architettura SOA

Una Service Oriented Architecture (SOA, Architettura Orientata ai Servizi) è un modello architetturale per la creazione di sistemi residenti su una rete che focalizza l'attenzione sul concetto di servizio. Un sistema costruito seguendo la filosofia SOA è costituito da servizi indipendenti tra loro, che risiedono su più computer all'interno di una rete. Ogni servizio mette a disposizione una certa funzionalità e può utilizzare quelle che gli altri servizi hanno reso disponibili, realizzando, in questo modo, applicazioni di maggiore complessità. Un Distributed System consiste di vari agenti software distinti che devono lavorare insieme per svolgere alcuni compiti. Inoltre, gli agenti in un sistema distribuito non operano nello stesso ambiente di calcolo, quindi devono comunicare per mezzo di stack di protocolli

hardware/software su una rete; nel nostro caso si sono utilizzati esclusivamente protocolli software basati su http, in particolare abbiamo utilizzato SOAP per utilizzare i servizi di GeoServer WFS, abbiamo utilizzato JSON esposto dal ApiController di MVC4 per comunicare tra il JavaScript e il Middle Tier. Ciò ha importanti implicazioni architetturali perché i sistemi distribuiti richiedono che gli sviluppatori (di infrastruttura e applicazioni) considerino la latenza, fattore imprevedibile dell'accesso remoto, e tengano presente questioni relative alla concorrenza e la possibilità di fallimenti parziali. L'astrazione delle SOA non è legata ad alcuna specifica tecnologia, ma semplicemente definisce alcune proprietà, orientate al riutilizzo e all'integrazione in un ambiente eterogeneo, che devono essere rispettate dai servizi che compongono il sistema. Un servizio deve poter essere ricercato in base alla sua interfaccia e richiamato a tempo di esecuzione. La definizione del servizio in base alla sua interfaccia rende quest'ultima indipendente dal modo in cui è stato realizzato il componente che lo implementa. La caratteristica fondamentale è che ogni servizio deve essere ben definito, completo e soprattutto indipendente dal contesto o dallo stato di altri servizi; questo è il punto focale di un servizio, deve fornire risposte ad una domanda senza sapere chi lo invoca e perché, per questo motivo deve essere definito in termini di ciò che fa, astraendo dai metodi e dalle tecnologie utilizzate per implementarlo. Questo determina l'indipendenza del servizio non solo dal linguaggio di programmazione utilizzato per realizzare il componente che lo implementa ma anche dalla piattaforma e dal sistema operativo su cui è in esecuzione, ad esempio la parte server di data storage, PostgreSQL/Postgis, la parte di data service, GeoServer, possono essere installate su qualsiasi sistema operativo, GeoServer infatti essendo scritto in Java ed esponendo interfacce SOAP può essere, ed è stato, usufruito da sistemi operativi Microsoft. Un'architettura è debolmente accoppiata se le dipendenze fra le sue componenti sono in numero limitato. Questo rende il sistema flessibile e facilmente modificabile. Questo è effettivamente il punto debole di questa architettura, la dipendenza tra i servizi, nella pratica è il maggior sforzo che un progettista deve fare è progettare un'architettura SOA parlando di domini cercando di evitare maggiormente le dipendenze e quindi cercare di definire una linea di confine molto forte tra i domini dell'intera applicazione.

Inoltre in un'architettura SOA il servizio deve essere reso disponibile sulla rete attraverso la pubblicazione della sua interfaccia (in un Service Directory o Service Registry) ed accessibile in modo trasparente rispetto alla sua allocazione. Essere disponibile sulla rete lo rende accessibile da quei componenti che ne richiedono l'utilizzo e l'accesso deve avvenire in maniera indipendente rispetto all'allocazione del servizio. La pubblicazione dell'interfaccia deve rendere noto anche le modalità di accesso al servizio. Deve mettere a disposizione un basso numero di operazioni, cioè poche funzionalità, in modo tale da non dover avere un programma di controllo complesso. Deve essere invece orientato ad un elevato livello di interazione con gli altri servizi attraverso lo scambio di messaggi. Per questo motivo e per il fatto che i servizi possono trovarsi su sistemi operativi e piattaforme diverse è necessario che i messaggi siano composti utilizzando un formato standard largamente riconosciuto (Platform Neutral). I dati che vengono trasmessi attraverso i messaggi possono essere costituiti sia dal risultato dell'elaborazione di un certo servizio sia da informazioni che più servizi si scambiano per coordinarsi fra loro. Nell'architettura SOA le applicazioni sono il risultato della composizione di più servizi. E per questo motivo che ogni servizio deve essere indipendente da qualsiasi altro, in modo tale da ottenere il massimo della riusabilità. La creazione di applicazioni o di servizi più complessi attraverso la composizione dei servizi di base viene definita Service Orchestration. Il Service Provider è un'entità che mette a disposizione un qualche servizio. Tale servizio, per poter essere trovato da altre entità che vogliono utilizzarlo, deve essere reso visibile sulla rete, in termine tecnico Pubblicato. A tal fine il Service Provider comunica al Service Registry le informazioni relative al servizio, perché vengano memorizzate. Il Service Registry possiede quindi le informazioni, come URL e modalità di accesso, di tutti i servizi disponibili. Nel momento in cui un Service Consumer dovrà utilizzare un servizio farà richiesta delle informazioni ad esso relative al Service Registry. Con queste informazioni il Service Consumer potrà comunicare direttamente con il Service Provider ed utilizzare il servizio.

Design Pattern

Il concetto di pattern scaturisce dal lavoro di un architetto edile, Christopher Alexander, che alla fine degli anni '70 fu il primo a proporre l'idea di usare i pattern, ovvero soluzioni collaudate a problematiche note, per progettare l'architettura di palazzi ed edifici in genere. Successivamente, nel corso degli anni '80 e nei successivi anni '90, l'argomento fu ripreso e sviluppato dagli esperti informatici del tempo nell'ottica della progettazione object-oriented, fino alla definitiva consacrazione nel 1995 coincidente con l'uscita del libro Design Patterns: Elements of Reusable Object-Oriented Software di Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, che rappresenta ancor oggi una delle pietre miliari della letteratura dedicata alla progettazione di sistemi basati sul paradigma ad oggetti. Negli ultimi anni molti autori hanno contribuito con ulteriori pubblicazioni dedicate al tema dei design pattern, fornendo nuove interpretazioni e ulteriori spunti di riflessione in merito al loro utilizzo. Possiamo definire pattern, sono orientati a risolvere particolari problematiche di progettazione e tendono ad introdurre nell'ambito di una struttura ad oggetti quella flessibilità che è necessaria per rendere il codice riutilizzabile ed estendibile. Non tutti i pattern sono uguali ed è bene capire come essi possono essere strutturati e organizzati. Dal momento che esistono diverse tipologie di pattern in funzione della loro area di applicazione, in generale essi possono essere raggruppati in cluster, ciascuna delle quali contenente pattern orientati a risolvere problematiche similari. Nell'ambito del cluster dei pattern relativi allo sviluppo di applicazioni software possiamo individuare tre categorie di pattern caratterizzate da un diverso livello di astrazione.

I Pattern architetturali: descrivono lo schema organizzativo della struttura che caratterizza un sistema software. In genere questi pattern individuano le parti del sistema a cui sono associate responsabilità omogenee e le relazioni che esistono tra i diversi sottosistemi. Pattern di disegno (*design pattern*): sono i pattern che si riferiscono alle problematiche legate al disegno object-oriented e di essi avremo modo di parlare in modo più approfondito. Pattern di implementazione (*idiomi*): sono pattern di basso livello specifici per una particolare tecnologia (per esempio, il .NET Framework). Essi descrivono le modalità implementative da utilizzare per risolvere problematiche di sviluppo sfruttando in modo mirato le caratteristiche peculiari di una particolare piattaforma. Come detto, ciascuno di questi gruppi è caratterizzato da un grado di astrazione differente. I design pattern si collocano tra

i pattern architetturali, troppo generici per essere orientati a risolvere problematiche di disegno, e i pattern idiomatici, molto legati alla tecnologia a cui si riferiscono e all'implementazione vera e propria. I design pattern descrivono soluzioni che lasciano sempre e comunque un certo grado di libertà nella loro adozione e implementazione, dal momento che non descrivono mai soluzioni che sono valide per una piattaforma specifica, ma al contrario hanno una validità più generale e trasversale rispetto alla tecnologia. Uno degli aspetti più delicati nel disegno object-oriented (OOD) consiste nella scomposizione del sistema in oggetti. Si tratta di una attività complessa dal momento che entrano in gioco fattori non direttamente collegati alle specifiche funzionali quali l'accoppiamento tra oggetti, la loro dipendenza, la coesione funzionale, la granularità, la flessibilità, l'estendibilità e la riusabilità. Questi aspetti devono necessariamente influenzare il processo di scomposizione, talvolta anche in modi tra loro discordanti.

Esistono diversi approcci che permettono di scomporre in oggetti un sistema. È possibile partire dai casi d'uso, individuare in essi i sostantivi e i verbi e da questi ricavare le classi e i metodi corrispondenti al fine di ottenere la struttura ad oggetti desiderata. In alternativa, è possibile porre l'attenzione principalmente sulle responsabilità e sulle collaborazioni nell'ambito del sistema in fase di studio e, in funzione di esse, individuare gli oggetti necessari per gestirle opportunamente. O ancora, è possibile partire da un modello del mondo reale e tradurre gli elementi individuati in altrettanti oggetti. Ognuno di questi approcci concorre a definire la struttura statica del sistema che, se da un lato rispecchia la realtà di oggi, dall'altro in genere non si presta direttamente ad evolvere nel tempo e ad adattarsi alla realtà di domani. L'esigenza di individuare una struttura flessibile ed estendibile, in grado di rispondere al meglio ai cambiamenti nel tempo, porta inevitabilmente a cercare soluzioni di disegno che facciano largo uso di astrazioni e oggetti non necessariamente collegati alla realtà in esame per non limitare in modo importante l'evoluzione del sistema. I design pattern aiutano ad individuare queste astrazioni e gli oggetti in grado di rappresentarle. Essi concorrono a limitare le dipendenze, incrementando l'estendibilità e la riusabilità, e rendono di conseguenza il sistema più mantenibile nel tempo, in grado di rispondere ai cambiamenti in modo efficiente ed elegante. I design pattern agevolano il riuso di soluzioni architetturali note, rendendo accessibili agli architetti e agli sviluppatori tecniche di disegno universalmente riconosciute come valide ed efficaci. In questo senso i design pattern

aiutano i progettisti a operare scelte consapevoli tra le varie alternative possibili allo scopo di favorire la riusabilità.
Possiamo suddividere pattern organizzandoli in tre categorie distinte e tra loro complementari, pattern creazionali, che riguardano la creazione di istanze; pattern strutturali, che si riferiscono alla composizione di classi e oggetti; pattern comportamentali, che si occupano delle modalità con cui classi e oggetti interagiscono tra loro in relazione alle loro diverse responsabilità.

I pattern creazionali sono cinque:

1. Abstract Factory: fornisce un'interfaccia per creare famiglie di oggetti correlati o dipendenti senza specificare le classi concrete; in particolare questo pattern è stato utilizzato per astrarre le sorgenti dato, siano essi dati presi da PostgreSQL tramite SQL o presi tramite WFS di GeoServer o ancora la gestione dei tile, presi da File System (mappe personalizzate) o presi via protocollo http, come ad esempio per i connettori a openstreetmap.org
2. Builder: separa la costruzione di un oggetto complesso dalla sua rappresentazione, in modo tale che lo stesso processo di costruzione possa creare rappresentazioni differenti;
3. Factory Method: definisce un'interfaccia per creare un oggetto, ma lascia alle classi derivate di decidere quale classe istanziare. Questo pattern permette a una classe di delegare la creazione di un'istanza alle sue classi derivate;
4. Prototype: specifica il tipo degli oggetti da creare usando un'istanza prototipale e crea i nuovi oggetti a partire da questo prototipo;
5. Singleton: assicura che una classe abbia solamente un'unica istanza e fornisce un entry-point globale ad essa.

I pattern strutturali sono sette:

1. Adapter: converte l'interfaccia di una classe in un'altra interfaccia compatibile con il client. Questo pattern consente a classi diverse di collaborare tra loro, cosa che non sarebbe possibile diversamente a causa della incompatibilità delle rispettive interfacce;
2. Bridge: disaccoppia un'astrazione dalla sua implementazione affinché entrambe possano variare in modo indipendente;

3. Composite: compone una serie di oggetti in una struttura ad albero secondo una gerarchia di tipo *part-whole* (parte-totalità). Questo pattern permette ai client di trattare oggetti singoli o loro raggruppamenti in modo uniforme;
4. Decorator: aggiunge dinamicamente responsabilità addizionali ad un oggetto. Questo pattern fornisce un meccanismo alternativo e flessibile all'ereditarietà per estendere le funzionalità base;
5. Facade: fornisce un'interfaccia unificata a un insieme di interfacce in un sottosistema. Questo pattern definisce un'interfaccia ad un livello più alto che rende il sottosistema più facile da usare, dato che ne maschera la complessità interna;
6. Flyweight: usa la condivisione per gestire in modo efficiente un numero considerevole di oggetti a granularità fine;
7. Proxy: fornisce un surrogato di un oggetto per controllare l'accesso ad esso.

I pattern comportamentali sono undici:

1. Chain of Responsability: evita di accoppiare il mittente di una richiesta con il suo destinatario dando la possibilità a più di un oggetto di gestire la richiesta. Collega tra loro gli oggetti ricevitori e fa passare la richiesta da un oggetto all'altro fino a destinazione;
2. Command: incapsula una richiesta in un oggetto, rendendo possibile parametrizzare i client con diverse tipologie di richieste, con richieste bufferizzate (*queue*), con richieste registrate (*log*) e con richieste annullabili (*undo*);
3. Interpreter: dato un linguaggio, definisce una rappresentazione della sua grammatica e del relativo interprete, che usa la rappresentazione per interpretare le frasi del linguaggio;
4. Iterator: fornisce un modo per accedere in modo sequenziale agli elementi di una collezione di oggetti senza esporre la sua rappresentazione sottostante;
5. Mediator: definisce un oggetto che incapsula le modalità di interazione di un insieme di oggetti. Questo pattern favorisce un basso accoppiamento, evitando che gli oggetti facciano riferimento l'uno con l'altro esplicitamente, e permette di variare le modalità di interazione in modo indipendente dagli oggetti stessi;

6. Memento: senza violare l'incapsulamento, recupera e rende esplicito lo stato interno di un oggetto in modo tale che l'oggetto stesso possa essere riportato allo stato originale in un secondo momento;
7. Observer: definisce una dipendenza uno-a-molti fra oggetti in modo tale che, se un oggetto cambia stato, tutti gli oggetti da esso dipendenti vengono notificati e aggiornati automaticamente;
8. State: permette ad un oggetto di modificare il suo comportamento quando il suo stato interno cambia;
9. Strategy: definisce una famiglia di algoritmi, li incapsula e li rende intercambiabili fra loro. Questo pattern permette di variare gli algoritmi in modo indipendente dal contesto di utilizzo;
10. Template Method: definisce lo scheletro di un algoritmo in un metodo di una classe base, delegando alcuni passi alle classi derivate. Questo pattern permette di ridefinire nelle classi derivate alcuni passi dell'algoritmo senza cambiare la struttura dell'algoritmo stesso;
11. Visitor: rappresenta un'operazione da svolgersi sugli elementi di una struttura ad oggetti. Questo pattern consente di definire nuove operazioni senza cambiare le classi degli elementi su cui opera.

Nel corso del progetto si vedrà l'applicazione di questi pattern, a volte vengono utilizzati inconsciamente, purtroppo, spesso l'applicativo viene forzato nell'utilizzo generando sovrastrutture; per lo sviluppo e il disegno architetturale si è cercato di utilizzare un approccio KISS (Keep It Simple, Stupid) coniato da U.S. Navy nel 1960 dall'ingegnere aereospaziale Kelly Johnson; si sono inoltre adottate metodologie di sviluppo Agile.

Installazione

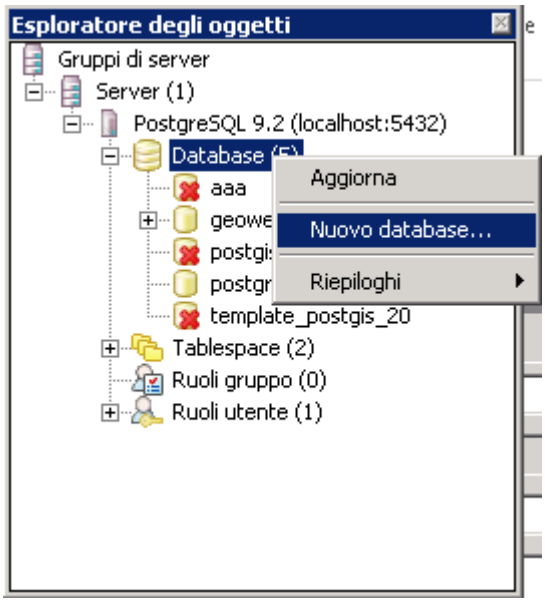
In questo capitolo si preparerà l'ambiente per l'esecuzione dell'applicativo, in particolare ci si concentrerà su quelle parti di installazione che influenzano la configurazione. Innanzitutto si dovrà preparare l'infrastruttura, si deve quindi procurare ed installare i server, in particolare l'infrastruttura necessita di:

- Un server con sistema operativo Windows (7/Server 2008R2) con Internet Information Services, Microsoft .Net Framework 4.5;
- PostgresSQL 9.2 con estensioni Postgis 2.0;
- GeoServer 2.2.3;

Ora si procederà con l'installazione e successivamente con la configurazione dell'applicativo; si partirà dalla base, lo strato dati, risalendo verso GeoServer per l'interfaccia dei dati ed infine installeremo l'applicativo in Internet Information Services.

Ambiente dei dati Postgres

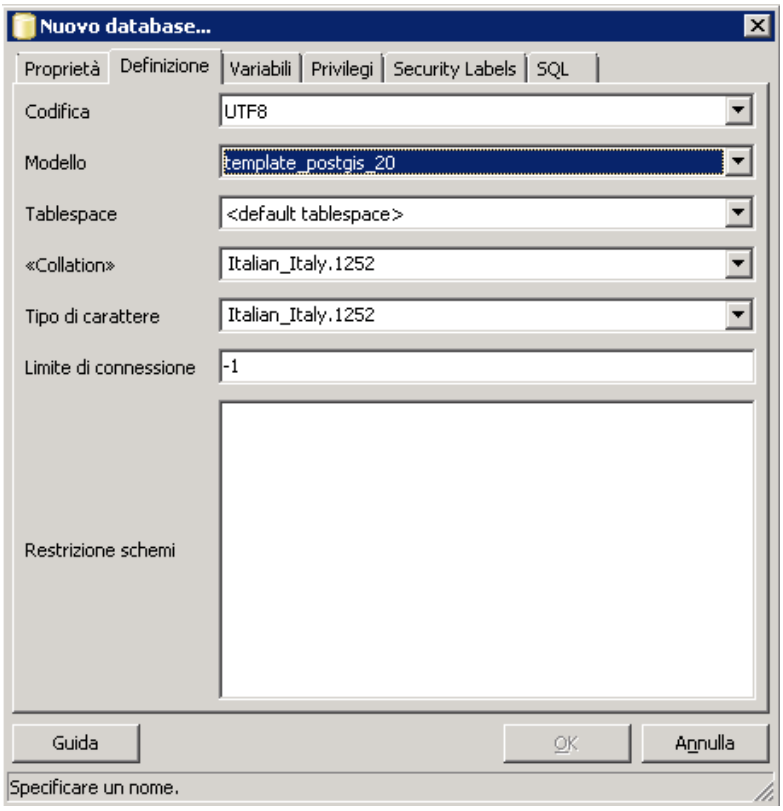
Dopo esserci assicurati di aver installato correttamente l'engine SQL e le estensioni GIS procediamo al primo step, la creazione del database con estensioni, in particolare dall'interfaccia di amministrazione pgAdmin III



Dopo essersi collegati con l'utente amministratore oppure con un utente avente i privilegi di CREATE DATABASE da interfaccia di amministrazione sull'icona Database fare tasto destro e selezionare "Nuovo database…"

A questo punto si aprirà un'interfaccia e ci si deve assicurare di creare il database partendo da un template contenente le estensioni GIS. Come si può osservare dalla figura sottostante nel tab Modello va utilizzato un template specifico che viene importato al momento dell'installazione di Postgis, in questo setup si chiama `template_postgis_20`. Nel tab precedente "Proprietà" si deve configurare il nome del database e il proprietario (db owner). Nel nostro applicativo il nome del database sarà `geoweb5`, ma il nome può essere scelto a piacimento. E' necessario ricordarsi tutte queste informazioni serviranno successivamente per la configurazione di GeoServer e dell'applicativo stesso, in particolare dobbiamo ricordare:

1. Server name ed eventualmente il nome dell'istanza PostgreSQL;
2. La porta tcp del listener di Postgres (default 5432);
3. Le credenziali dell'utente owner del db (postgres/postgres);
4. Il nome del database (geoweb5);



Una volta creato il database si procederà alla creazione dello schema; schema necessario all'applicativo per recuperare le informazioni:

1. Tabella `romagna_customers` in cui saranno memorizzate le caratteristiche dei clienti;
2. Tabella `sol` in cui saranno memorizzate le soluzioni al problema di ottimizzazione;

Utilizzando lo strumento per eseguire Query, eseguiamo queste istruzioni DML che permettono la creazione delle due tabelle.

```
CREATE TABLE romagna_customers
(
  gid serial NOT NULL,
  pk_uid double precision,
  id double precision,
  idnodo double precision,
  xcoord numeric,
  ycoord numeric,
  demand double precision,
  ready_time double precision,
  due_date double precision,
  service_ti double precision,
  zona double precision,
```

```
  freq double precision,
  geometry geometry(Point,4326),
  CONSTRAINT romagna_customers_pkey PRIMARY KEY (gid)
);

CREATE TABLE sol
(
  gid serial NOT NULL,
  idroute double precision,
  geom geometry(MultiLineString,4326),
  color text,
  CONSTRAINT sol_pkey PRIMARY KEY (gid)
);
```

Suscita particolare interesse il tipo dato `geometry` che rappresenta l'estensione Postgis per la rappresentazione di dati geospaziali pienamente compatibili con GeoServer. Dopo l'inserimento di informazioni all'interno di queste tabelle possiamo procedere con la configurazione di GeoServer.

Ambiente dei dati GeoServer

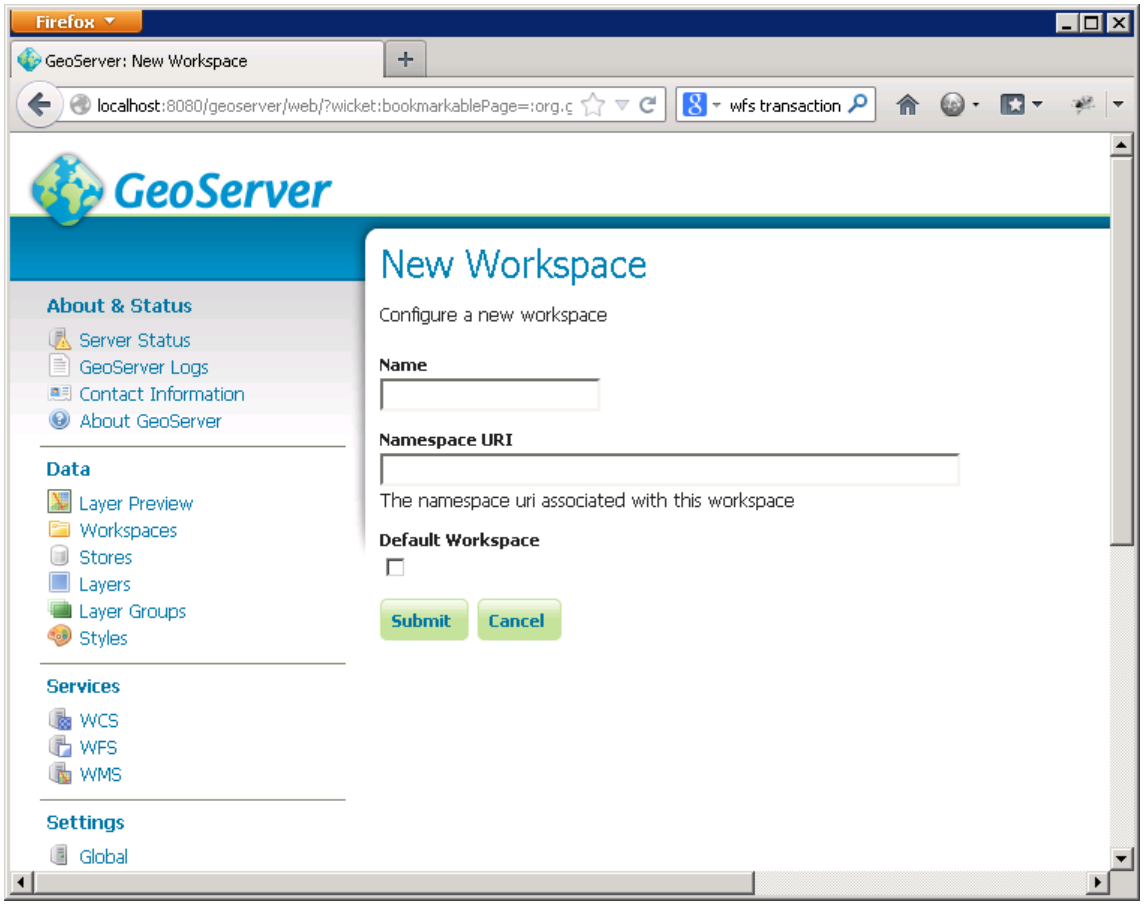
Si è quindi giunti alla configurazione di una parte cruciale dell'architettura, il middle-tier dei dati che ci permette l'astrazione dalla sorgente dati. Dopo aver installato l'applicativo si deve configurare il server in modo da poter utilizzare i servizi WFS, vedremo nei prossimi capitoli la realizzazione tramite esempi di chiamate.

Questa parte di configurazione si articola in diverse fasi:

1. Accedere all'interfaccia di amministrazione web di GeoServer, tramite installazione standard l'interfaccia è reperibile dal server all'url http://localhost:8080/geoserver/web/ ed è possibile accedere con le credenziali impostate nella fase di setup;
2. Creare un nostro workspace (`geoweb5`);
3. Creare un nostro datastore (`dsgeoweb5`);
4. Pubblicare i nostri layer (`romagna_customers, sol`);

Creazione del workspace

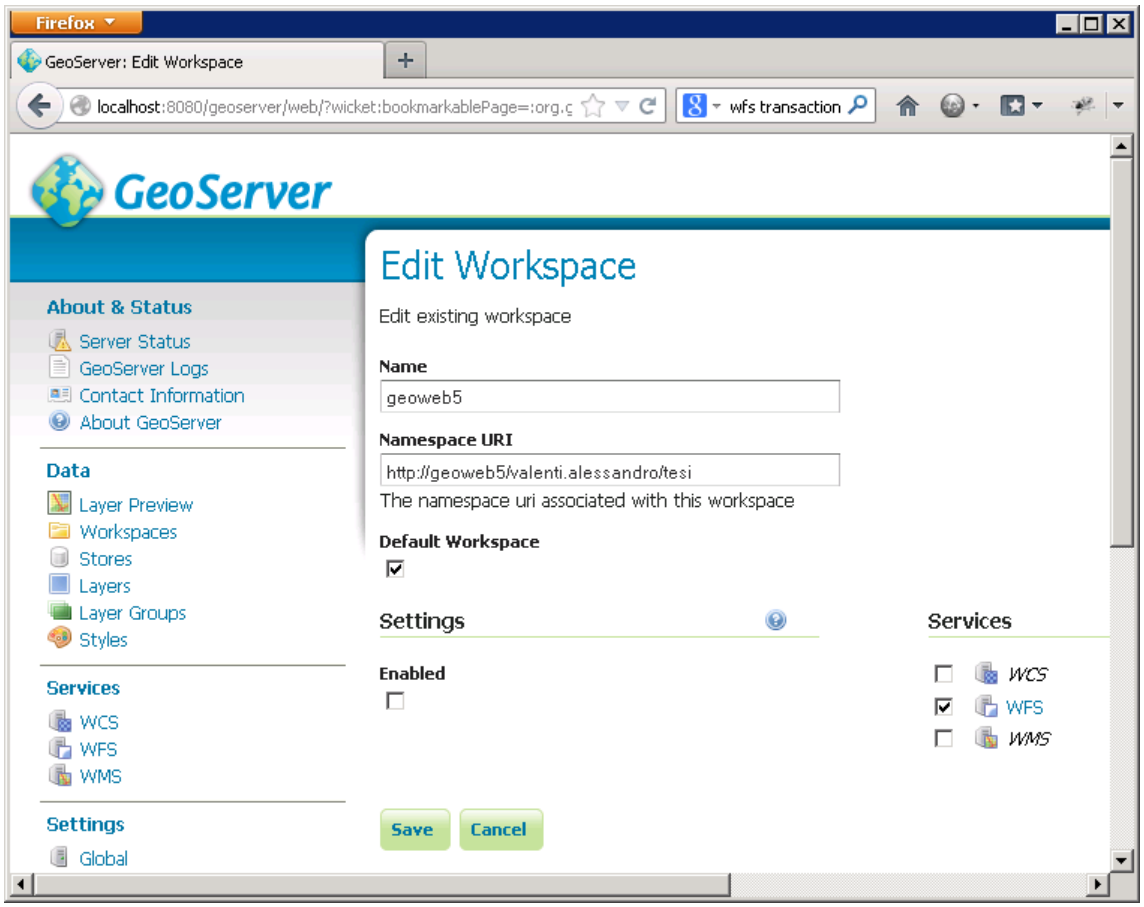
Questa è la parte fondamentale per poter utilizzare correttamente GeoServer WFS in quanto in questa schermata definiamo il namespace URI degli oggetti su cui vogliamo operare via WFS, quindi dopo aver cliccato sull'icona Workspaces ed in altro avendo selezionato "Add New Workspace" apparirà



Inseriamo i dati che ci serviranno dall'applicativo

A questo punto dopo aver salvato rientriamo in editing e selezioniamo per questo workspace l'abilitazione a WFS

Il risultato finale sarà un Workspace con queste caratteristiche



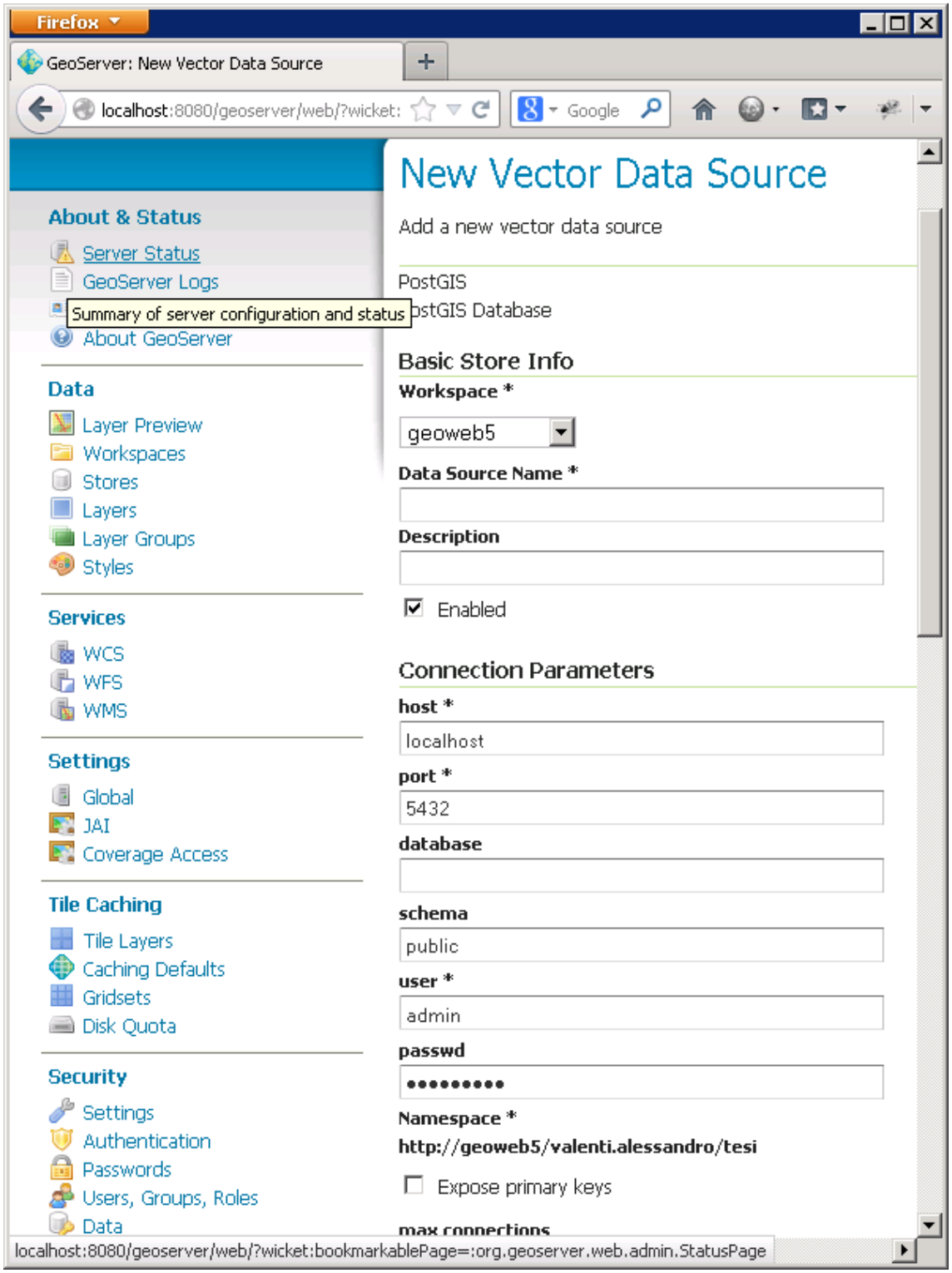
Da questa procedure dobbiamo memorizzarci il Namespace URI che nel nostro caso è `http://geoweb5/valenti.alessandro/tesi`

Creazione dello store dati

A questo punto si è creato il nostro ambiente compatibile con WFS, ora si devono creare gli oggetti; partendo dalla sorgente dati, si dovrà cliccare sul pulsante Stores "New Data Store" e scegliendo come interfaccia "Postgis" per il collegamento al database; si dovrà fare attenzione ad utilizzare il Workspace precedentemente creato (`geoweb5` scegliendolo dalla tendina) editando il nome in "Data Source Name" nel nostro caso `dsgeoweb5`.

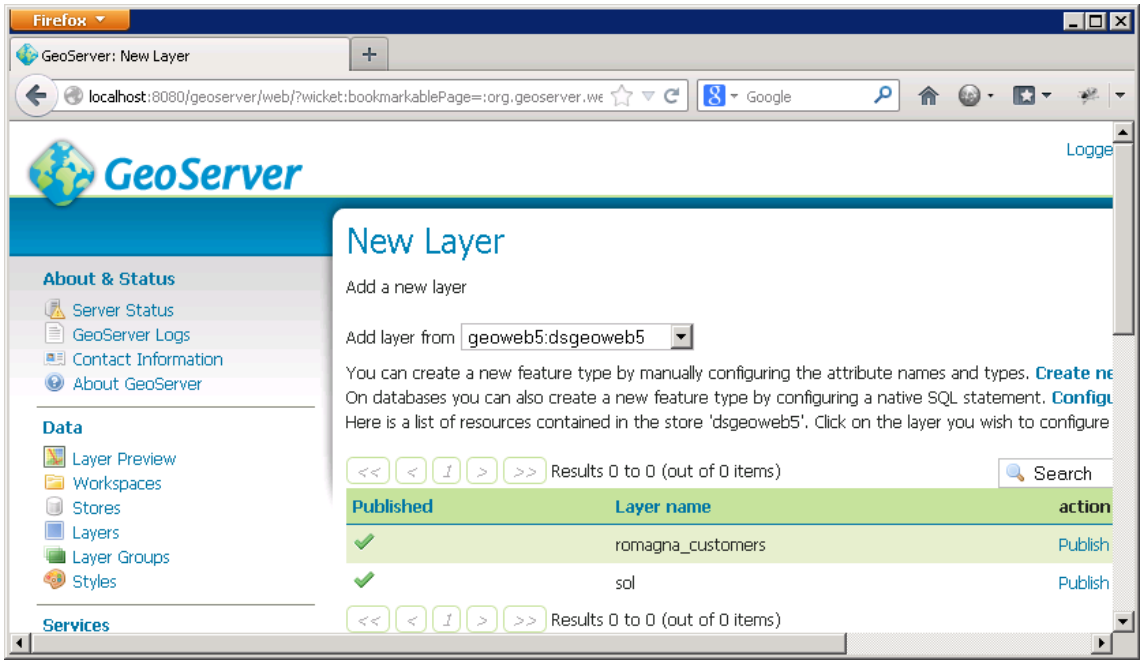
Nella pagina successiva si può osservare un esempio di questa configurazione. Inoltre si può osservare dall'immagine che si rendono già necessarie le informazioni generate nel capitolo precedente per la connessione alla base di dati.

Terminata questa fase di connessione si potrà procedere all'esposizione/pubblicazione dei dati tramite WFS di GeoServer.


Firefox
GeoServer: New Vector Data Source
localhost:8080/geoserver/web/?wicket:
Google
About & Status
Server Status
GeoServer Logs
Summary of server configuration and status
About GeoServer
Data
Layer Preview
Workspaces
Stores
Layers
Layer Groups
Styles
Services
WCS
WFS
WMS
Settings
Global
JAI
Coverage Access
Tile Caching
Tile Layers
Caching Defaults
Gridsets
Disk Quota
Security
Settings
Authentication
Passwords
Users, Groups, Roles
Data
New Vector Data Source
Add a new vector data source
PostGIS
Basic Store Info
Workspace *
geoweb5
Data Source Name *
Description
Enabled
Connection Parameters
host *
localhost
port *
5432
database
schema
public
user *
admin
passwd
Namespace *
http://geoweb5/valenti.alessandro/tesi
Expose primary keys
max connections
localhost:8080/geoserver/web/?wicket:bookmarkablePage=:org.geoserver.web.admin.StatusPage

Pubblicazione dei dati

Si è giunti alla pubblicazione vera e propria dei dati memorizzati sul database, per esporre questi dati ci avvalleremo del menu Layers aggiungendo nuove risorse tramite "Add new resource" selezionando la nostra sorgente dati `geoweb5:dsgeoweb5` si avrà quindi modo di pubblicare i nostri oggetti



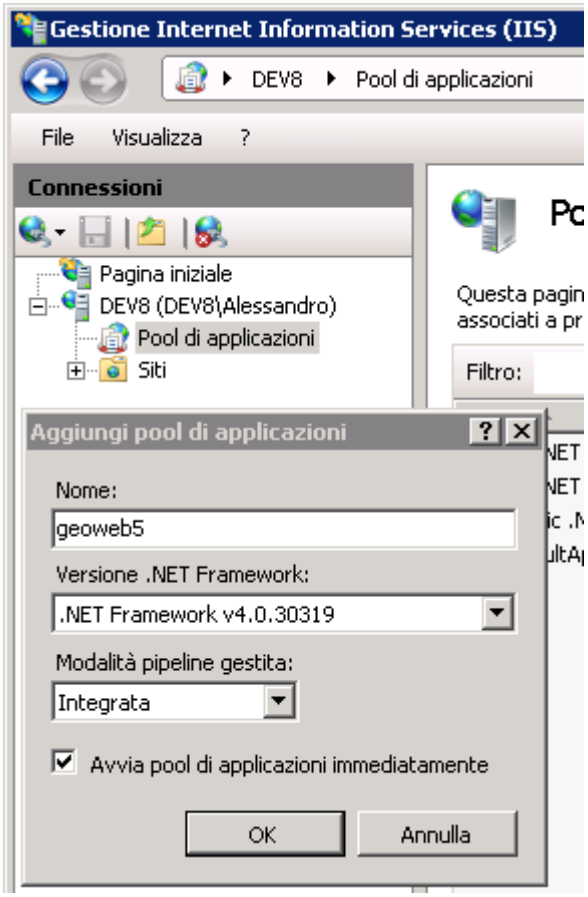
Abbiamo concluso la parte di esposizione dei dati tramite GeoServer, ora i layer sono visibili e modificabili tramite WFS.

Ambiente di runtime IIS

Una volta installato e configurato la parte dati si potrà iniziare la procedura di installazione dell'applicativo.
Questa semplice procedura consiste esclusivamente in due parti:

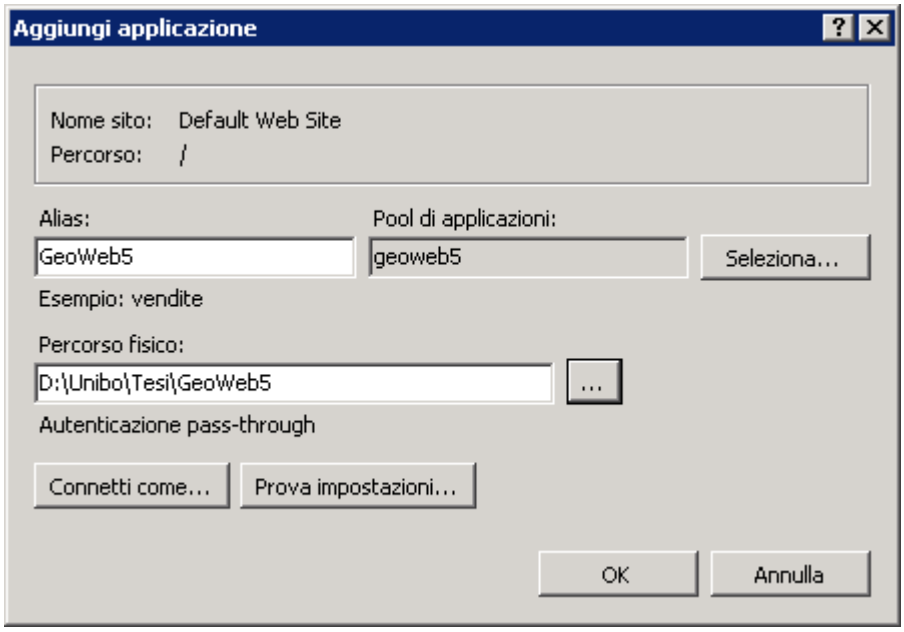
1. Creazione dell'application pool;
2. Creazione della directory virtuale;

Creazione dell'application pool



Dalla tendina "Pool di applicazioni" con il tasto destro si esegue "Aggiungi pool di applicazioni…" e lo si configurerà dandogli un nome a piacimento, `geoweb5` nel nostro progetto; selezioniamo la versione 4 del Framework .NET che deve essere preinstallato. Inoltre per motivi di performance si è utilizzata la modalità Integrata (Vedi Tecnologie Utilizzate)

A questo punto per installare l'applicativo sarà sufficiente copiare i file dell'applicativo in una directory a piacere, successivamente, aprendo la tendina "Siti" e selezionato il sito Web su cui installare la directory virtuale, eseguire il Wizard "Aggiungi Applicazioni…"



A questo punto l'applicazione è pronta per girare con le configurazioni di default ed è in grado di visualizzare i primi risultati.

L'applicativo è fornito un file di installazione che automatizza questa parte, è parte integrante di Visual Studio e permette di pubblicare un "Pacchetto" integrato dentro Internet Information Services.

Configurazioni

L'intero applicativo è guidato dal file di configurazione web.config in cui si configura il comportamento dell'applicazione, nel prossimo capitolo si entrerà nell'implementazione vera e propria, però, già dalla configurazione si intuiranno diversi aspetti dell'applicazione.

E' riportata una versione ridotta del file, che sarà spiegata step by step:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<configSections>
<connectionStrings>
<add name="GeoWeb5.Properties.Settings.DBConn"
connectionString="Server=localhost;Port=5432;User
Id=postgres;Password=postgres;Database=geoweb5;" />
<add name="GeoWeb5.Properties.Settings.WFSConn"
connectionString="http://localhost:8080/geoserver/wfs" />
<add name="GeoWeb5.Properties.Settings.WFSNameSpace"
connectionString="http://geoweb5/valenti.alessandro/tesi"
providerName="" />
</connectionStrings>
<system.web>
<compilation debug="true" targetFramework="4.5" />
<httpRuntime targetFramework="4.5" />
<caching>
<outputCacheSettings>
<outputCacheProfiles>
<add name="CacheGeoWeb5" duration="3600" varyByParam="*"
location="ServerAndClient"/>
</outputCacheProfiles>
</outputCacheSettings>
</caching>
</system.web>
<applicationSettings>
<GeoWeb5.Properties.Settings>
<setting name="bbox" serializeAs="String">
<value>44.232,12.031,9</value>
</setting>
<setting name="imgUrlProxy" serializeAs="String">
<value>http://b.tile.openstreetmap.org/</value>
</setting>
<setting name="minZoom" serializeAs="String">
<value>8</value>
</setting>
<setting name="maxZoom" serializeAs="String">
<value>18</value>
```

```
</setting>
<setting name="imgUrl" serializeAs="String">
<value>/ImageApi/GetImage?lat={lat}&amp;lon={lon}&amp;zoom={zoom
}&amp;mapWidth={width}&amp;mapHeight={height}</value>
</setting>
<setting name="debug" serializeAs="String">
<value>False</value>
</setting>
<setting name="warehouseId" serializeAs="String">
<value>13058</value>
</setting>
<setting name="ApplicationRoot" serializeAs="String">
<value>/GeoWeb5/</value>
</setting>
<setting name="DataSourceType" serializeAs="String">
<value>WFS</value>
</setting>
</GeoWeb5.Properties.Settings>
</applicationSettings>
</configuration>
```

Stringhe di connessione ai dati

```
<connectionStrings>
<add name="GeoWeb5.Properties.Settings.DBConn"
connectionString="Server=localhost;Port=5432;User
Id=postgres;Password=postgres;Database=geoweb5;" />
<add name="GeoWeb5.Properties.Settings.WFSConn"
connectionString="http://localhost:8080/geoserver/wfs" />
<add name="GeoWeb5.Properties.Settings.WFSNameSpace"
connectionString="http://geoweb5/valenti.alessandro/tesi"
providerName="" />
</connectionStrings>
```

Si identificano due sezioni principali per l'accesso ai dati, uno diretto a PostgreSQL in cui è possibile configurare la stringa di connessione ADO a PostgreSQL editando la configurazione DBConn, in alternativa è possibile utilizzare GeoServer e il servizio WFS, in questo caso dobbiamo editare WFSConn e WFSNameSpece che sono rispettivamente l'Url al servizio WFS esposto da GeoServer e in WFSNameSpece si deve inserire quel Namespace URI definito al capitolo precedente (Ambiente dei dati GeoServer – Creazione del Workspace).

L'attivazione del tipo di Data Source avviene mediante una chiave di configurazione in calce al file di configurazione stesso

```
<setting name="DataSourceType" serializeAs="String">
<value>WFS</value>
</setting>
```

I valori ammessi sono “WFS” e “DB”, nel prossimo capitolo “Realizzazione Server Side” si vedrà l’implementazione e l’astrazione mediante Abstract Factory di un generico Data Access Layer trasparente all’implementazione; questa astrazione permette di utilizzare il linguaggio SQL o il protocollo SOA di WFS in modo trasparente al chiamante.

Output Cache

Questo è un esempio di un Managed Module messo a disposizione dal framework e utilizzato nell’applicativo per migliorare le performance.

```
<caching>
<outputCacheSettings>
<outputCacheProfiles>
<add name="CacheGeoWeb5" duration="3600" varyByParam="*"
location="ServerAndClient"/>
</outputCacheProfiles>
</outputCacheSettings>
</caching>
```

In particolare è interessante osservare la semplicità con cui il framework .Net permette l’utilizzo di questi moduli in modo trasparente, qui sotto viene anticipato un pezzo di codice che sarà approfondito nel capitolo “Realizzazione”

```
[OutputCache(CacheProfile = "CacheGeoWeb5")]
public class ImageApiController : Controller
{  }
```

Configurazioni Applicative

Le configurazioni in "bbox" rappresentano le informazioni sulla posizione iniziale dell'applicativo ed in particolare il punto di centro (latitudine e longitudine) dell'applicativo e lo zoom di partenza

```
<GeoWeb5.Properties.Settings>
<setting name="bbox" serializeAs="String">
<value>44.232,12.031,9</value>
</setting>
```

Nella configurazione "imgUrlProxy" risiede un'altra configurazione polimorfica che è in grado di attivare un image provider che può recuperare tile tramite http, oppure, tramite file system; in particolare se questa chiave inizia con "http" si attiverà l'ImageProvider su chiamate http diversamente attiva l'ImageProvider basato su file system, in particolare, nel progetto finale si sono utilizzati tile specifici che risiedono a FS ma anche esposti tramite http da IIS come "Virtual Directory", qualunque sia la sorgente è importante che la sorgente rispetti i vincoli dell'organizzazione secondo la gerarchia TMS.

```
<setting name="imgUrlProxy" serializeAs="String">
<value>http://b.tile.openstreetmap.org/</value>
</setting>

<setting name="imgUrlProxy" serializeAs="String">
<value>http://localhost/tiles/all</value>
</setting>
```

Oppure per attivare l'Image Provider relativo al File System

```
<setting name="imgUrlProxy" serializeAs="String">
<value>C:\tiles\all</value>
</setting>
```

La chiave "imgUrl" è una parte sistemistica, il template del collante tra la parte client e la parte server, non può quindi essere alterata; rappresenta la rotta che

viene chiamata da JavaScript per ottenere l'immagine della cartina su cui poi verranno disegnate le soluzioni.

```
<setting name="imgUrl" serializeAs="String">
<value>/ImageApi/GetImage?lat={lat}&amp;lon={lon}&amp;zoom={zoom
}&amp;mapWidth={width}&amp;mapHeight={height}</value>
</setting>
```

La chiave "ApplicationRoot" come dice il nome stesso, rappresenta il nome della Virtual Directory dove è stato installato l'applicativo, se avessimo fatto un nuovo sito web la chiave dovrebbe essere semplicemente "/".

Nella nostra implementazione essendo un'Applicazione all'interno del default site, e avendola chiamata GeoWeb5 l'application root sarà:

```
<setting name="ApplicationRoot" serializeAs="String">
<value>/GeoWeb5/</value>
</setting>
```

Infine viene specificato quale nodo è il nostro warehouse, il warehouse ha un comportamento particolare, è sempre il primo nodo di una soluzione, non può essere alterato, aggiunto o rimosso da una soluzione, questo Id deve essere presente nella tabella `romagna_customers`

```
<setting name="warehouseId" serializeAs="String">
<value>13058</value>
</setting>
```

Realizzazione

In questo capitolo verrà presentato nel dettaglio l'applicativo, mostrando e motivando, le scelte architetturali mostrandone l'implementazione. Si inizierà parlando dal layer data, per risalire al middle tier in cui risiede la composizione e la preparazione dei dati per poi giungere alla parte finale che è l'implementazione di front end in cui risiede gran parte della logica applicativa.

Server Side – Data Layer

In questo capitolo si vedrà l'implementazione di due component fondamentali dell'architettura; entrambi implementati tramite il pattern creazionale Abstract Factory ed in particolare il componente DataAccessFactory che restituisce i dati e ImageFactory che restituisce le immagini dei tile. In base al file di configurazione le factory istanziano il provider dei dati richiesto; implementano protocolli diversi ma condividendo la stessa interfaccia rendono trasparente il dato al client. Ricordando la definizione "Abstract Factory fornisce un'interfaccia per creare famiglie di oggetti correlati o dipendenti senza specificare le classi concrete".

DataAccessFactory

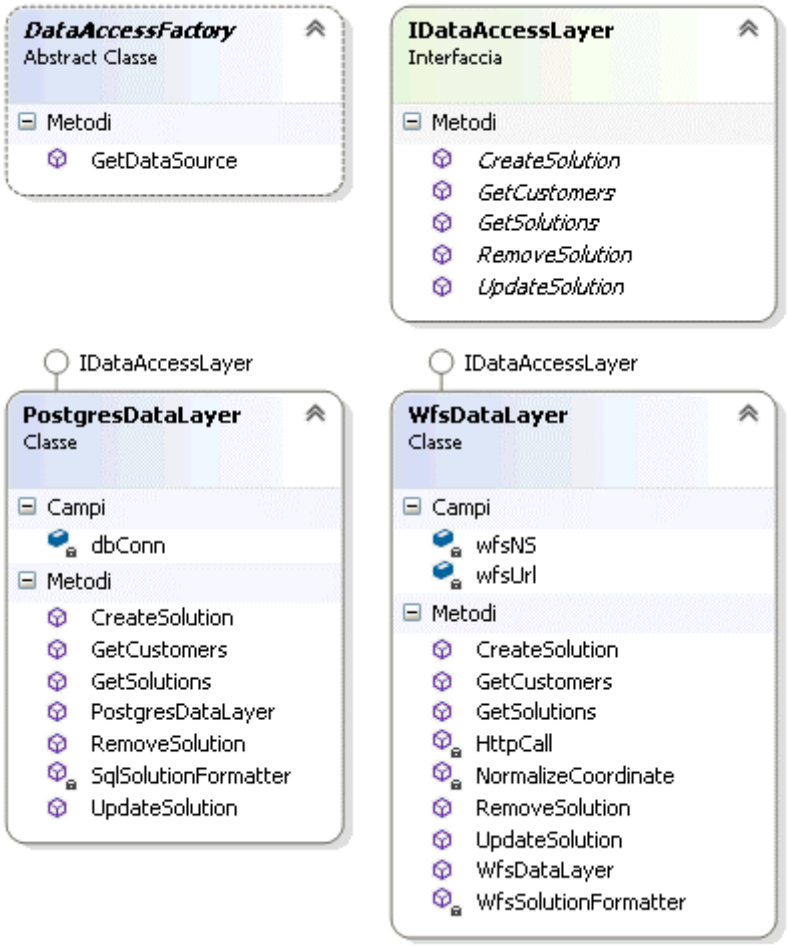
Questa classe astratta contiene un solo metodo "creazionale" GetDataSource che legge il file di configurazione e crea le classi concrete che implementano l'interfaccia IDataAccessLayer utilizzata dall'applicativo per interagire con i dati.

```
public static IDataAccessLayer GetDataSource(string dataSourceType){
       if (dataSourceType.ToLower() == "wfs")
              return new WfsDataLayer();

       return new PostgresDataLayer();
}
```

Il valore in input come detto precedentemente viene preso dalla relativa chiave nel file di configurazione (web.config)

```
<setting name="DataSourceType" serializeAs="String">
<value>WFS</value>
</setting>
```

Qui sotto viene riportata la struttura delle classi, l'applicativo non utilizzerà mai le classi concrete PostgresDataLayer e WfsDataLayer ma lavorerà con la generica interfaccia IDataAccessLayer; questa architettura permette una crescita verticale dei provider dei dati, implementando, l'accesso a diverse sorgenti dati; ad esempio potrebbero implementarsi OracleDataLayer, UniboDataLayer senza che l'applicativo si accorga del cambio di DataLayer sottostante.



L'interfaccia IDataAccessLayer definisce i contratti di business fondamentali per l'applicativo, sarà interessante entrare nel dettaglio applicativo per mostrare le estensioni GIS di PostgreSQL e il protocollo WFS di GeoServer. Prima di entrare nel dettaglio si nota immediatamente che le classi concrete, necessitano e recuperano, autonomamente dal file di configurazione le configurazioni che gli servono per svolgere il proprio lavoro ed in particolare la stringa di connessione per PostgreSQL (dbConn) e l'Url del servizio WFS e il namespace uri per operare con gli oggetti di GeoServer. Nei successivi paragrafi vedremo l'implementazione dei singoli metodi.

GetCustomers

Questo metodo ritorna una lista di Customer ricavati dalla tabella `romagna_customers` entrambi i provider risultano interessanti.

PostgresDataLayer:

```
SELECT DISTINCT idnodo, st_astext(geometry) AS geom
FROM romagna_customers
```

Di particolare interesse la funzione `st_astext` che permette di trasformare il tipo dato geometrico in un tipo dato comprensibile all'utente, un estratto del risultato che verrà poi normalizzato dal Provider e adattato al tipo dell'interfaccia

```
31585;"POINT(12.5123582 44.0719606)"
28934;"POINT(12.4154118 44.180266)"
```

WfsDataLayer esegue una chiamata SOAP con method action POST e nel risultato otteniamo una struttura Xml che rappresenta gli stessi dati di cui sopra:

Request POST:

```
<?xml version="1.0" encoding="utf-8" ?>
<wfs:GetFeature service='WFS' version='2.0.0'
xmlns:fes='http://www.opengis.net/fes/2.0'
xmlns:wfs='http://www.opengis.net/wfs/2.0'>
  <wfs:Query typeNames='geoweb5:romagna_customers'/>
</wfs:GetFeature>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection . . .
xmlns:geoweb5=http://geoweb5/valenti.alessandro/tesi
xmlns:wfs="http://www.opengis.net/wfs/2.0">
  <wfs:member>
    <geoweb5:romagna_customers gml:id="romagna_customers.1">
      <geoweb5:idnodo>24605.0</geoweb5:idnodo>
      <geoweb5:geometry>
        <gml:Point srsDimension="2" srsName="urn:ogc:def:crs:EPSG::4326">
          <gml:pos>44.4894436 12.2836943</gml:pos>
        </gml:Point>
      </geoweb5:geometry>
    </geoweb5:romagna_customers>
  </wfs:member>
  <wfs:member>
    <geoweb5:romagna_customers gml:id="romagna_customers.2">
      <geoweb5:idnodo>25759.0</geoweb5:idnodo>
      <geoweb5:geometry>
        <gml:Point srsDimension="2" srsName="urn:ogc:def:crs:EPSG::4326">
          <gml:pos>44.1186895 12.327729</gml:pos>
        </gml:Point>
      </geoweb5:geometry>
    </geoweb5:romagna_customers>
  </wfs:member>
</wfs:FeatureCollection
```

Nelle risposte WFS a Query è di particolare importanza l'attributo

`gml:id="romagna_customers.1"`

Questo attributo e il suo valore `romagna_customers.1` è l'identificativo univoco che sarà utilizzto per manipolare il dato con le istruzioni DML (Insert, Update, Delete) di WFS.

Le tecnologie utilizzate all'interno dei Provider concreti sono totalmente diverse, PostgreSQL utilizza ADO.NET, tutti oggetti messi a disposizione dal relativo provider, connessione, comando e result set che viene normalmente ciclato per creare la lista di ritorno; per WFS abbiamo utilizzato il protocollo http e utilizzato Xml con XPath per capire la risposta e trasformarlo nella lista i oggetti richiesti dall'interfaccia.

GetSolutions

Non ci si soffermerà molto su questo metodo, è speculare al metodo GetCustomers visto precedentemente.

```
SELECT gid, idroute, st_astext(geom) AS geom, color
FROM sol
ORDER BY idroute
```

Una nota interessante è che la funzione vista precedentemente `st_astext` è polimorfica rispetto al tipo di dato definito nel tipo Geometry.

```
1;0;"MULTILINESTRING((12.031421 44.232374,13.028345
43.839599,12.988309 43.831646,13.073654 43.81585,13.02284
43.836481,13.015981 43.833737,12.415412 44.180266))";"#FF0000"
2;1;"MULTILINESTRING((12.031421 44.232374,12.571096
44.042364))";"#FF0000"
```

La versione WFS è identica al precedente metodo

```
<?xml version="1.0" encoding="utf-8" ?>
<wfs:GetFeature service='WFS' version='2.0.0'
xmlns:fes='http://www.opengis.net/fes/2.0'
xmlns:wfs='http://www.opengis.net/wfs/2.0'>
  <wfs:Query typeNames='geoweb5:sol'/>
</wfs:GetFeature>
```

La risposta merita di nota soprattutto nella parte di rappresentazione di MULTILINESTRING

```
<wfs:member>
  <geoweb5:sol gml:id="sol.2">
    <geoweb5:idroute>1.0</geoweb5:idroute>
    <geoweb5:geom>
      <gml:MultiCurve srsDimension="2"
srsName="urn:ogc:def:crs:EPSG::4326">
        <gml:curveMember>
          <gml:LineString srsDimension="2">
            <gml:posList>44.2323742 12.0314211 44.042364
12.5710957 43.6755417 12.6460579 43.8046553 12.9126099
43.7909984 13.0010638</gml:posList>
          </gml:LineString>
        </gml:curveMember>
      </gml:MultiCurve>
    </geoweb5:geom>
```

```
    <geoweb5:color>#FF0000</geoweb5:color>
  </geoweb5:sol>
</wfs:member>
```

Fino ad ora si è visto solo semplici risposte a Query, successivamente entreremo nel cuore del WFS, le transazioni dette WFS-T con le quali si potranno eseguire Insert, Update e Delete. Ricordando sempre l'attributo che sarà fondamentale per queste istruzioni `gml:id="sol.2"`

CreateSolution

Nella versione PostgreSQL è una normale istruzione SQL di INSERT, utilizzando la funzione `st_geomfromtext` le estensioni Postgis sono in grado di trasformare una stringa MULTILINESTRING in un dato binario memorizzabili sulla tabella

```
INSERT INTO sol(idroute, geom, color) VALUES (999,
ST_GeomFromText('SRID=4326;MULTILINESTRING((44.1234 12.567,
45.453 13.999, . . .))'), '#00FF00')
```

Mentre nella versione WFS dobbiamo eseguire una chiamata POST specificando la chiamata wfs:Transaction e wfs:Insert come qui sotto:

```
<?xml version="1.0" encoding="utf-8" ?>
<wfs:Transaction version="1.1.0" service="WFS" handle="Transaction 01"
xmlns="http://www.someserver.com/myns" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:geoweb5="http://geoweb5/alessandro.valenti/tesi">
  <wfs:Insert handle="Statement 1">
    <geoweb5:sol>
      <geoweb5:idroute>999</geoweb5:idroute>
      <geoweb5:color>#00FF00</geoweb5:color>
      <geoweb5:geom>
        <gml:MultiLineString xmlns:gml="http://www.opengis.net/gml"
srsName="EPSG:4326">
          <gml:lineStringMember>
            <gml:LineString srsName="EPSG:4326">
              <gml:coordinates  decimal="." cs="," ts=" ">
                    44.1234 12.567, 45.453 13.999, ...
              </gml:coordinates>
```

```
            </gml:LineString>
          </gml:lineStringMember>
        </gml:MultiLineString>
      </geoweb5:geom>
    </geoweb5:sol>
  </wfs:Insert>
</wfs:Transaction>
```

La risposta a questa chiamata è interessante per vedere le risposte generiche a istruzioni WFS-T

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:TransactionResponse version="1.1.0"
xsi:schemaLocation="http://www.opengis.net/wfs
http://localhost:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:tiger="http://www.census.gov"
xmlns:cite="http://www.opengeospatial.net/cite"
xmlns:nurc="http://www.nurc.nato.int" xmlns:sde="http://geoserver.sf.net"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:aaa="http://aaa"
xmlns:topp="http://www.openplans.org/topp" xmlns:it.geosolutions="http://www.geo-
solutions.it" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sf="http://www.openplans.org/spearfish"
xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:geoweb5="http://geoweb5/valenti.alessandro/tesi"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <wfs:TransactionSummary>
    <wfs:totalInserted>1</wfs:totalInserted>
    <wfs:totalUpdated>0</wfs:totalUpdated>
    <wfs:totalDeleted>0</wfs:totalDeleted>
  </wfs:TransactionSummary>
  <wfs:TransactionResults/>
  <wfs:InsertResults>
    <wfs:Feature handle="Statement 1">
      <ogc:FeatureId fid="sol.44"/>
    </wfs:Feature>
  </wfs:InsertResults>
</wfs:TransactionResponse>
```

Come si può notare restituisce un sommario delle tuple alterate, ma soprattutto, restituisce l'id univoco del record inserito (`fid="sol.44"`), valore univoco che sarà poi fondamentale per alterare o rimuovere la soluzione.

UpdateSolution e RemoveSolution

Questi metodi sono molto interessanti in quanto sulla chiamata WFS deve essere specificato quell'identificativo univoco che abbiamo visto restituirci sia nelle semplici Query e visto precedentemente nell'istruzione di Insert. Vedremo quindi le istruzioni wfs:Update e wfs:Delete

```
<?xml version="1.0" encoding="utf-8" ?>
<wfs:Transaction version="1.1.0" service="WFS"
xmlns:myns="http://www.someserver.com/myns"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
xmlns:geoweb5=" http://geoweb5/valenti.alessandro/tesi ">
  <wfs:Update typeName="geoweb5:sol">
    <wfs:Property>
      <wfs:Name>geoweb5:geom</wfs:Name>
      <wfs:Value>
        <gml:MultiLineString xmlns:gml="http://www.opengis.net/gml"
srsName="EPSG:4326">
          <gml:lineStringMember>
            <gml:LineString srsName="EPSG:4326">
              <gml:coordinates  decimal="." cs="," ts=" ">
                    44.1234 12.567, 45.453 13.999, ...
              </gml:coordinates>
            </gml:LineString>
          </gml:lineStringMember>
        </gml:MultiLineString>
      </wfs:Value>
    </wfs:Property>
    <ogc:Filter>
      <ogc:FeatureId fid="sol.44"/>
    </ogc:Filter>
  </wfs:Update>
</wfs:Transaction>
```

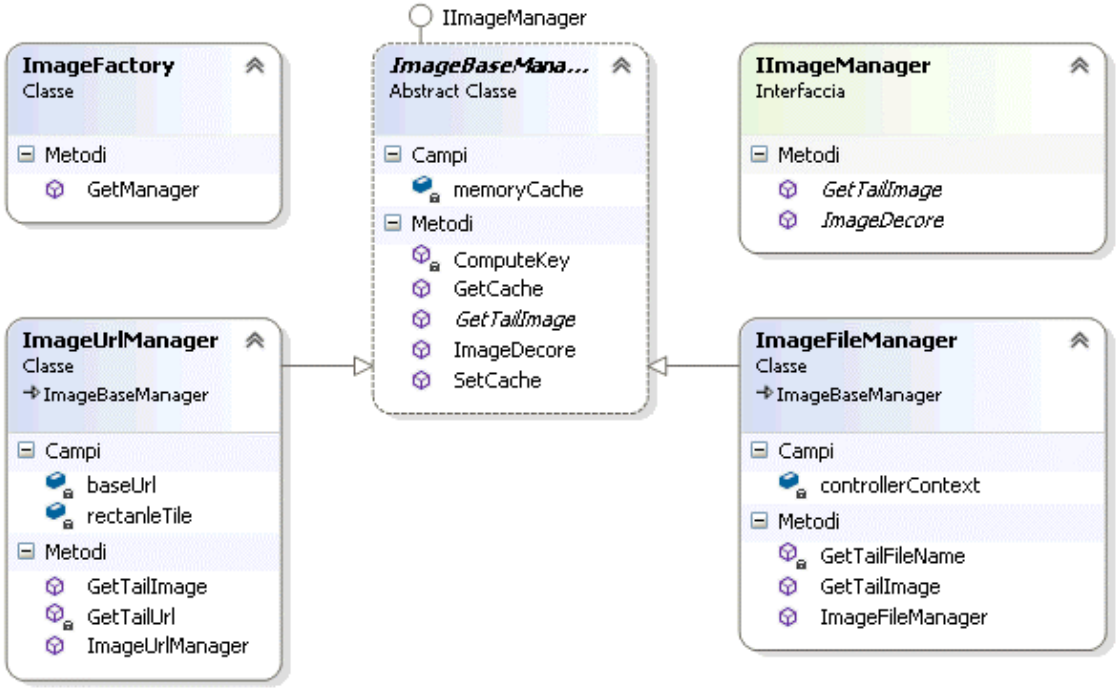
```
<?xml version="1.0" encoding="utf-8" ?>
<wfs:Transaction version="1.1.0" service="WFS"
xmlns="http://www.someserver.com/myns" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
xmlns:geoweb5=" http://geoweb5/valenti.alessandro/tesi ">
  <wfs:Delete typeName="geoweb5:sol">
    <ogc:Filter>
      <ogc:FeatureId fid="sol.44"/>
    </ogc:Filter>
  </wfs:Delete>
</wfs:Transaction>
```

In entrambi i metodi è interessante notare la proprietà `ogc:Filter` che permette di identificare il record su cui operare.

ImageFactory

Questa Factory astratta è deputata a recuperare lato server i tile provenienti da diverse fonti, anche questa implementazione è stata progettata in versione Abstract Factory per avere un'architettura estensibile, per il momento possiamo recuperare i tile organizzati secondo la tecnologia TMS tramite http o tramite file system, a sua volta utilizzando http possiamo utilizzare i tile di openstreetmap.org, i tile esposti tramite IIS come directory virtuale, quindi, si possono utilizzare tile custom generati ad esempio con Mapnik. In alcuni casi particolari possiamo accedere direttamente a questi tile tramite file system per maggiori performance. Si vedranno quindi le due implementazioni concrete ImageUrlManager e ImageFileManager.

Grazie a questa scelta architetturale si possono immaginare future estensioni come ImageGoogleManager, ImageBingManager, sarà poi l'ImageController che sarà presentato nel prossimo capitolo a comporre i tile per generare l'immagine completa che sarà utilizzata dal tag `canvas` di Html5.



Nota su questa implementazione; entrambi le classi concrete ereditano da una classe base in cui è stata definita una gestione cache utile per le performance, in particolare, un tile può essere facilmente messo in cache in quanto è univocamente identificato e a scarsa variabilità, di conseguenza può essere mantenuto in cache per diverso tempo.

Middle Tier – Business Logic

Adesso che si hanno a disposizione gli oggetti deputati a recuperare le informazioni lato server e si possono manipolare, si possono creare strutture (model) ad hoc per l'applicativo vero e proprio; in questo capitolo saranno mostrati gli oggetti che il server metterà a disposizione del client Html, il quale, a sua volta li manipolerà, li restituirà al server. In pratica definiremo i protocolli, i messaggi e le strutture dati.

Partiremo dalla presentazione dei protocolli utilizzati, protocolli che il server mette a disposizione del client Html. Per implementare questo strato software si è utilizzata l'ultimissima tecnologia messa a disposizione dal Microsoft Framework 4, ApiController, il quale implementa il pattern MVC orientato a servizi senza interfaccia UI, fornendo programmaticamente il tipo di risposta, sia Json o Xml. Successivamente si vedrà come il client Html utilizzerà le rotte esposte da questi servizi.

Api – Results

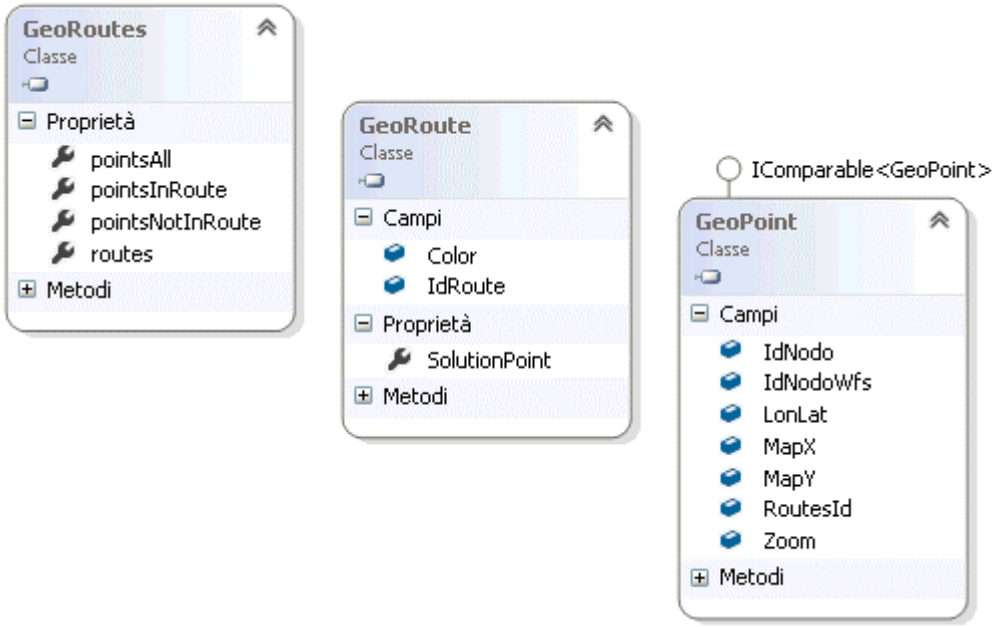
Per presentare questo metodo si partirà dall'entry point in un'architettura MVC, cioè dalla rotta esposta; quando la request rispetta il pattern "routeTemplate" attiva il controller e l'azione, controller inteso come gruppo di azioni, vediamo innanzitutto la rotta per ottenere le strutture dati dei `romagna_customers` e `sol`

```
config.Routes.MapHttpRoute(
 name: "Results",
 routeTemplate: "api/{controller}/{action}
                    /{zoom}/{canvasXCenter}/{canvasYCenter}
                    /{mapLonCenter}/{mapLatCenter}",
);
```

Esempio di chiamata

```
/api/GeoWeb5Api/Results/9/162/265/12,031/44,232
```

Attiverà il controller GeoWeb5ApiController e l'azione Results metodo che si occupa di tornare in formato Json o Xml la serializzazione dell'oggetto GeoRoutes, il quale contiene tutte le informazioni sui punti (`romagna_customers`) e sulle soluzioni nel campo routes (`sol`)



Questo oggetto root GeoRoutes contiene tutte le informazioni che permettono all'interfaccia client di capire le funzionalità disponibili per quel punto, se è possibile associarlo ad una soluzione, se è una coordinata valida ecc...

```
[HttpGet]
public GeoRoutes Results(string zoom, string canvasXCenter, string canvasYCenter,
string mapLonCenter, string mapLatCenter){ . . . }
```

L'intera infrastruttura ASP.NET MVC4 è molto legata a naming convention, in particolare quando associa il placeholder `/{controller}/` alla richiesta `/GeoWeb5Api/` cerca una classe che eredita Controller e si chiami esattamente come nella request più la parola chiave Controller; di conseguenza cerca un controller di nome GeoWeb5ApiController. Una volta identificato il Controller al suo interno cerca l'azione, cioè la funzione associata al placeholder `/{action}/` ed in questo caso `/Results/`; è inoltre in grado di selezionale l'overload della funzione da eseguire grazie al binding degli altri parametri della query string come:

```
/{zoom}/{canvasXCenter}/{canvasYCenter}/{mapLonCenter}/{mapLatCe
nter}
```

Al suo interno questo MVC Controller utilizza il DataAccessFactory introdotto nel precedente capitolo, per recuperare le informazioni server, si può dire che questo metodo prepara il Model che sarà utilizzata dalla parte di applicativo JavaScript che sarà documentato nel prossimo capitolo "Client Side".

Api – Image

ImageController espone una classica rotta MVC fornendo UI (l'immagine) mappa sulla quale verranno disegnati i punti e le soluzioni, in particolare come l'Api Results anche l'Api Image viene attivata secondo una rotta

```
/ImageApi/GetImage?lat=44.232&lon=12.031&zoom=9&mapWidth=530&map
Height=323
```

Ricordiamo che il template di questa chiamata è configurabile, per cui è facile cambiare il fornitore di immagine, riportiamo qui sotto la configurazione

```
<value>/ImageApi/GetImage?lat={lat}&amp;lon={lon}&amp;zoom={zoom
}&amp;mapWidth={width}&amp;mapHeight={height}</value>
```

Risulta particolarmente interessante questo Controller in quanto non è un ApiController ma nemmeno un normale controller con la View Razor di rendering. Questo Controller ritorna uno stream byte che è l'immagine dinamica ottenuta combinando i tile lato server esposti tramite l'ImageFactory esposta nel precedente capitolo.

Si vedranno allora le varie particolarità di questo Controller, la gestione della cache, la custom action di ritorno; si partirà dalla gestione della cache implementata semplicemente inserendo l'attributo OutputCache che attiva il modulo managed spiegato nel capito "Tecnologie Utilizzate" e configurato nel web.config

```
[OutputCache(CacheProfile = "CacheGeoWeb5")]
public class ImageApiController : Controller
{
 public ImageTailResult GetImage(string lat, string lon, string mapWidth, string
mapHeight, string zoom){
return new ImageTailResult(dLat, dLon, iZoom, iWidth, iHeight, "image/jpeg");
        }
    }
```

Successivamente una volta terminato il paragrafo si entrerà nell'implementazione Client Side con Javascript dove si vedrà con codice l'invocazione di questi metodi.

Prima di addentrarci nell'implementazione client è interessante approfondire come utilizzare l'architettura MVC ed estenderla, in questo caso il Controller invece di attivare una Vista Razor, scrive direttamente sullo stream di risposta il byte stream dell'immagine composta lato server dall'ImageFactory

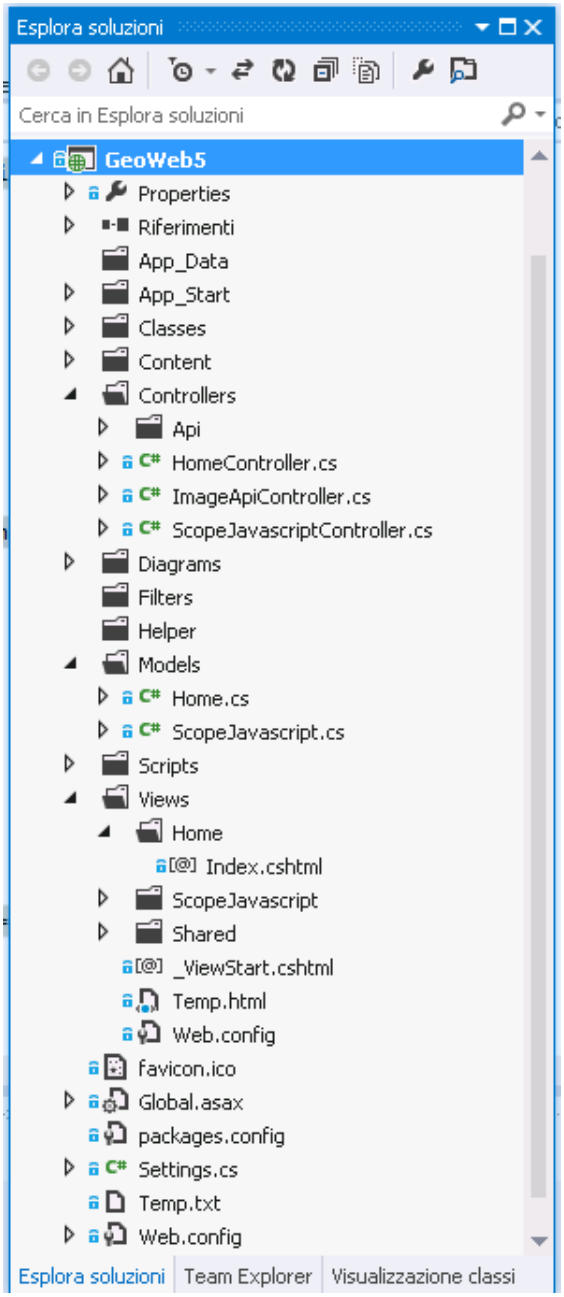
```
public class ImageTailResult : ActionResult
{
    public override void ExecuteResult(ControllerContext context)
    {
        var imageManager = ImageFactory.GetManager(context);
        . . .
        mapImage.Save(mapStream, codecInfo, encoderParameter);
        mapStream.Seek(0, SeekOrigin.Begin);
        try
        {
           HttpResponseBase response = context.HttpContext.Response;
           response.ContentType = ContentType;

           var buffer = new byte[4096];
```

```
            while (true)
            {
                int read = mapStream.Read(buffer, 0, buffer.Length);
                if (read == 0)
                    break;

                response.OutputStream.Write(buffer, 0, read);
            }

            response.End();
        }catch
        {

        }
    }
}
```

Risulta così evidente come sia semplice alterare il comportamento della struttura classica MVC, e come in particolare, è possibile definire Action Result personalizzate ereditando proprio dalla classe avente questo nome, semplicemente eseguendo l'override della funzione ExecureResult che ha il contesto della richiesta come parametro, mettendo quindi a disposizione l'oggetto risposta del contesto della richiesta. In questo caso si vede come il controller riceve la richiesta, viene calcolata l'immagine tramite l'oggetto `imageManager` ed infine lo stream dell'immagine generata venga riversata nell'oggetto risposta `response.OutputStream`.

MVC – Home con Razor

Si è quindi giunti all'implementazione dell'Html con ASP.NET MVC4, questa parte si occupa di generare l'Html di partenza, ossia il DOM che interagirà con l'applicativo JavaScript, ma questa parte sarà spiegata dettagliatamente nel capitolo "Client Side". Questa parte è minimale in quanto l'intero applicativo è single page senza postback.



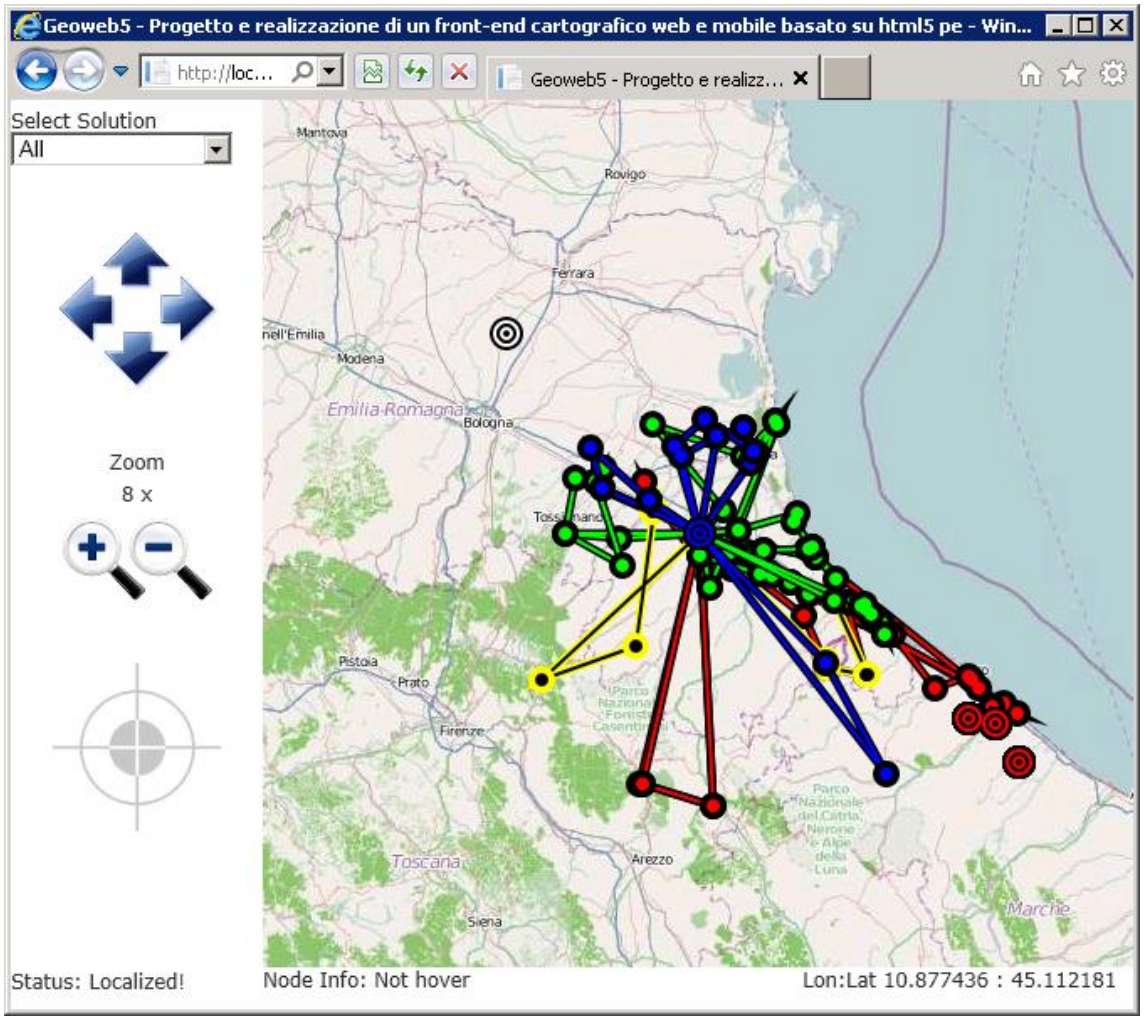
Nel progetto si hanno quindi tre cartelle principali per la renderizzazione della Home page. Una cartella dedicata ai Controller in cui troviamo HomeController e ScopeJavascriptController che recuperano rispettivamente il Model per generare le View e la PartialView le quali genereranno l'Html. Una volta che la rotta ha attivato l'azione specifica del Controller è il Controller a preparare il Modello per la View.

I modelli si trovano nella cartella Models, in queste classi ci sono le informazioni che la vista utilizza per scrivere l'Html dinamico.

Quindi, una volta che il controller ha creato i Models vengono attivate le viste all'interno della cartella Views e tramite Razor viene eseguito il "mash-up" tra il Model e la relativa View (Index.cshtml)

Ora abbiamo a disposizione l'Html con all'interno diversi tag `<div>` che rappresentano le varie parti dell'applicazione sul display e che tramite CSS verranno distribuite sul display, sia esso desktop o mobile; attiva JQuery e come si vedrà nel successivo capitolo per animare l'applicazione.

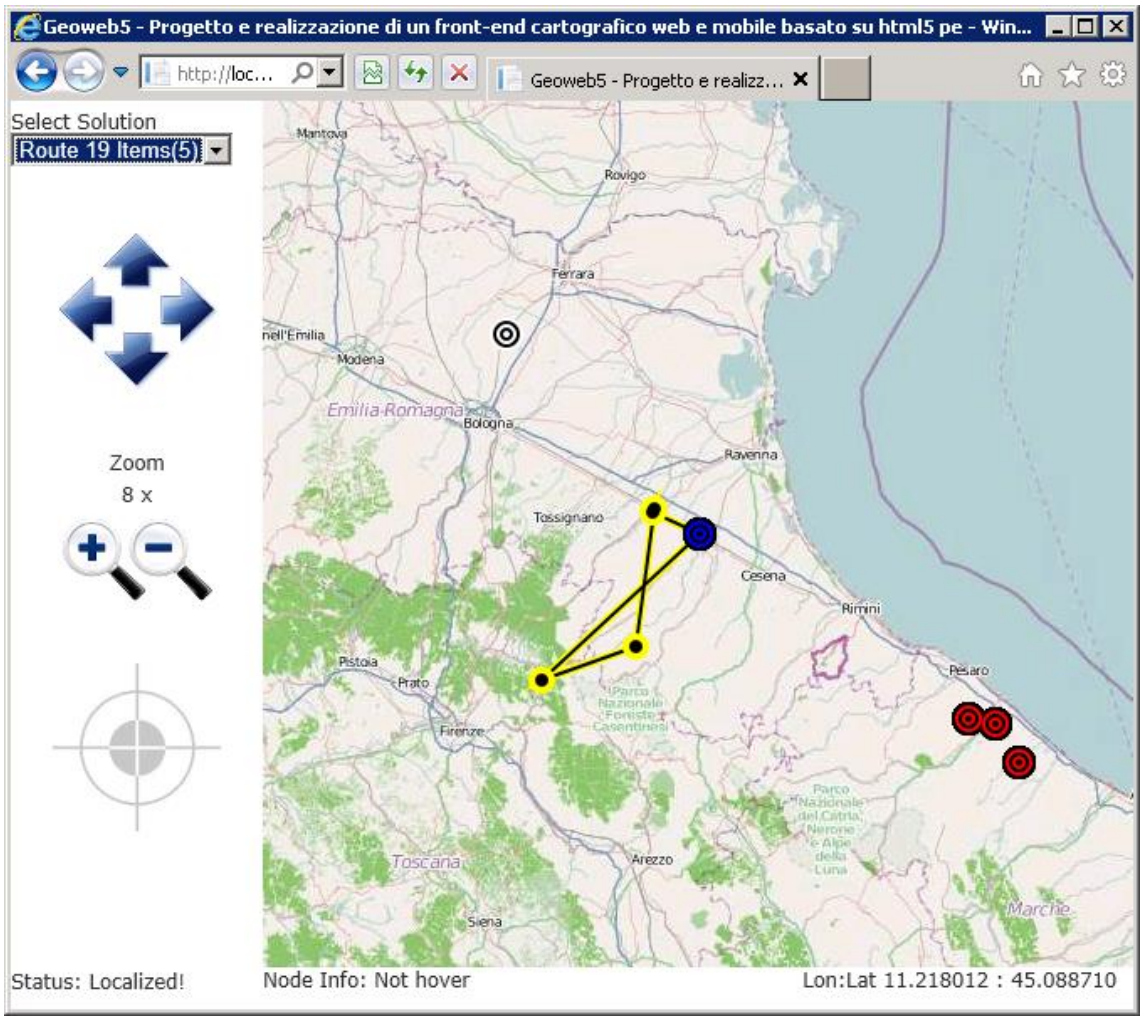
A questo punto è interessante cominciare a vedere il risultato dell'applicazione che risulta aderire allo standard Html5, risulta inoltre cross-browser e cross-device.



Dall'interfaccia si possono osservare alcune funzionalità ed in particolare:

- Il warehouse (web.config) identificato dal punto blu/nero;
- La nostra posizione geo localizzata (vedremo nel prossimo capitolo) identificato dal punto bianco/nero;
- I punti non associati a nessuna soluzione identificati da punti rossi/neri;

Inoltre possiamo identificare sul fondo del device lo stato della geo localizzazione (Status: Localized!), le info sul nodo una volta che il mouse lo attraversa (Node Info: Not hover.). Infine la latitudine e longitudine della posizione del mouse sulla cartina.

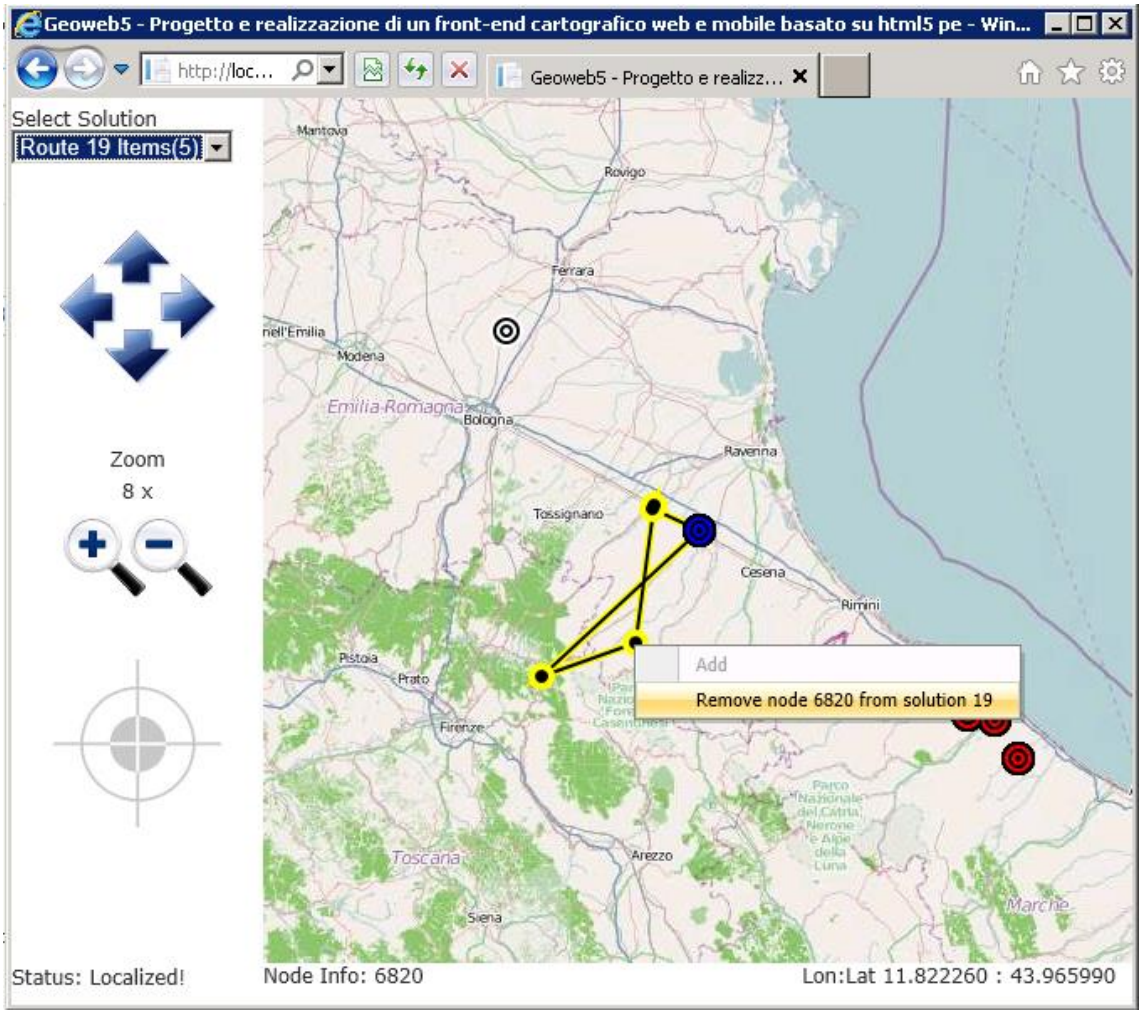


Sul lato sinistro dell'applicativo si evincono alcune funzionalità :

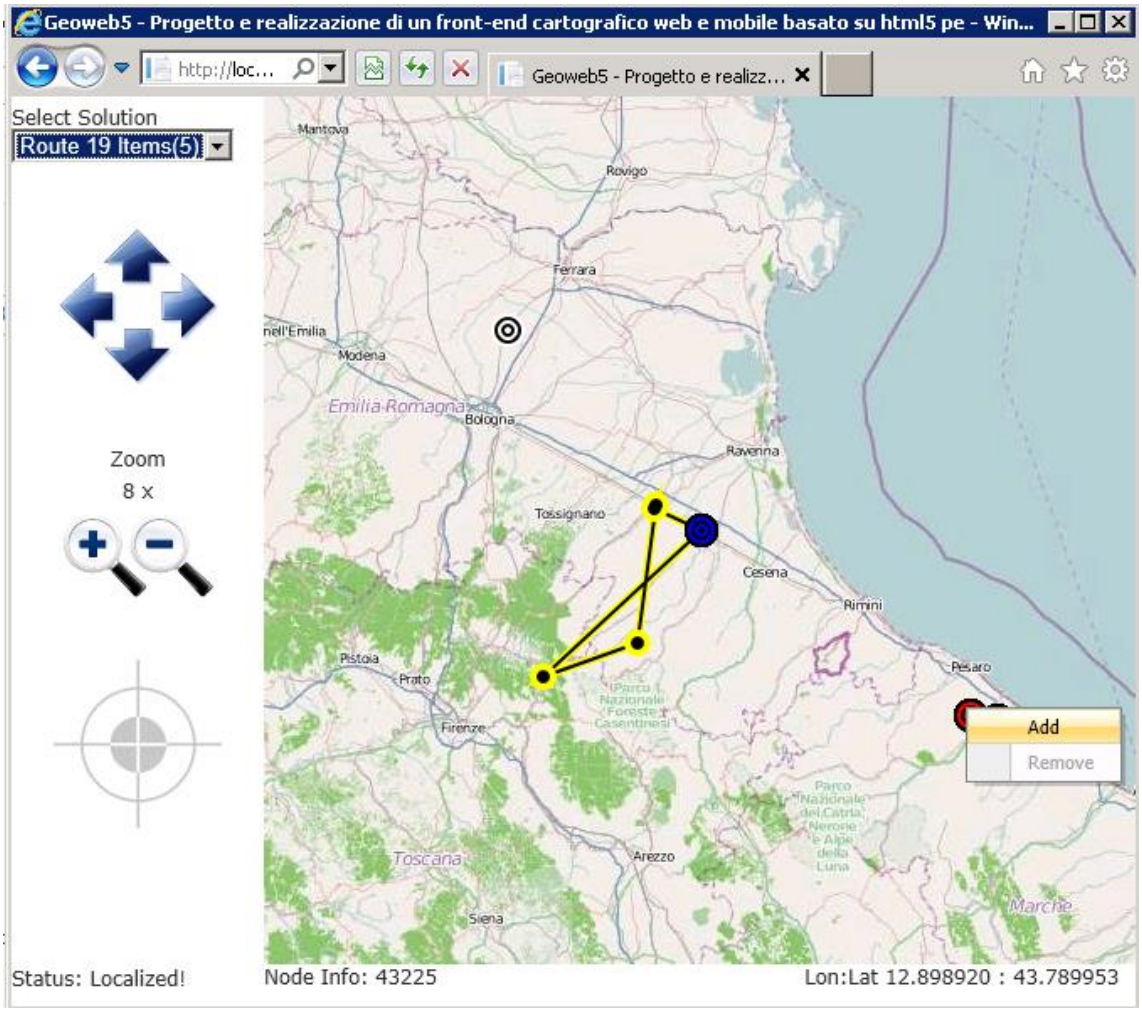
- Selezione della singola rotta (in questa immagine selezionata la sol 19 di colore giallo);
- Eseguire pan action della cartina, eseguibile cliccando sulle frecce blu oppure utilizzando il mouse (nella versione desktop);
- Eseguire lo zoom;
- Posizionare la cartina nel punto in cui siamo stati localizzati (punto bianco/nero);

Inoltre è possibile direttamente sui nodi della cartina alterare le soluzioni:

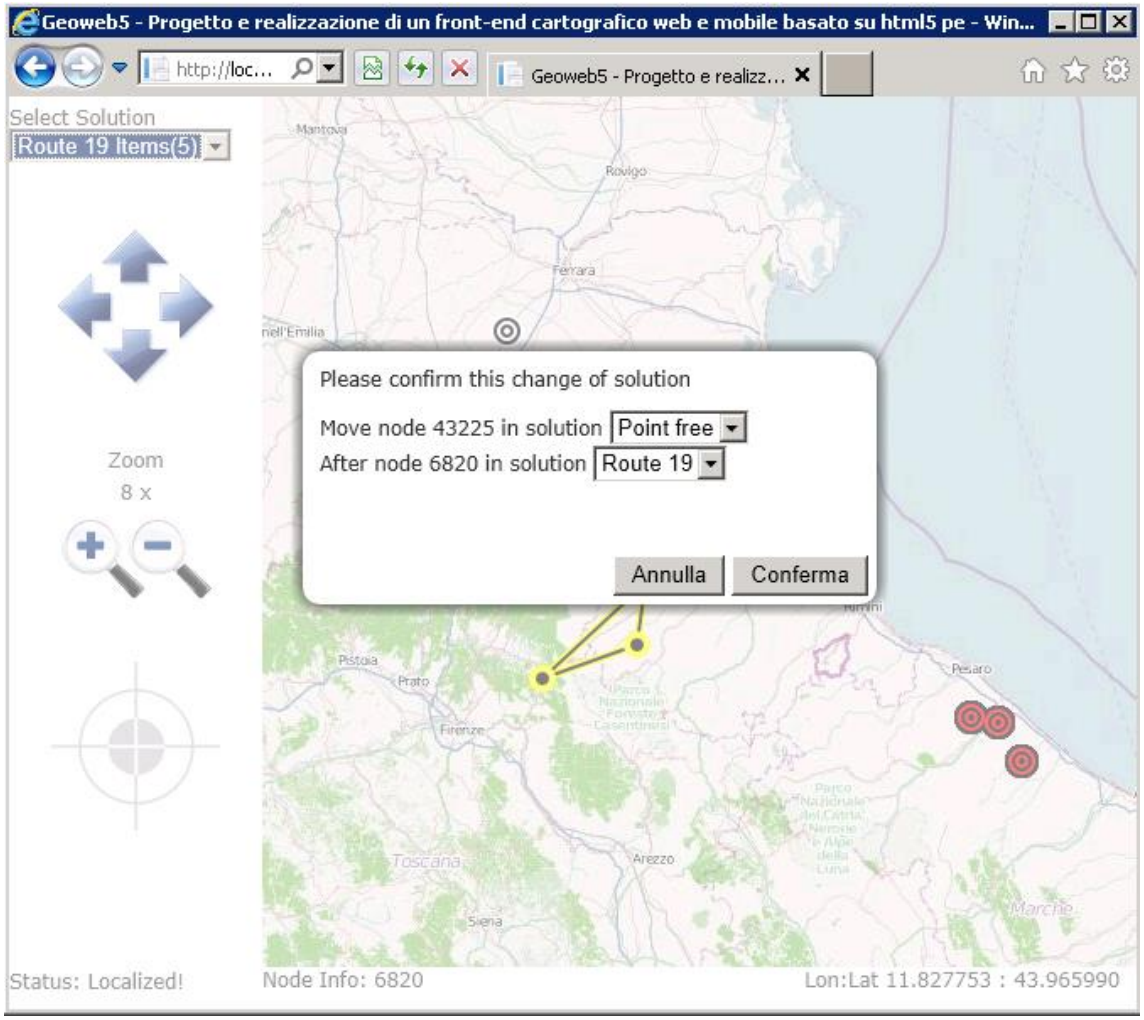
- Rimuovere un punto da una soluzione



- Creare una soluzione da un punto libero



- Funzioni drag & drop per associare un punto a una soluzione



Mobile

Grazie a CSS3 e Media Query precedentemente introdotte nel capitolo "Tecnologie Utilizzate" non è risultata particolarmente impegnativa la fase di realizzazione, lo sforzo è stato nel progettare un'interfaccia facile da utilizzare da un dispositivo con un desktop ridotto.

Sono quindi stati definiti due file fogli di stile, uno deputato alla macro struttura del sito "Site.css" ed un foglio "SiteMobile.css", in ASP.NET MVC4 abbiamo utilizzato l'Api IsMobileDevice che altera il foglio di stile di `<body>` cambiando così in una riga di codice l'attivazione di un foglio di stile rispetto all'altro.

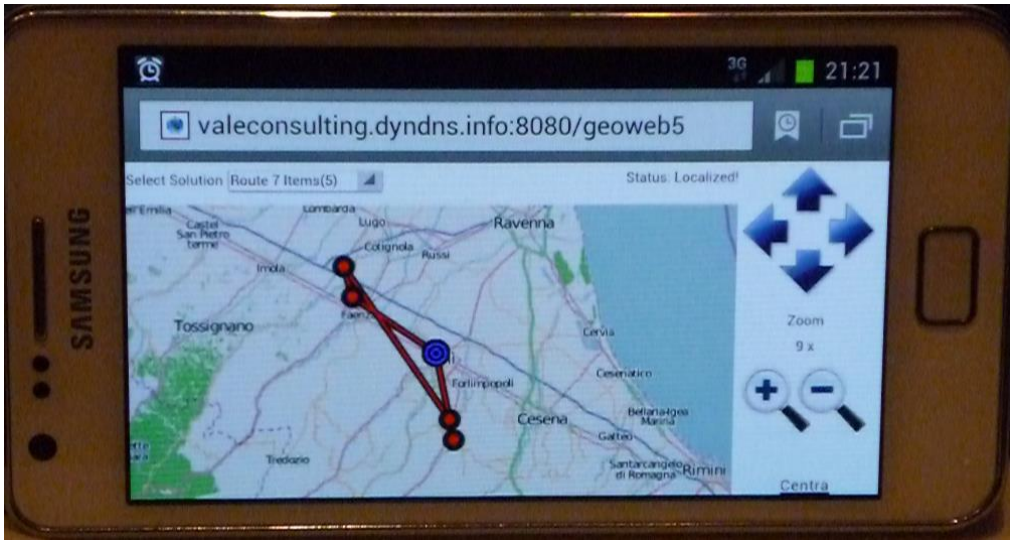
```
<body class="@(Request.Browser.IsMobileDevice ? "mobile" : "desktop")">
```

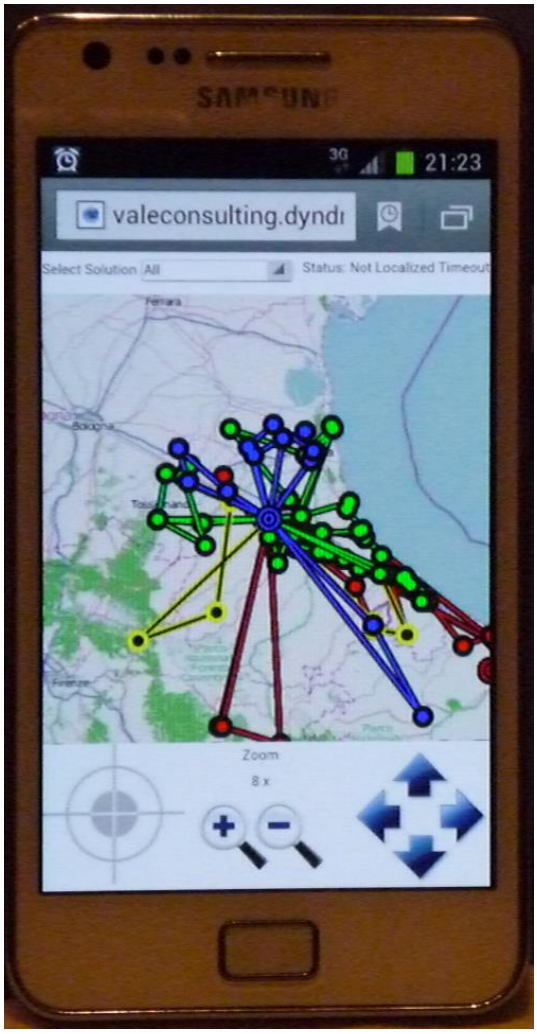
Ora si può entrare nel dettaglio della tecnologia Media Query che permette l'implementazione della tecnica "Responsive Web Design", cioè , l'applicativo è in grado di capire se il dispositivo è in orizzontale o in verticale ed è quindi in grado di adattarsi anche in fase di rotazione. Questa funzionalità intrinseca nella gestione Media Query è sito nel file "SiteMobile.css", le istruzioni partono da @media specificando le caratteristiche del dispositivo è possibile alterare l'aspetto tramite fogli di stile personalizzati, nel nostro esempio l'unica discriminante è l'orientazione del dispositivo.

```
#panelCenterText {
    display: none;
}

@media only screen and (orientation : landscape) {
    .mobile #divMap {
        position: absolute;

        top: 30px;
        left: 0px;
        bottom: 0px;
        right: 105px;
    }

. . .

    .mobile #panelCenter {
        display: none;
```

```
    }

    .mobile #panelCenterText {
        position: absolute;
        display: table-cell;

        top: auto;
        left: auto;
        bottom: 0px;
        right: 0px;

        width: 105px;

        text-align: center;
    }

}

@media only screen and (orientation : portrait) {
    .mobile #divMap {
        position: absolute;

        top: 30px;
        bottom: 105px;
        left: 0px;
        right: 0px;
    }

. . .

    .mobile #panelCenter {
        position: absolute;

        top: auto;
        left: 0px;
        right: auto;
        bottom: 0px;

        height: 105px;
        width: 105px;
    }
}
```

Nell'immagine sottostante viene riportato l'esempio di rendering su un dispositivo mobile in modalità "landscape", notiamo che il CSS abilita il div `#panelCenterText` e nasconde `#panelCenter` con l'icona del mirino, infatti si vede solo il link "Centra" in basso a destra. Nell'immagine inoltre è stata selezionata una sola soluzione.





Nell'immagine a fianco viene riportato l'esempio di rendering su un dispositivo mobile in modalità "portrait", notiamo che il CSS questa volta ridispone i singoli div in modo diverso, questa volta vediamo l'immagine del mirino che centra il centro dell'immagine sul punto geolocalizzato, oppure, in assenza di localizzazione centra il magazzino centrale.

Si è cercato anche uno sforzo nel rendere l'applicativo ergonomico, cioè, la funzione di zoom e spostamento della cartina mantenerli sempre comodi per l'utilizzo con dito pollice; dare quindi la possibilità di utilizzare comodamente l'applicativo con una sola mano.

Client Side – Business Logic e Web Design

In questo capitolo si esporrà la parte di applicativo implementata lato client con JQuery, in particolare nella pagina vengono caricati e attivati i file JavaScript che implementeranno le funzioni viste precedentemente.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta charset="utf-8" />
<title>Geoweb5 - Progetto e realizzazione di un front-end cartografico web e
mobile basato su html5 per servizi di logistica distributiva</title>
<link href="/GeoWeb5/favicon.ico" rel="shortcut icon" type="image/x-icon" />
<meta name="viewport" id="view-port "content="width=device-width, initial-
scale=1.0, user-scalable=0, minimum-scale=1, maximum-scale=1" />
<link href="/GeoWeb5/Content/Site.css" rel="stylesheet"/>
<link href="/GeoWeb5/Content/SiteMobile.css" rel="stylesheet"/>
<link href="/GeoWeb5/Content/Modal.css" rel="stylesheet"/>

<script src="/GeoWeb5/Scripts/jquery-1.7.1.js"></script>
<script src="/GeoWeb5/Scripts/modernizr-2.5.3.js"></script>

<link href="/GeoWeb5/Scripts/Plugins/JContextMenu/jquery.contextmenu.css"
rel="stylesheet"/>

<script
src="/GeoWeb5/Scripts/Plugins/JMouseWeel/jquery.mousewheel.js"></script>
<script
src="/GeoWeb5/Scripts/Plugins/JContextMenu/jquery.contextmenu.js"></script>

<script src="/GeoWeb5/Scripts/Fix/ios-orientationchange-fix.js"></script>
<script src="/GeoWeb5/Scripts/GeoWeb5/MapClass.js"></script>
<script src="/GeoWeb5/Scripts/GeoWeb5/GeoWeb5.js"></script>
</head>
```

Nella sezione `<head>` dell'Html si nota oltre ai fogli di stile precedentemente citati per la disposizione degli oggetti in desktop e mobile vediamo il caricamento dei file JavaScript (.js) tra cui il fremework JQuery, due plug-in utilizzati rispettivamente per utilizzare gestire lo zoom tramite rotella ed un plug-in per la gestione dei menu contestuali utilizzati per aggiungere/modificare soluzioni, cioè cliccando con il tasto destro del mouse su un punto per visualizzare la funzione "Add" o "Remove" sul nodo stesso.

Infine i due file che implementano il progetto "MapClass.js" che è una classe di utilità deputata alla gestione della mappa, convertire le coordinate da immagine a

coordinate e viceversa. Il file “GeoWeb5.js” che rappresenta il core dell’applicazione gestendo gli eventi lati client legati al DOM.

Si analizzerà ora step by step le fasi applicative in cui si capirà nel dettaglio come le parti precedentemente esposte si legano. Una volta caricato l’Html il browser lancia il primo evento `document.ready` e grazie al framework JQuery parte l’applicazione inizializzando le variabili ed eseguendo il codice (sarà ripotrato solo un estratto del sorgente commentando le parti principali).
Si partirà dalla struttura DOM basata su `<div>` entrando del dettaglio del codice Html e Css che servirà come base per capire il flusso degli eventi JavaScript dell’applicativo (anche in questo caso sarà riportato solo un estratto del sorgente commentando le parti principali).

```
<body class="desktop">
<div id="panelWrapper">
    . . .

    <div id="divMap">
        <canvas id="canvasMap" class="mapdefault">
            Your browser does not support the HTML5 canvas tag.
        </canvas>
    </div>

    . . .
</div>
</body>
```

```
#panelWrapper {
    position: absolute;

    top: 0px;
    left: 0px;
    bottom: 0px;
    right: 0px;
}
```

```
#divMap {
    position: absolute;
    top: 0px;
    left: 150px;
    bottom: 25px;
    right: 0px;

    width: auto;
    height: auto;

    overflow: hidden;
}
```

Da questo breve estratto si notano immediatamente le tecniche di Web Design utilizzate “div wrapping” ed il componente Html5 principale per la realizzazione di questo progetto, il tag `<canvas>`. Una pagina web può utilizzare uno o più elementi div che vengono utilizzati come involucri o contenitori. Questi div possono avere ID diversi, tra cui ”wrapper” e ”contenitore”, ma non sono limitati a questi ID, nel nostro caso l’ID è “panelWrapper”. I div sono identificati con un ID o classe in modo che le regole di stile possono essere applicate a loro tramite un foglio di stile.
Lo scopo di racchiudere un’intera pagina contenuto in un involucro è per fare sì che margini, confini, altre decisioni di presentazione possono essere applicati a tutta la pagina. Div wrapper sono utilizzati anche per creare layout con i CSS, nel nostro caso infatti vogliamo che l’intera pagina occupi tutto lo schermo del dispositivo, ed infatti usando le tecniche CSS

```
#panelWrapper {
    position: absolute;

    top: 0px;
    left: 0px;
    bottom: 0px;
    right: 0px;
}
```

Abbiamo ottenuto un contenitore che occupa esattamente lo schermo, al suo interno disponiamo un altro pannello “divMap” che conterrà la cartina vera e proria.

```
#divMap {
    position: absolute;
    top: 0px;
    left: 150px;
    bottom: 25px;
    right: 0px;

    width: auto;
    height: auto;

    overflow: hidden;
}
```

Come si può notare l’oggetto canvas su cui verrà disegnata la cartina non specifica ancora la propria posizione e dimensione, infatti, sarà la parte JavaScript una volta disegnata la pagina e dopo aver “atteso” il resize dei due div padri ad eseguire il

lavoro di richiedere al DOM l'effettiva dimensione del divMap, ridimensionare l'oggetto canvas ed infine chiedere all'Api Image la mappa.

L'aver "atteso" il resize dei due div significa che il DOM della pagina è pronto, cioè correttamente renderizzato, a questo punto viene lanciato l'evento "ready" ed intercettato dalla funzione qui sotto definita nel file "GeoWeb5.js".

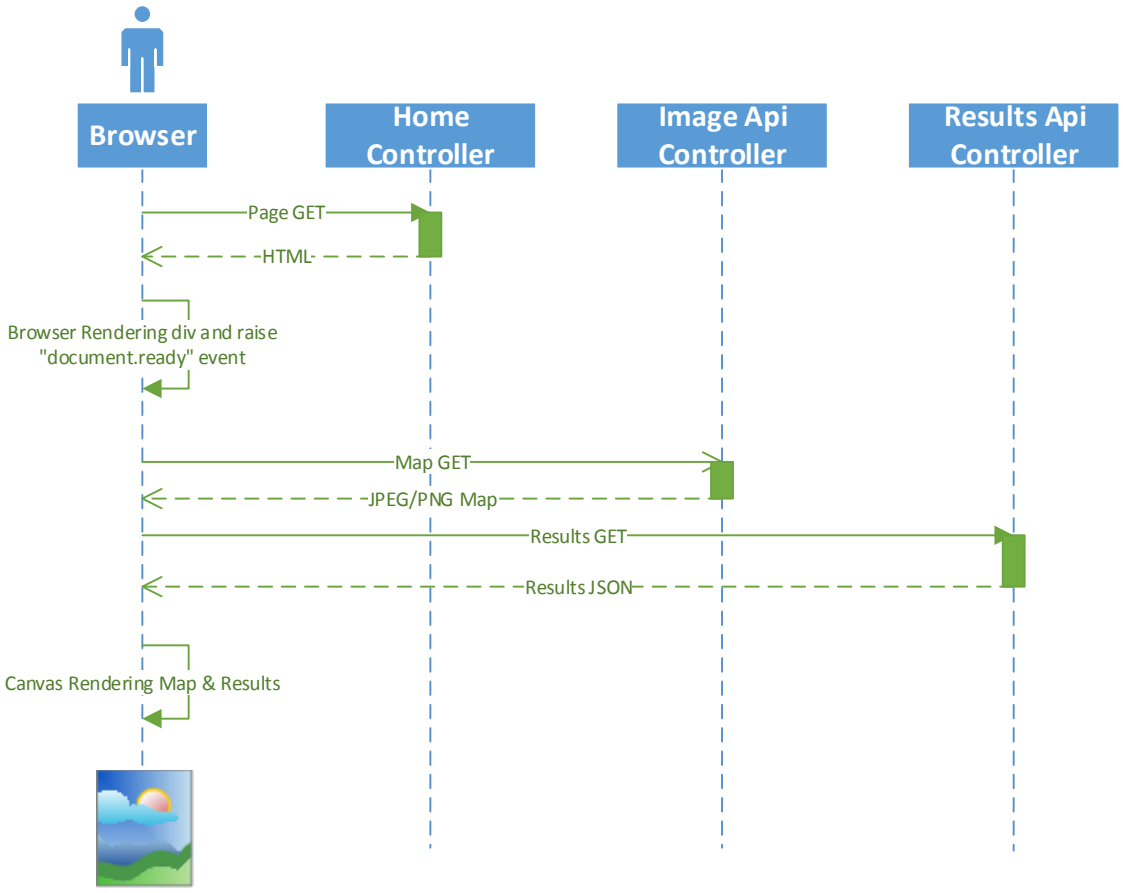
```
$('document').ready(function() {
    function init() {
        divMap = document.getElementById('divMap');
        tmpImgObj = document.getElementById('tmpImgObj');

        // Find the canvas element.
        canvas = document.getElementById('canvasMap');
        if (!canvas) {
            alert('Error: I cannot find the canvas element!');
            return;
        }

        if (!canvas.getContext) {
            alert('Error: no canvas.getContext!');
            return;
        }

        // Get the 2D canvas context.
        context = canvas.getContext('2d');
        if (!context) {
            alert('Error: failed to getContext!');
            return;
        }

        // init mapObj size
        evResize();

        window.addEventListener('resize', evResize, false);
        window.addEventListener('orientationchange', evOrientationChange, false);

        tmpImgObj.addEventListener('load', evRedrawMap, false);

        // Attach the mousemove event handler.
        canvas.addEventListener('mousedown', evMousedown, false);
        canvas.addEventListener('mouseup', evMouseup, false);
        canvas.addEventListener('mousemove', evMousemove, false);

        $(".mapCenter").bind('click', evMapCenter);
        $("#dp").bind('click', evRedPoint);
        $("#routesList").bind('change', evRouteSelected);

        $(".zoomIn").bind('click', evZoomUp);
        $(".zoomOut").bind('click', evZoomDown);
```

```
    $(".panTop").bind('click', function () { evMapMove(0, -128); });
    $(".panDown").bind('click', function () { evMapMove(0, 128); });
    $(".panLeft").bind('click', function () { evMapMove(-128, 0); });
    $(".panRight").bind('click', function () { evMapMove(128, 0); });

    $('#divMap').mousewheel(function (event, delta, deltaX, deltaY) {
        //  console.log(delta, deltaX, deltaY);
        if (deltaY > 0) evZoomUp(); else evZoomDown();
    });

    localize();
}
```

Le parti più significative di questo frammento di codice sono:
L'inizializzazione degli oggetti Html5 (canvas e context), la verifica del supporto di questi tag nel browser

- L'invocazione della funzione "evResize" il quale chiamerà "evRedrawMap" che rappresenta l'entry point di tutta la logica applicativa e sarà spiegata nel prossimo capitolo nel dettaglio;
- L'associazione degli eventi per le funzioni di pan, zoom, mouse click;
- La localizzazione, altra standardizzazione introdotta in Html5;

Prima di entrare nel dettaglio del codice riassumo in questo diagramma UML Sequence Diagram l'inizializzazione dell'applicazione in cui si vedono chiaramente le fasi di creazione.



Fasi

1. L'utente effettua una chiamata GET all'Home Controller che attiva la Vista Index.cshtml ritornando l'Html della pagina (DOM – Document Object Model)
2. Una volta terminata l'interpretazione dell'Html il browser lancia l'evento "document.ready" intercettato dall'handler definito nel file GeoWeb5.js che si incarica del resize e rendering della cartina
3. Localizzazione

Rendering sull'oggetto canvas

Questa fase introduce il core dell'applicativo, il calcolo della dimensione dell'oggetto mappa, il recupero della mappa, il disegno dei punti e delle soluzioni; essendo l'handler dell'evento document.ready implica che i "div wrapper" con posizionamento assoluto imposto dal CSS (panelWrapper e divMap) hanno già ottenuto la loro dimensione in pixel sul dispositivo in cui sta girando il browser, di conseguenza possiamo sapere ed impostare la dimensione dell'oggetto canvas in modo da riempire completamente l'oggetto divMap.

```
function evResize() {
    var divMapSize = getDivMapSize();

    evMapResize(divMapSize.Width, divMapSize.Height);
}

function getDivMapSize() {
    var divMapSize = new Object();

    divMapSize.Width = $("#divMap").width();
    divMapSize.Height = $("#divMap").height();

    return divMapSize;
}

function evMapResize(divWidth, divHeight) {
    mapObj = new MapClass(tmpImgObj, lat, lon, zoom, divWidth, divHeight);

    canvas.height = mapObj.mapHeight;
    canvas.width = mapObj.mapWidth;

    showWait(true);
    routesObjInitialized = false;
    mapObj.GetImage();
}
```

Osserviamo tralasciando momentaneamente l'oggetto mapObj (che analizzeremo in seguito) come tramite le due righe di assegnazione

```
    canvas.height = mapObj.mapHeight;
    canvas.width = mapObj.mapWidth;
```

Abbiamo fatto sì che il tag canvas occupasse tutta la superficie del pannello divMap, inoltre notiamo dopo il resize viene invocato il metodo mapObj.GetImage(), questo metodo che analizzeremo nel dettaglio successivamente fa sì che una volta effettuato

il download dell'immagine venga scatenato l'evento "evRedrawMap", il quale, si occupa di invocare l'Api Results, di interpretarne i dati e di visualizzare i punti e le linee sull'oggetto canvas in cui precedentemente è stata disegnata la cartina proveniente dall'altra chiamata all'Api Image.

```
    function evRedrawMap() {
        $("#lbl_zoom").text(zoom);

        var mapXCenter = Math.ceil(mapObj.mapHeight / 2);
        var mapYCenter = Math.ceil(mapObj.mapWidth / 2);

        var mapUrl = applicationRoot + 'api/GeoWeb5Api/Results/' + zoom + '/' +
mapXCenter + '/' + mapYCenter + '/' + mapObj.lon.toString().replace('.', ',') +
'/' + mapObj.lat.toString().replace('.', ',');

        // get row points
        $.ajax({
            url: mapUrl,
            async: false,
            dataType: 'json',
            success: drawMapInitData,
            error: function (request, ajaxOptions, thrownError) {
alert("Unexpected Error"); document.location.reload(true); }
        });
    }
```

Finalmente si è giunti al primo punto di collegamento tra la parte dell'applicativo Client scritto in JavaScript e l'ApiController, la quale avviene mediante una chiamata AJAX alla rotta dell'Api Results.

Esempio di chiamata

```
mapUrl = "/api/GeoWeb5Api/Results/9/162/265/12,031/44,232"
```

```
$.ajax({
  url: mapUrl,
  async: false,
  dataType: 'json',
  success: drawMapInitData,
  error: function (request, ajaxOptions, thrownError) { alert("Unexpected
Error"); document.location.reload(true); }
        });
```

Questa chiamata AJAX al suo interno specifica benissimo il workflow dell'applicazione; reperisce i dati dal client, esegue una chiamata GET all'ApiController specificandone la non asincronicità (`async: false`) e il tipo di formato restituito (`dataType: 'json'`), inoltre definisce due entry point in caso di successo (http status 200) alla chiamata e un altro punto in caso di errore (message box per tutti gli http status diversi da 200).

In caso di successo viene quindi invocata la funzione "drawMapInitData" a cui vengono passati i dati Json restituiti, quindi dopo una breve inizializzazione di alcune variabili a scope globale

```
    function drawMapInitData(allRoutes) {
        routesObj = allRoutes;
        pointsObj = routesObj.pointsAll;
        pointsResultObj = routesObj.pointsInRoute;
        pointsNotInResultObj = routesObj.pointsNotInRoute;

. . .

        routesObjInitialized = true;

        drawMap();
    }
```

Inizia tramite l'oggetto "context" di "canvas" il vero e proprio "draw" definitivo della rappresentazione della soluzione

```
    function drawMap() {
        context.drawImage(tmpImgObj, 0, 0);

        if (!routesObjInitialized)
            return;

        var d, i, obj, canvasCoordinate;

        for (var r = 0; r < routesObj.routes.length; r++) {
            var firstX = 0;
            var firstY = 0;

            var routeObj = routesObj.routes[r];
            var pointsSolutionObj = routeObj.SolutionPoint;

            for (d = 0; d < 2 && showRoute(routeObj.IdRoute); d++) {
                context.beginPath();
                context.strokeStyle = (d == 0) ? ((routeObj.Color == "#000000" ||
routeObj.Color == "black") ? "yellow" : "black") : routeObj.Color;
                context.lineWidth = (d == 0) ? 5 : 2;

                for (i = 0; i < pointsSolutionObj.length; i++) {
                    obj = pointsSolutionObj[i];
                    canvasCoordinate =
mapObj.Map2CanvasCoordinate(obj.LonLat.Longitude, obj.LonLat.Latitude);
```

```
                    if (i == 0) {
                        firstX = canvasCoordinate.X;
                        firstY = canvasCoordinate.Y;

                        context.moveTo(canvasCoordinate.X, canvasCoordinate.Y);
                    } else {
                        context.lineTo(canvasCoordinate.X, canvasCoordinate.Y);
                    }
                }

                context.lineTo(firstX, firstY);

                context.stroke();
                context.closePath();
            }

            for (d = 0; d < 2 && showRoute(routeObj.IdRoute); d++) {
                for (i = 0; i < pointsSolutionObj.length; i++) {
                    obj = pointsSolutionObj[i];
                    canvasCoordinate =
mapObj.Map2CanvasCoordinate(obj.LonLat.Longitude, obj.LonLat.Latitude);

                    context.beginPath();
                    context.fillStyle = (d == 0) ? ((routeObj.Color == "#000000"
|| routeObj.Color == "black") ? "yellow" : "black") : routeObj.Color;
                    var size = (d == 0) ? 8 : 4;
                    //  writeMessage(i, canvasCoordinate.X, canvasCoordinate.Y -
5);

                    context.arc(canvasCoordinate.X, canvasCoordinate.Y, size, 0,
2 * Math.PI, false);
                    context.fill();

                    context.closePath();
                }
            }

            for (d = 5; d >= 1 ; d--) {
                for (i = 0; i < pointsNotInResultObj.length; i++) {
                    obj = pointsNotInResultObj[i];
                    canvasCoordinate =
mapObj.Map2CanvasCoordinate(obj.LonLat.Longitude, obj.LonLat.Latitude);

                    context.beginPath();
                    context.fillStyle = ((d % 2) == 1) ? 'black' : 'red';

                    context.moveTo(canvasCoordinate.X, canvasCoordinate.Y);
                    context.arc(canvasCoordinate.X, canvasCoordinate.Y, d * 2, 0,
2 * Math.PI, false);
                    context.fill();

                    context.closePath();
                }

                for (i = 0; i < pointsObj.length; i++) {
                    obj = pointsObj[i];
                    if (obj.IdNodo == warehouseId) {
                        canvasCoordinate =
mapObj.Map2CanvasCoordinate(obj.LonLat.Longitude, obj.LonLat.Latitude);
```

```
                        context.beginPath();
                        context.fillStyle = ((d % 2) == 1) ? 'black' : 'blue';

                        context.moveTo(canvasCoordinate.X, canvasCoordinate.Y);
                        context.arc(canvasCoordinate.X, canvasCoordinate.Y, d *
2, 0, 2 * Math.PI, false);
                        context.fill();

                        context.closePath();
                    }
                }
            }
        }
```

Da questo breve scorcio di codice notiamo immediatamente come:

- L'oggetto serializzato Json sia utilizzabile direttamente da JavaScript come un oggetto nativo
    - `routesObj.routes.length`
    - `obj = pointsSolutionObj[i];`
    - `obj.LonLat.Longitude e obj.LonLat.Latitude`
- L'oggetto canvas ci fornisca primitive per il disegno paragonabili alle primitive delle normali applicazioni desktop
    - `context.moveTo(canvasCoordinate.X, canvasCoordinate.Y);`
    - `context.lineTo(canvasCoordinate.X, canvasCoordinate.Y);`
    - `context.arc(canvasCoordinate.X, canvasCoordinate.Y, d * 2, 0, 2 * Math.PI, false);`
    - `context.fill();`
    - ecc...

Ed infine si affronta il punto della localizzazione, in realtà Html5 fornisce un'Api che ritorna le coordinate recuperate tramite IP (bassa precisione) o raccolte direttamente dal dispositivo tramite GPS.

```
    function localize() {
        if (localizationTime == 0) {
            $("#lbl_localization").text("Waiting...");

            localizationTime = 1;

            navigator.geolocation.getCurrentPosition(
                gotPosition,
                errorGettingPosition,
                { 'enableHighAccuracy': true, 'timeout': 30000, 'maximumAge': 0 }
            );
        }else {
            localizationTime = (localizationTime + 1) % (5 * 4); /* 5 significa
5per2000 cioe 10 secondi*/
```

```
    }

    if (localized && mapStatus != "mapmove") {
        drawMap();
        evRedPoint();
    }

    setTimeout(function () { localize(); }, 2000);
}

function gotPosition(pos) {
    $("#txt_xmap").val(pos.coords.longitude);
    $("#txt_ymap").val(pos.coords.latitude);

    $("#lbl_localization").text("Localized!");

    localized = true;
}

function errorGettingPosition(err) {
    var errorString;
    if(err.code == 1) {
        errorString = "Autorization Denied";
    } else if(err.code == 2) {
        errorString = "Position Unavaiable";
    } else if(err.code == 3) {
        errorString = "Timeout";
    } else {
        errorString = err.message;
    }

    $("#lbl_localization").text("Not Localized " + errorString);

    $("#txt_xmap").val(lon_original);
    $("#txt_ymap").val(lat_original);

    localized = false;
}
```

Il lavorò si è quindi ridotto ad aggiornare la posizione tramite un timer di JavaScript, ad interpretare le Coordinate dal sistema geografico alla nostra cartina mediante l'oggetto MapClass che vedremo successivamente.

L'oggetto MapClass

La classe JavaScript MapClass (MapClass.js) è l'oggetto deputato a recuperare l'immagine dall'Api Image ed in grado di convertire i pixel del canvas in coordinate e le coordinate in pixel dell'oggetto canvas. Prima ci eravamo lasciati con il metodo mapObj.GetImage()

Questo metodo parte dalla configurazione di un template per l'invocazione dell'Api Image

```
<setting name="imgUrl" serializeAs="String">
<value>/ImageApi/GetImage?lat={lat}&amp;lon={lon}&amp;zoom={zoom
}&amp;mapWidth={width}&amp;mapHeight={height}</value>
</setting>
```

Sostituisce i placeholder con le informazioni di runtime ed esegue la chiamata dirottandola su un oggetto temporaneo del DOM `tempImage` rappresentato da un normale tag `<img>`

```
function getImage() {
    var urlImage = imgTemplate;

    urlImage = urlImage.replace("{lat}", this.lat);
    urlImage = urlImage.replace("{lon}", this.lon);
    urlImage = urlImage.replace("{zoom}", this.zoom);

    urlImage = urlImage.replace("{width}", this.mapWidth);
    urlImage = urlImage.replace("{height}", this.mapHeight);

    this.tempImage.src = urlImage;
}
```

A questo oggetto del DOM ricordo essere associato un evento al termine del load, cioè al termine del download dell'immagine

```
tmpImgObj.addEventListener('load', evRedrawMap, false);
```

Viene attivato il processo precedentemente spiegato di recupero dei dati delle soluzioni in formato Json e attivato il processo di disegno, tutto lato Client.

L'alto aspetto fondamentale di questa classe di utilità sono i metodi per trasformare le coordinate geo in pixel e viceversa facendo attenzione alla gestione dell'asse "y" che nell'oggetto div parte da 0 in alto e cresce verso il basso mentre la latitudine parte dal basso crescendo verso l'alto.

```
    function canvas2MapCoordinate(x, y) {
        var layerCoordinate = new Object();

        // fix Y
        var nY = this.mapHeight - y;

        var xFromCenter = x - Math.ceil(this.mapWidth / 2);
        var yFromCenter = nY - Math.ceil(this.mapHeight / 2);

        layerCoordinate.Lon = new Number(this.lon + (xFromCenter *
this.x2lon)).toFixed(6);
        layerCoordinate.Lat = new Number(this.lat + (yFromCenter *
this.y2lat)).toFixed(6);

        return layerCoordinate;
    }

    //  Converte la coordinata Mappa nella coordinata del Canvas
    function map2CanvasCoordinate(xlon, ylat) {
        var layerCoordinate = new Object();

        var lonFromCenter = xlon - this.lon;
        var latFromCenter = this.lat - ylat;

        layerCoordinate.X = Math.ceil(this.xCenter + (lonFromCenter /
this.x2lon));
        layerCoordinate.Y = Math.ceil(this.yCenter + (latFromCenter /
this.y2lat));

        return layerCoordinate;
    }

    function lon2tile(lon) {
         return (Math.floor((lon+180)/360*Math.pow(2, this.zoom)));
    }

    function lat2tile(lat) {
        return (Math.floor((1-Math.log(Math.tan(lat*Math.PI/180) +
1/Math.cos(lat*Math.PI/180))/Math.PI)/2 *Math.pow(2,this.zoom)));
    }

    function tile2lon(x) {
        return (x / Math.pow(2, this.zoom) * 360 - 180);
    }

    function tile2lat(y) {
        var n = Math.PI - 2 * Math.PI * y / Math.pow(2, this.zoom);
        return (180 / Math.PI * Math.atan(0.5 * (Math.exp(n) - Math.exp(-n))));
    }
```

Concludendo questa classe attiva la chiamata per ottenere l'immagine corrente comunicando la dimensione desiderata dell'immagine, la posizione corrente espressa il latitudine e longitudine ed il livello di zoom; è anche in grado e viene utilizzata sull'evento mouse move di tradurre la posizione sullo schermo in coordinate.

Performance Optimization

A livello di ottimizzazione performance quasi tutto è già stato detto parlando di Cache e dei moduli managed per la gestione tramite attributi e file di configurazione.

Meritano però una citazione:

- Il concetto di Bundle delle risorse JavaScript e CSS ed in particolare il meccanismo messo a disposizione da ASP.NET MVC 4 per la compressione di questi oggetti (fornendo anche una sorta di protezione al diritto d'autore rendendo il codice non interpretabile), in fase di startup dell'applicazione vengono definiti i cluster di risorse tra loro omogenee ed associati ad un alias, ad esempio:

```
bundles.Add(new
ScriptBundle("~/bundles/geoweb5").Include("~/Scripts/GeoWeb5/MapClass.js",
"~/Scripts/GeoWeb5/GeoWeb5.js"));
```

  E utilizzati nella Vista cshtml in questo modo

```
@Scripts.Render("~/bundles/geoweb5")
```

  Questo meccanismo ha il duplice vantaggio di eseguire una sola richiesta al server e di minimizzare la dimensione dei JavaScript, ad esempio otteniamo:

```
<script
src="/geoweb5/bundles/geoweb5?v=yIvPg1cqhp4qCNmmus8ZD5Dr30ot7xZIFjFV2PYNyO
Q1"></script>
```

  Dove il parametro "v=" è l'hash dell'unione dei file facenti parte del Bundle. Analogamente per i Css è stata adottata la stessa tecnica.

- (Content Delivery Network) Un sistema di distribuzione su Internet che accelera la consegna di pagine Web, audio, video e altro contenuto basato su Internet per gli utenti di tutto il mondo. Il CDN replica file del fornitore di contenuti in server, chiamati "server di caching" o "server di bordo," che

si trovano nei data center geograficamente distanti. La maggior parte delle CDN sono servizi di terze parti, tuttavia, le grandi imprese possono sviluppare la propria "enterprise CDN" (eCDN) per supportare postazioni remote in modo più efficace. Quando il contenuto viene replicato in tutto il paese o il mondo, viene consegnato agli utenti con una maggiore velocità e affidabilità. La CDN veicola la richiesta dell'utente per contenuto al server di caching appropriato in base alla posizione dell'utente. Le CDN sono spesso collegati a più backbone ISP offrendo un'elevata disponibilità per gli utenti.

Funzionalità

Precedentemente sono state esposte tutte le funzionalità dell'applicativo, questa sezione si limiterà a fare un breve riassunto:

1. Applicativo cross-browser e cross-device;
2. Visualizzazione parziale delle soluzioni;
3. Spostamento della cartina utilizzando le frecce blu dell'interfaccia o il pulsante sinistro del mouse;
4. Zoom della cartina utilizzando le icone a lente di ingrandimento o utilizzando la rotella del mouse;
5. Localizzazione e possibilità di centrare la cartina sulla posizione corrente;
6. Alterare le soluzioni (solo versione desktop)
    a. Rimuovere un punto da una soluzione (tasto destro del mouse su un punto);
    b. Assegnare un punto ad una soluzione diversa (utilizzando il tasto sinistro del mouse e trascinando il punto da spostare sul punto di destinazione);
    c. Creare una nuova soluzione partendo da un punto non associato (tasto destro del mouse su un punto libero rosso/nero);
7. Informazioni sul nodo (solo versione desktop)
    a. Identificativo del nodo su cui si trova il mouse;
    b. Latitudine e Longitudine del punto su cui si trova il mouse;
8. Rappresentazione punti significativi
    a. Stato della geolocalizzazione (riuscita o no) ed in caso di successo rappresentazione sulla cartina (nodo bianco/nero);

b. Rappresentazione del magazzino centrale (nodo blu/nero);
c. Rappresentazione dei nodi non associati a soluzioni (nodo rosso/nero);

Conclusioni

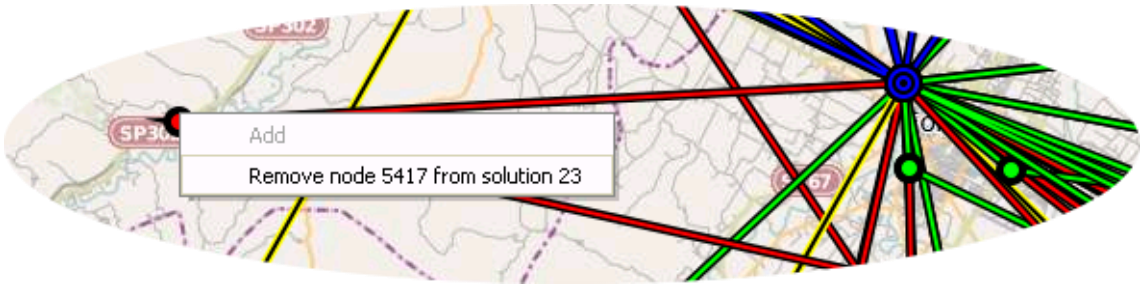
Nei precedenti capitoli si è analizzato le tecnologie utilizzate ed i pattern architetturali su cui si basa il progetto "GeoWeb5".

GeoWeb5 è quindi un applicativo web basato su standard che permettano la visualizzazione tramite un normale PC oppure tramite un dispositivo mobile; per venire incontro a diverse esigenze; ad esempio ricordiamo l'utente amministrativo, il quale, a seguito di imprevisti, dall'ufficio, deve poter alterare le soluzioni di ottimizzazione calcolate basatosi su dati ottimi, quindi dal browser del desktop l'applicativo può utilizzando esclusivamente il mouse per assegnare, spostare un punto da una soluzione all'altra, cioè può alterare la soluzione precedentemente calcolata

"On Drag", quando il mouse si trova su un punto non incluso in una soluzione (rosso/nero), oppure su un punto di una soluzione alterabile, l'icona del puntatore del mouse diventa come nell'immagine a sinistra, il che significa che tenendo premuto il pulsante sinistro del mouse e trascinando su un punto appartenente ad un'altra soluzione l'icona cambierà ed attiverà la funzione

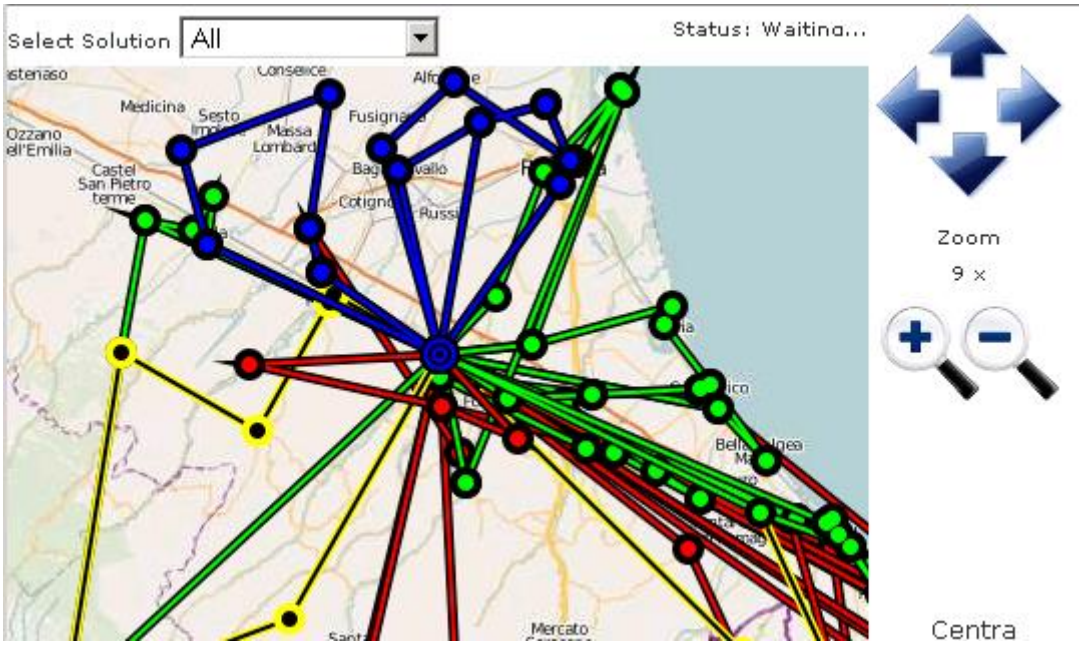
"On Drop"; in questo stato l'icona cambia e si presenta come l'immagine di sinistra, a questo punto se rilasciato il tasto sinistro del mouse in questo stato apparirà una Message Box di conferma alterazione, rispondendo positivamente avvera l'effettiva associazione del punto di origine a seguito del punto di destinazione appartenente ad un'altra soluzione

Può inoltre tramite l'utilizzo del mouse (rotella) effettuare zoom e spostare la cartina sempre con l'utilizzo del pulsante sinistro; con il pulsante destro, invece, si possono rimuovere punti da una soluzione o crearne di nuove in quanto la pressione di questo pulsante abilita un menu contestuale html per l'effettiva scelta di queste funzionalità .

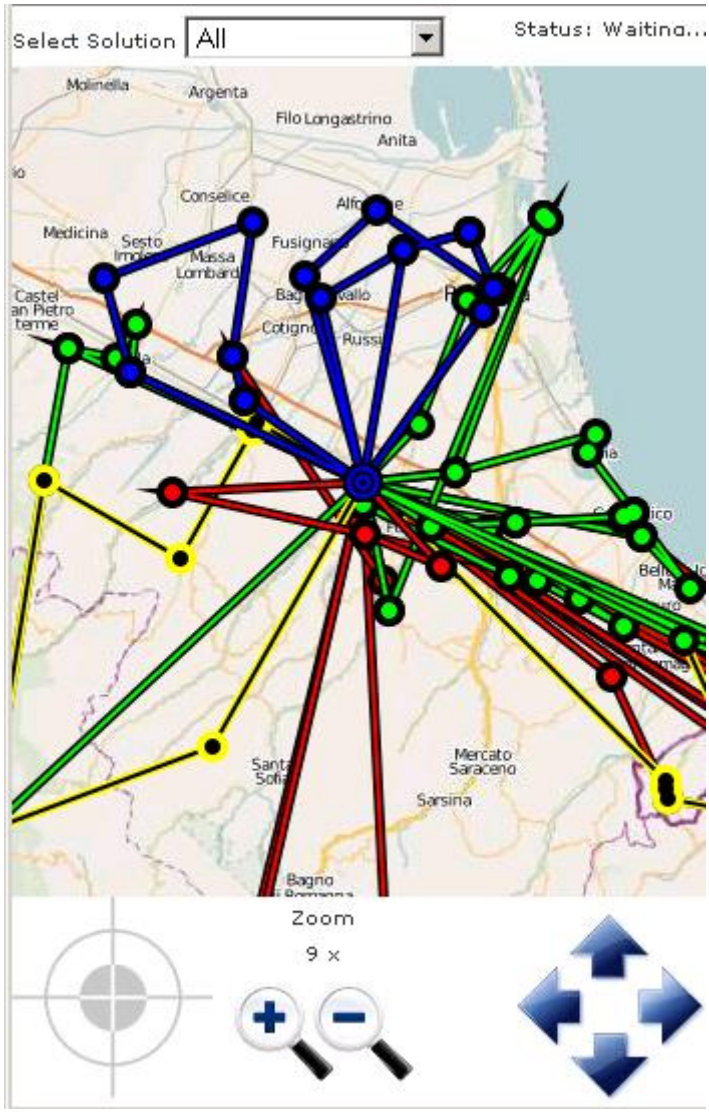


Purtroppo l'utente incaricato di eseguire le consegne avendo sul proprio mezzo a disposizione solo un dispositivo mobile, quindi senza mouse, deve poter conoscere le proprie missioni ed eventualmente visualizzare le modifiche apportate dall'utente amministrativo; con questa soluzione non può utilizzare l'interfaccia di amministrazione; per questi dispositivi (Android, iPhone, Windows) verrà presentata un'interfaccia ottimizzata all'utilizzo delle funzioni base quali zoom e spostamento cartina tramite bottoni, interessante inoltre notare l'approccio RWD (responsive web design) per una visualizzazione ottimizzata del dispositivo quando posto in verticale e quando posto in orizzontale.

Grafica mobile in orizzontale



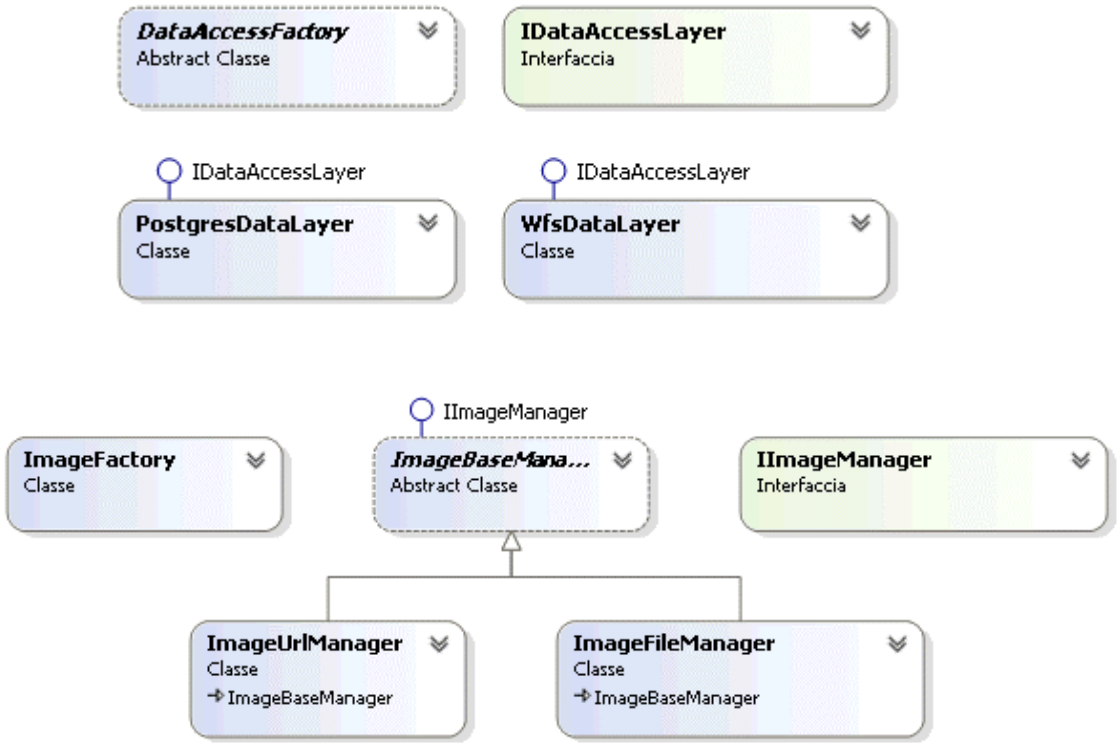
Grafica mobile in verticale



Per realizzare questo progetto web oriented, ci siamo basati su framework in grado di gestire dati geospaziali e abbiamo creato un'architettura ed una infrastruttura che ci ha permesso di legare queste entità fornendo un pacchetto completo e altamente integrato ed integrabile.

Il cuore del progetto consiste nel Middle Tier, cioè la parte applicativa lato server, il fornitore dell'Html, il fornitore di servizi e le Api che si occupa di interagire con il dominio dei dati, in particolare ci soffermeremo su:

- ASP.NET MVC4
  - Home page, utilizzando ASP.NET MVC4 standard basato sul concetto di Controller Home, Model Home e View (Index.cshtml) scritta con il Razor View Engine e con l'utilizzo degli helper per scrivere Html5 (`@Html`);
  - Api Image, utilizzando ASP.NET MVC4 ma utilizzando un'" Action Result" custom che non attiva una vista Razor ma scrive direttamente sulla Respose http l'immagine calcolata unendo i tile recuperati da diverse fonti;
  - Api Results, utilizzando il nuovo tipo di Controller, ApiController di ASP.NET MVC4 che si occupa in autonomia di serializzare l'oggetto del Model in Json o Xml, a seconda della richiesta del client;
- Pattern Utilizzati
  - Particolare attenzione al pattern creazionale Abstract Factory che permette una estendibilità dei Data Provider (Image e Results)



- Architettura SOA
    - Ricordando che ogni componente dell'applicativo opera come un componente isolato che comunica i dati tramite contratti prestabiliti
        - Tra Middle Tier e Data Layer, via WFS e protocollo SOAP basato su Xml su http;
        - Tra Presentation Layer e Middle Tier, via Ajax e protocollo Json su http;

In conclusione questo progetto ha utilizzato le tecnologie attualmente disponibili per creare vere e proprie applicazioni web, tecnologie che in questi ultimi 15 anni si sono particolarmente evolute per rendere più omogeneo lo sviluppo di due aspetti fondamentali delle applicazioni web, la parte di web design e la parte server side, infatti, nella programmazione web questi due aspetti spesso si sovrappongono pur provenendo da storie completamente diverse.

Utilizzando queste tecnologie e alcune tecniche di programmazione universalmente riconosciute (design pattern) si è fornito un applicativo performante, estendibile e basato su standard.

Bibliografia

Dino Esposito (2006), Programming Microsoft ASP.NET 2.0 Core Reference, Microsoft Press

Dino Esposito (2006), Programming Microsoft ASP.NET 2.0 Applications: Advanced Topics, Microsoft Press

David Sceppa (2006), Programming Microsoft ADO.NET 2.0 Core Reference, Microsoft Press

Kenn Scribner & Scott Seely (2009), Effective REST Service via .NET, Addison-Wesley

Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley

Jimmy Nulsson (2010), Applying Domain-Driven Design and Patterns, Addison-Wesley

Christopher Schimitt, Mark Trammel, Ethan Marcotte, Dunstan Orchard & Todd Dominey (2005), Professional CSS Cascading Style Sheets for Web Design, Wrox

Steve Fulton & Jeff Fulton (2011), HTML5 Canvas Native Interactivity and Animation for the Web, O'Reilly

Matthew MacDonald (2011), HTML5: The Missing Manual, O'Reilly

Adam Freeman (2012), Pro jQuery, Apress

Jon Galloway, Phil Haack, Brad Wilson & K. Scott Allen (2012), Professional ASP.NET MVC4, Wrox

Jeffrey Palermo, Jimmy Bogard, Eric Hexter, Matthew Hinze & Jeremy Skinner (2012), ASP.NET MVC4 in Action, Manning

http://www.w3.org/html/wg/drafts/html/master/Overview.html

http://www.w3schools.com/html/html5_intro.asp

http://www.w3.org/Style/CSS/current-work

http://www.asp.net/mvc

http://www.asp.net/web-api

http://www.iis.net/

http://www.oracle.com/it/products/database/options/spatial/index.html

http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?casestudyid=710000000547

http://www.postgresql.org/

http://postgis.net/

http://npgsql.projects.pgfoundry.org/

http://geoserver.org/display/GEOS/Welcome

http://www1.isti.cnr.it/~polini/downloads/WS_Unicam.pdf

http://msdn.microsoft.com/it-it/library/cc185081.aspx

http://wiki.openstreetmap.org/wiki/Mapnik

http://www.ehow.com/how_5145511_use-wrapper-div.html

http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

http://www.pcmag.com/encyclopedia_term/0,1237,t=CDN&i=39466,00.asp

http://www.microsoft.com/maps/

http://msdn.microsoft.com/en-us/library/cc749633(v=sql.100).aspx

http://alastaira.wordpress.com/2011/07/06/converting-tms-tile-coordinates-to-googlebingosm-tile-coordinates/

http://blogs.microsoft.co.il/blogs/shair/archive/2012/06/24/how-to-bing-maps-custom-tile-overlay-google-maps.aspx