UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Premar

Water Forecast and Resampling Software

Sviluppo software per la previsione a breve termine delle anomalie di livello del mare a Venezia con tecniche di data streaming.

LAUREANDO: ENRICO EUGENIO

RELATORE: PROF. SERGIO CONGIU

ANNO ACCADEMICO 2011/2012

... a Lino ...

Indice

1 Introduzione

1.1 L'azienda

Il progetto mi è stato assegnato da Thetis S.p.a., una società di ingegneria e di servizi ambientali, con sede a Venezia.

Thetis opera nello sviluppo e nel management di progetti e applicazioni tecnologiche innovative in campo dell'Ambiente e Territorio, dell'Ingegneria Civile, Energia e Impianti, della Direzione Lavori, dell'ITS e Sistemi Integrati, del Centro Sistemi di Previsioni e Modelli, dell'Unione Europea e R&D (Ricerca e Sviluppo).

Fornisce i propri servizi ad un ampio spettro di clienti locali, nazionali e internazionali in diversi paesi come ad esempio Cina e India, operando dall'Arsenale di Venezia, del quale sta curando ristrutturazione e rigenerazione economica. Le competenze della società sono state costruite sul campo attraverso le attività di salvaguardia del delicato ecosistema lagunare veneziano sviluppate in oltre un decennio.

FIGURA 1 LOGO DI THETIS

Costituita nel 1993, ha un capitale di 11 milioni di euro e annovera tra i propri soci soggetti sia privati sia pubblici. La società, con un fatturato di circa 27 milioni di euro, impiega 150 persone altamente qualificate, con forte presenza di donne. L'investimento in ricerca e sviluppo rappresenta circa il 9,5% dei ricavi.

1.2 Il Progetto

Sviluppo software per la previsione a breve termine delle anomalie di livello del mare a Venezia con tecniche di data streaming. L'applicazione, sviluppata in linguaggio Java, dovrà fare una stima sul breve periodo del contributo meteo medio sulla marea totale tramite l'elaborazione dei dati storici presi dal database e in tempo reale dalle stazioni di rilevamento poste nel nord Adriatico.

Il progetto, sviluppato durante il tirocinio della durata di 250 ore iniziato il 2 novembre 2011 e concluso il 27 gennaio 2012, è stato svolto nella nel settore Previsione e Modelli di Thetis, e servirà come modello di previsione per la realizzazione del progetto principale, il MOSE.

1.3 Il MOSE

Il **MO.S.E.** (acronimo di **MO**dulo **S**perimentale **E**lettromeccanico) consistente in un sistema integrato di opere di difesa costituito da schiere di paratoie mobili a scomparsa in grado di isolare la laguna di Venezia dal Mare Adriatico durante gli eventi di alta marea superiori a una quota concordata (110 cm) e fino a un massimo di 3 metri. Insieme ad altre opere complementari come il rafforzamento dei litorali, il rialzo di rive e pavimentazioni e la riqualificazione della laguna, dovrebbero permettere la difesa della città di Venezia da eventi estremi come le alluvioni e come il degrado morfologico, per il quale la laguna sta progressivamente cedendo al mare e il livello del suolo si sta abbassando. L'opera è stata avviata nel 2003 alle tre bocche di porto del Lido, di Malamocco e di Chioggia, i varchi che collegano la laguna con il mare e attraverso i quali si svolge il flusso e riflusso della marea. Il fenomeno dei picchi di marea pronunciati si verificano con periodicità nell'Adriatico settentrionale e con particolare intensità nella laguna di Venezia, essi provocano allagamenti nelle aree urbane di Venezia e Chioggia e, molto più raramente, di Grado e Trieste. Il fenomeno è frequente soprattutto nel periodo autunnale-primaverile, quando si combina con i venti di scirocco, che, spirando dal canale d'Otranto lungo tutta la lunghezza del bacino marino, impediscono il regolare deflusso delle acque, o di bora, che ostacolano invece localmente il deflusso delle lagune e dei fiumi del litorale veneto.

In futuro il fenomeno delle acque alte potrebbe aggravarsi per il previsto aumento del livello del mare come effetto dei cambiamenti climatici. Rispetto a questo problema, il Mose (insieme al rinforzo del cordone litoraneo) è stato progettato, secondo un criterio precauzionale, per fronteggiare l'eustatismo fino a 60 cm, cioè superiore anche alle recentissime stime dell'IPCC (Intergovernmental Panel on Climate Change), che prevedono un innalzamento del mare nei prossimi 100 anni in un range compreso tra 18 e 59 cm. Grazie alla flessibilità di gestione, il Mose può far fronte a un aumento delle acque alte in modi diversi in base alle caratteristiche e all'entità dell'evento di marea. Le strategie di difesa possono prevedere sia la chiusura contemporanea di tutte e tre le bocche di porto, in caso di evento eccezionale, sia, in alternativa e a seconda dei venti, della pressione e dell'entità di marea prevista, anche la chiusura differenziata delle bocche di porto o, ancora, chiusure solo parziali di ciascuna bocca, essendo le paratoie indipendenti l'una dall'altra.

Da un punto di vista tecnico per la città di Venezia esistono definizioni rigorose, basate sui livelli di marea osservati alla stazione idrografica di Punta della Salute:

- *marea sostenuta* quando il livello di marea è compreso tra +80 cm e +109 cm sullo zero mareografico (definito come il livello medio del mare misurato nel 1897);
- *marea molto sostenuta* quando il valore è compreso tra +110 cm e +139 cm;
- *acqua alta eccezionale* quando il valore raggiunge o supera i +140 cm.

Il livello di marea è determinato da due contributi:

- La *marea astronomica*, dipendente dal moto degli astri, principalmente la Luna e in proporzione minore il Sole e via via tutti gli altri corpi celesti, e dalla geometria del bacino. E' causata dalle forze gravitazionali e centrifughe esistenti nei sistemi terra-luna e terra-sole in rotazione attorno ai rispettivi centri di massa. Il contributo di questi fattori è soggetto a pochissime incertezze ed è regolato da leggi di meccanica fisica.
- Il *contributo meteorologico*, che dipende da moltissimi fattori variabili, quali direzione e intensità dei venti, campi barici, precipitazioni eccetera, tutti legati da relazioni complesse e regolati da leggi fisiche di tipo probabilistico, per cui le previsioni possono essere elaborate solamente tramite modelli statistici, a pochi giorni di distanza e con un'approssimazione crescente con l'anticipo della previsione.

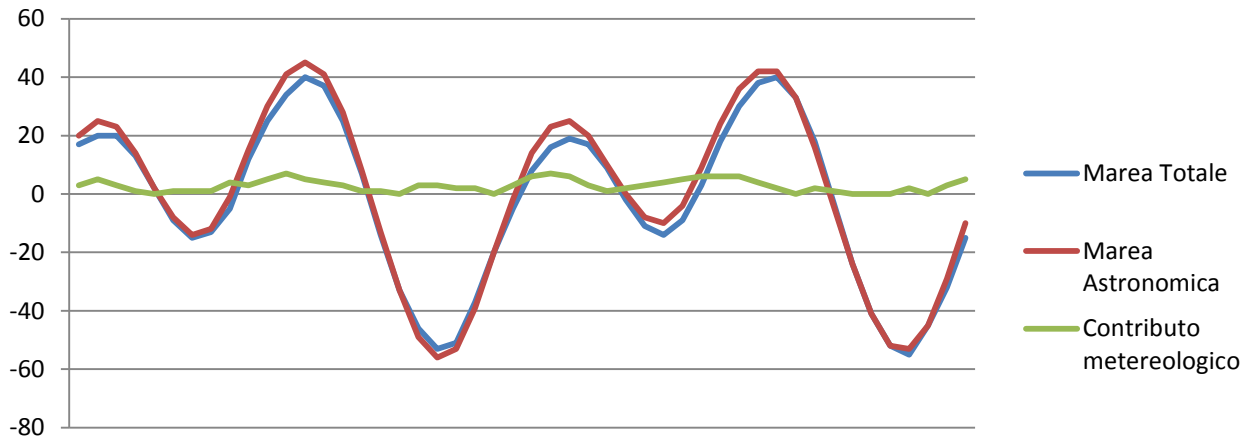
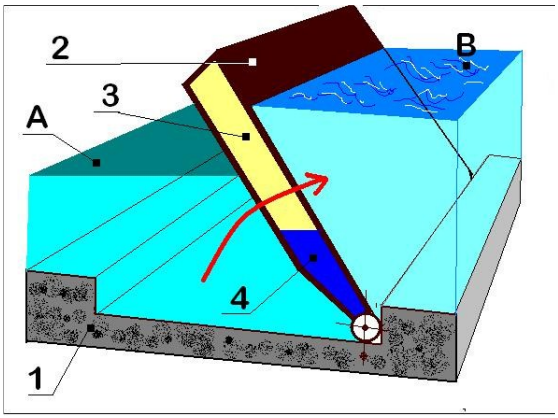


FIGURA 2 ESEMPIO DI GRAFICO DI MAREA E CONTRIBUTO METEO

Funzionamento

Il MOSE consiste in un sistema di paratoie a ventola a spinta di galleggiamento oscillante a scomparsa che modificano lo scambio idrico mare-laguna per non nuocere alla morfologia e alla qualità delle acque, non ostacolare la navigazione e quindi interferire con le attività portuali e di pesca e non alterare il paesaggio. Le paratoie, consistenti in strutture scatolari metalliche larghe 20 metri, lunghe tra i 20 e i 30 metri e spesse circa 5 metri, sono definite "mobili a gravità", in condizioni normali sono piene d'acqua e rimangono adagiate sul fondo, dove sono alloggiate in cassoni prefabbricati in calcestruzzo che poggiano sul fondale. Quando è prevista una marea superiore ai 110 cm, vengono svuotate dall'acqua mediante l'immissione di aria compressa; in questo modo esse si sollevano, ruotando sull'asse delle cerniere, fino a emergere isolando temporaneamente la laguna dal resto del mare, fermando il flusso della marea. Il tempo di sollevamento delle paratoie è di circa 30 minuti, mentre la fase di rientro è di circa 15 minuti.

La quota di 110 cm oltre la quale le barriere entreranno in funzione è stata concordata dagli enti competenti come ottimale rispetto all'attuale livello del mare. Il MOSE, essendo un sistema flessibile, può comunque essere utilizzato anche per acque alte inferiori alla quota stabilita. Il sistema inoltre, a seconda dei venti, della pressione, dell'entità della marea, può essere gestito con modalità differenti, quindi anche con chiusure differenziate delle bocche di porto o chiusure parziali di ciascuna bocca.



SCHEMA CHE MOSTRA IL FUNZIONAMENTO DEL MOSE.
A: LAGUNA
B: MARE APERTO
1: BASAMENTO
2: PARATOIA
3: ARIA COMPRESSA
4: ACQUA ESPULSA

Per la difesa delle tre bocche di porto sono previste complessivamente 78 paratoie, divise in 4 schiere: due schiere di 21 e 20 paratoie alla bocca di porto di Lido-San Nicolò, la più ampia, collegate da un'isola artificiale; una schiera di 19 paratoie alla bocca del porto di Malamocco e una di 18 alla bocca del porto di Chioggia. Per assicurare la navigazione anche nei periodi in cui le barriere mobili sono in funzione, sono stati realizzati porti rifugio e conche di navigazione che consentiranno il transito delle imbarcazioni.

1.4 Strumenti utilizzati

Per lo sviluppo è stato utilizzata la piattaforma Java con l'ausilio dell'IDE NetBeans, un ambiente di sviluppo integrato multilinguaggio e multipiattaforma dotato fra l'altro di strumenti di evidenziazione, compilazione e debug integrati.
Per poter sviluppare un applicativo in Java sono necessari due strumenti principali:

- Software Development Kit: il pacchetto per la compilazione dei file sorgenti
- Java Runtime Environment: cioé ambiente di esecuzione per applicazioni, il quale contiene la Java Virtual Machine, le librerie standard e un launcher per le applicazioni Java

Per l'analisi dei dati e il plottaggio è stato utilizzato Microsoft Excel 2010, il database dal quale sono stati ottenuti i vari dati delle stazioni di rilevamento utilizza come DBMS Microsoft SQL Server al quale il software si è interfacciato tramite apposite librerie.

Java

Java è stato creato a partire da ricerche effettuate alla Stanford University agli inizi degli anni Novanta. Nel 1992 nasce il linguaggio Java, prodotto da Sun Microsystems e realizzato da un gruppo di esperti sviluppatori capitanati da James Gosling.

FIGURA 2 LOGO DI JAVA

Per facilitare il passaggio a Java ai programmatori old-fashioned, legati in particolare a linguaggi come il C++, la sintassi di base è stata mantenuta pressoché identica a quella del C++. Per le caratteristiche object-oriented del linguaggio ci si è ispirati al C++ e soprattutto all'Objective C.

In un primo momento Sun decise di destinare questo nuovo prodotto alla creazione di applicazioni complesse per piccoli dispositivi elettronici; fu solo nel 1993 con l'esplosione di internet che Java iniziò a farsi notare come strumento per iniziare a programmare per internet.

I programmi scritti in linguaggio Java sono destinati all'esecuzione sulla piattaforma Java, ovvero saranno lanciati su una Java Virtual Machine e, a tempo di esecuzione, avranno accesso alle API della libreria standard. Ciò fornisce un livello di astrazione che permette alle applicazioni di essere interamente indipendenti dal sistema su cui esse saranno eseguite.
La piattaforma Java è composta da due blocchi: la macchina virtuale Java (Java Virtual Machine o JVM) e le API Java.

La macchina virtuale è la base della piattaforma Java, mentre le API sono una collezione di componenti software (librerie) già scritti e pronti all'uso.

Perché una applicazione software possa girare su una piattaforma Java, essa deve essere scritta nel linguaggio di programmazione Java, deve essere quindi compilata, fornendo come prodotto il cosiddetto bytecode dell'applicazione; il bytecode verrà poi interpretato dalla macchina virtuale e quindi eseguito.

Il prodotto della fase 2 è costituito da file contenenti le istruzioni che compongono il programma. Il linguaggio in cui queste istruzioni sono espresse non è specifico di alcuna macchina o sistema operativo particolare; al contrario, viene utilizzato un linguaggio appositamente progettato per essere il più possibile 'neutro' e, quindi, indipendente dal sistema su cui verrà effettivamente eseguita l'applicazione. Questo comportamento differisce da quanto avviene in molti altri linguaggi di programmazione, che, una volta compilati, producono codice macchina che può essere eseguito solo su sistemi specifici. Ciò permette l'indipendenza dalla piattaforma hardware, in quanto il codice prodotto è lo stesso per ogni macchina.

Ciò rende possibile eseguire la stessa applicazione su qualsiasi macchina; l'unica cosa da cambiare è l'interprete per quella specifica macchina.

Microsoft SQL Server 2008

Un database e' una collezione di dati che viene gestita e organizzata da un software specifico, il DBMS (DataBase Management System, Sistema di Gestione di DataBase). Un DBMS e' sostanzialmente uno strato software che si frappone fra l'utente ed i dati veri e propri. Grazie a questo strato intermedio l'utente e le applicazioni non accedono ai dati cosi' come sono memorizzati effettivamente, cioe' alla loro rappresentazione fisica, ma ne vedono solamente una rappresentazione logica. Cio' permette un elevato grado di indipendenza fra le applicazioni e la memorizzazione fisica dei dati.

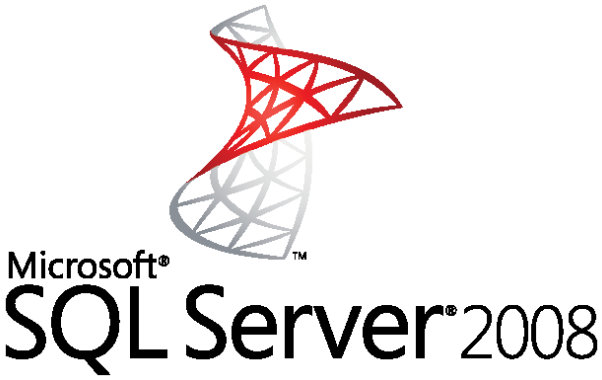


FIGURA 3 LOGO DI SQL SERVER 2008

Nel corso degli anni sono stati adottati numerosi modelli per i dati, a fronte dei quali esistono quindi vari tipi di database. I tipi piu' comuni sono i database relazionali essi si basano sul modello relazionale la cui struttura principale e' la relazione, cioe' una tabella bidimensionale composta da righe e colonne. Ciascuna riga, che in terminologia relazionale viene chiamata tupla, rappresenta un'entità che noi vogliamo memorizzare nel database. Le caratteristiche di ciascuna entita' sono definite invece dalle colonne delle relazioni, che vengono chiamate attributi. Entita' con caratteristiche comuni, cioe' descritti dallo stesso insieme di attributi, faranno parte della stessa relazione.

SQL è un linguaggio di interrogazione per database progettato per leggere, modificare e gestire dati memorizzati in un sistema di gestione di basi di dati basato sul modello relazionale , per creare e modificareschemi di database, per creare e gestire strumenti di controllo ed accesso ai dati.

MS SQL Server è un DBMS relazionale, prodotto da Microsoft. Nelle prime versioni era utilizzato per basi dati medio-piccole, ma a partire dalla versione 2000 è stato utilizzato anche per la gestione di basi dati di grandi dimensioni.

2 Analisi del progetto

La parte iniziale del tirocinio è stata dedicata all'analisi delle esigenze dell'azienda, studio della laguna di Venezia e dei suoi fenomeni per poter capire l'evoluzione temporale della marea e il fenomeno dell'*acqua alta*.

Convenzionalmente si denomina "acqua alta" una marea che supera a Venezia il valore di 80 cm sopra lo zero mareografico di Punta Salute: a questa quota sorgono problemi di trasporto e di viabilità pedonale nei punti più bassi della città (Piazza San Marco). Quando la marea supera i 100 cm, il fenomeno inizia ad interessare tratti più consistenti dei percorsi cittadini. All'aumento della frequenza degli eventi di acqua alta a Venezia nel corso degli ultimi decenni, hanno contribuito fenomeni quali l'eustatismo (innalzamento del livello del mare) e la subsidenza (abbassamento del suolo).
L'acqua alta segue il ciclo della marea (6 ore la marea cresce e le successive 6 cala): nei giorni in cui c'è acqua alta, questa dura solo per le ore centrali della fase crescente. Quindi solitamente l'acqua alta permane a Venezia per circa 3-4 ore. Subito dopo, una volta calata l'acqua, la città ritorna alla sua normalità. Questo è un fenomeno che si verifica solitamente nei mesi autunno-invernali, con maggiore probabilità tra novembre e dicembre, in questo periodo può succedere che solo in alcuni giorni si verifichi il fenomeno dell'acqua alta, spesso con un livello che interessa solo le parti più basse della città. Una marea eccezionale (>= 140 cm) si presenta, statisticamente, 1 volta ogni 4 anni. Quando si misura la quota di marea, viene usato come riferimento il livello sullo zero mareografico di Punta Salute. La città è nella sua quasi totalità (97%) a circa 100 cm sopra il livello del medio mare, questo significa che la quantità di acqua che può invadere le strade è sempre ben inferiore alla massima di marea annunciata. Per fare un esempio, una marea eccezionale di 140 cm corrisponde, in realtà, a circa 60 cm di acqua nei punti più bassi della città (Piazza San Marco) e allaga circa il 54 % del centro storico.
L'acqua alta non viene portata in città da un fiume o da un torrente in piena ma arriva dal mare. L'acqua pertanto invade la città salendo con lentezza dai canali. Questa fase dura, di norma, alcune ore. Una volta raggiunto il picco massimo l'acqua inizia a defluire fino a lasciare, come unica traccia, le strade bagnate come quando piove.

Le stazioni di rilevamanto, ovvero *mareografi,* rilevano valori di marea ad intervalli regolari, queste serie di valori possono essere reppresentate come *serie temporali* a tempi discreti*,* una successione di valori numerici, relativi all'evoluzione di un determinato fenomeno, ricavati da osservazioni eseguite ad intervalli regolari di tempo. Più in generale una serie storica è una successione ordinata di numeri reali che misura un certo fenomeno x_t esaminato rispetto al parametro indipendente t.

La marea totale è definita in ogni suo istante dalla somma di due componenti, la prima è la marea astronomica, la quale è un segnale deterministico, data la conoscenza delle leggi che regolano il moto di Terra, Sole e Luna rende possibile una descrizione accurata della forza di marea lunare e solare e delle conseguenti variazioni del livello marino in ogni punto dell'oceano. La marea astronomica viene comunemente calcolata mediante il cosiddetto metodo armonico, ovvero sommando un certo numero di componenti di marea rappresentate da onde sinusoidali. La seconda è il contributo meteo, segnale aleatorio causato dal vento e dalle differenze della pressione dell'aria sul mare durante il transito delle perturbazioni atmosferiche.
Nel nostro caso la marea può essere intesa come un processo stocastico, cioé una famiglia di

variabili casuali $\{x_t , t \in T\}$ indicizzate da un parametro t (tempo) variabile in un insieme indice T, detto spazio parametrico.
Analizzando le serie di valori, salvate all'interno di un database, ci si scontra con il problema dei "buchi" temporali.

Queste mancanze di valori, dovuti da guasti o malfunzionamenti dei mareografi, vengono salvati nel database come valore NULL.
Per risolvere questo problema è stato implementato un algoritmo ad Hoc, descritto nel prossimo paragrafo, che utilizza due tecniche per il riempimento dei "buchi".

Riempimento :

$$P(x) = \sum_{i=1}^{n} f(a_i) \prod_{j \neq i} \frac{x - a_j}{a_i - a_j}$$

Nel caso in cui la mancanza non sia eccessiva viene costruita una curva polinomiale dalla quale vengono interpolati i valori mancanti.

Sia assegnata una funzione f(x) di cui sono noti i valori $f(x_1)$, $f(x_2)$,…,$f(x_N)$ nei punti x_1, x_2, …, x_N, e si voglia calcolare il valore di f(x) in un punto $x \neq x_1, x_2, \ldots, x_N$, se x è compreso nell'intervallo $[x_1,x_N]$ si parla di interpolazione. Con l'interpolazione si costruisce un modello della funzione f(x) valido all'interno dell'intervallo $[x_1,x_N]$.
Per verificare l'errore commesso sono state fatte delle simulazioni prendendo una serie temporale e togliendo 5 valori contigui, dopo l'elaborazione la serie riempita prodotta viene confrontata con quella originale.
Vengono riportati in seguito i grafico di parte della simulazione con il relativo errore.

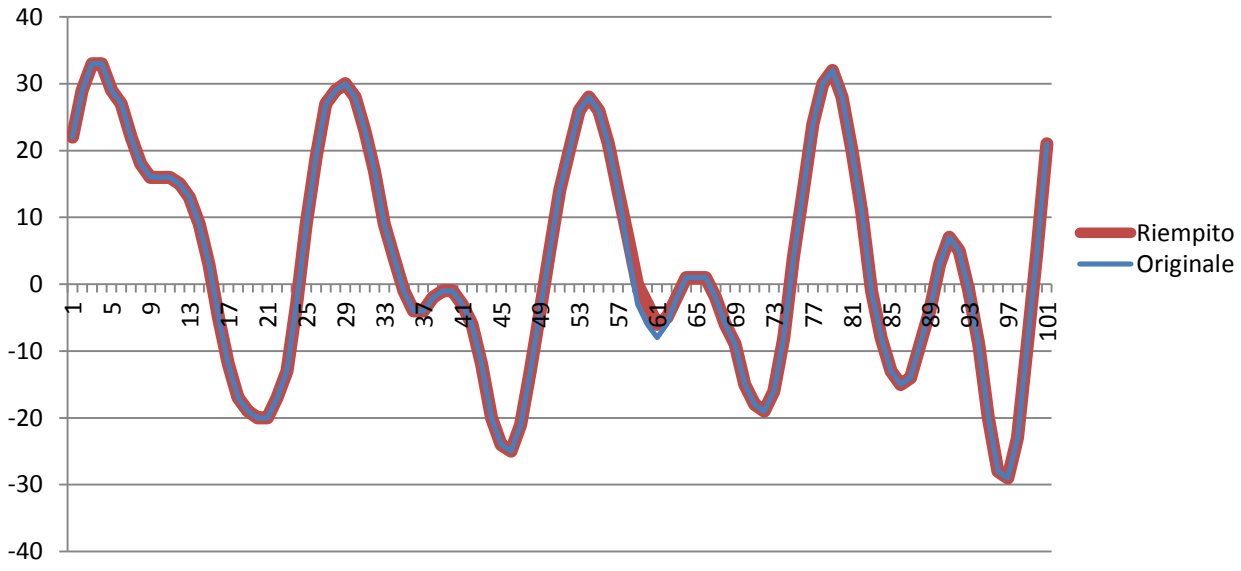


FIGURA 5 ESEMPIO DI RIEMPIMENTO

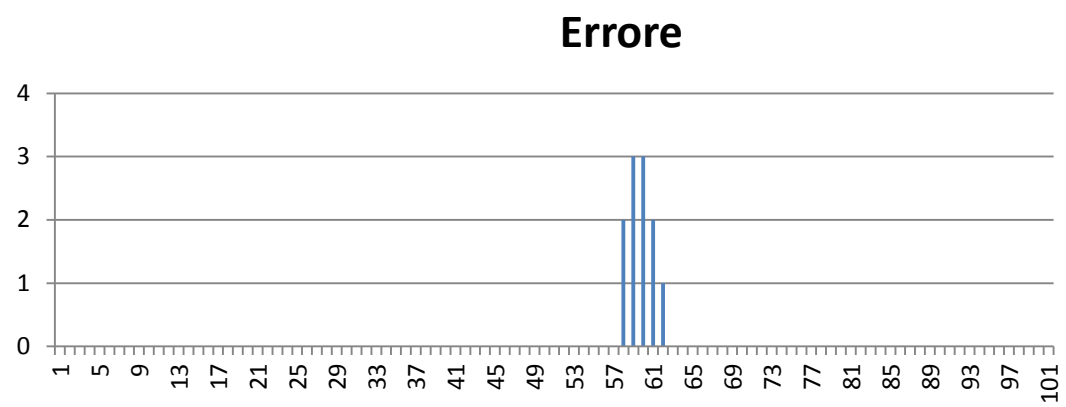


FIGURA 6 ERRORE COMMESSO NEL RIEMPIMENTO

L'errore assoluto massimo prodotto è di 3 cm, tenedo conto dell'escursione massima dei valori simulati e di più di 60 cm si ha un errore relativo percentuale di circa il 5%, valore accettabile tenedo conto il riempimento è stato fatto in valori di cresta dell'onda.

Stazione amica:

Nel caso in cui la mancanza di dati sia maggiore di una certa quantità viene usata la tecnica della *stazione amica.*

Questa tecnica consiste nella sostituzione in blocco dei dati mancanti con i dati di marea di un'altra stazione scelta in base alla correlazione incrociata.

$$R_{xy}[n] = (x \otimes y)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} x^*[m]\ y[n+m]$$

La correlazione incrociata rappresenta la misura di similitudine di due segnali come funzione di uno spostamento o traslazione temporale applicata ad uno di essi, nel caso in esame, avendo un *t*-upla di valori NULL prendo i *t* valori precedenti a questi e confrontandoli per ogni *n* sottoinsieme di valori della stazione candidata.

Di seguito viene riportato un esempio di analisi con Excel di correlazione incrociata tra due stazioni.

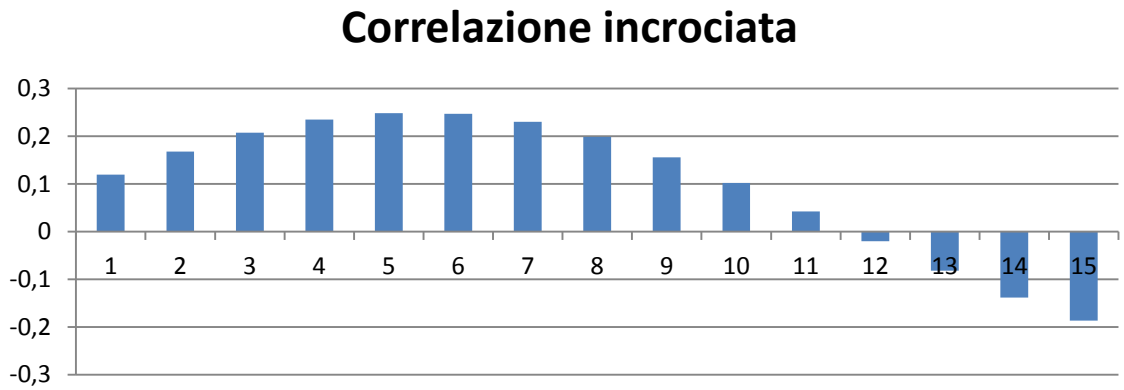


FIGURA 7 ESEMPIO DI CORRELAZIONE

In questo caso si può notare che si ha correlazione maggiore quando il segnale y ha una traslazione di 5 valori rispetto al segnale x, dopo di ché si andrà a sostituire i valori da y(5) a y(5+*t*) al posto dei valori NULL che avevamo in x.

Questo grafico mostra uno scorcio di simulazione di come era l'andamento originale della serie (X) e l'andamento della nuova serie trovata con la correlazione incrociata(Y).

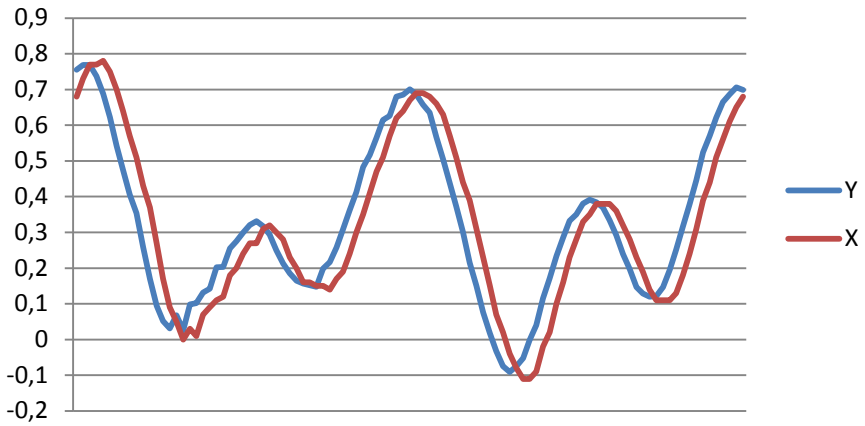


FIGURA 4 ESEMPIO DI DATI ORIFINALI E DATI TROVATI DA STAZIONE AMICA

2.0.3 Ricampionamento:

Uno degli obiettivi del software era quello di aumentare la risoluzione temporale per portare tutte le stazioni ad un passo temporale di rilevamento uniforme.
Per risolvere questo problema è stato implementato un algoritmo che ricampiona le serie storiche infittendone i valori utilizzando l'intepolazione, un metodo per individuare nuovi punti del piano cartesiano a partire da un insieme finito di punti dati.
L'algoritmo di ricampionamento inizialmente utilizzava l'interpolazione polinomiale, cioé l'interpolazione di una serie di valori con una funzione polinomiale che passa per i punti dati. In particolare, un qualsiasi insieme di *n+1* punti distinti può essere sempre interpolato da un polinomio di grado *n* che assume esattamente il valore dato in corrispondenza dei punti iniziali.Ma con questo metodo non sempre si avevano dei valori affidabili a causa del fenomeno di Runge, un problema relativo all'interpolazione polinomiale con polinomi di grado elevato. Esso consiste nell'aumento di ampiezza dell'errore in prossimità degli estremi dell'intervallo. Per questo motivo si è optato per l'intepolazione Spline, che, a differenza dell'interpolazione polinomiale che utilizza un unico polinomio per approssimare la funzione su tutto l'intervallo di definizione, l'interpolazione spline è ottenuta suddividendo l'intervallo in più sotto-intervalli e scegliendo per ciascuno di essi un polinomio di grado inferiore rispetto al quello polinomiale.

Una spline è una funzione, costituita da un insieme di polinomi raccordati tra loro, il cui scopo è interpolare in un intervallo un insieme di punti (detti *nodi* della spline), in modo da essere continua (almeno fino ad un dato ordine di derivate) in ogni punto dell'intervallo. Nel software si utilizza la **Spline Cubica Naturale**, funzione polinomiale di grado tre che consente un calcolo semplice e una approssimazione più che soddisfacente.

$$S(x) = \begin{cases} s_1(\mathrm{x})\ per\ x\ \in [x_0, x_1], \\ s_2(\mathrm{x})\ per\ x\ \in [x_1, x_2], \\ s_n(\mathrm{x})\ per\ x\ \in [x_{n-1}, x_n], \end{cases}$$

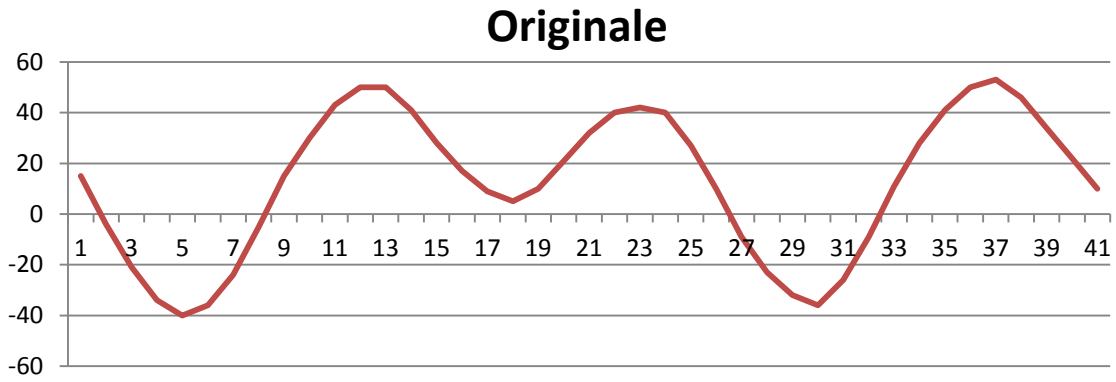
Dove

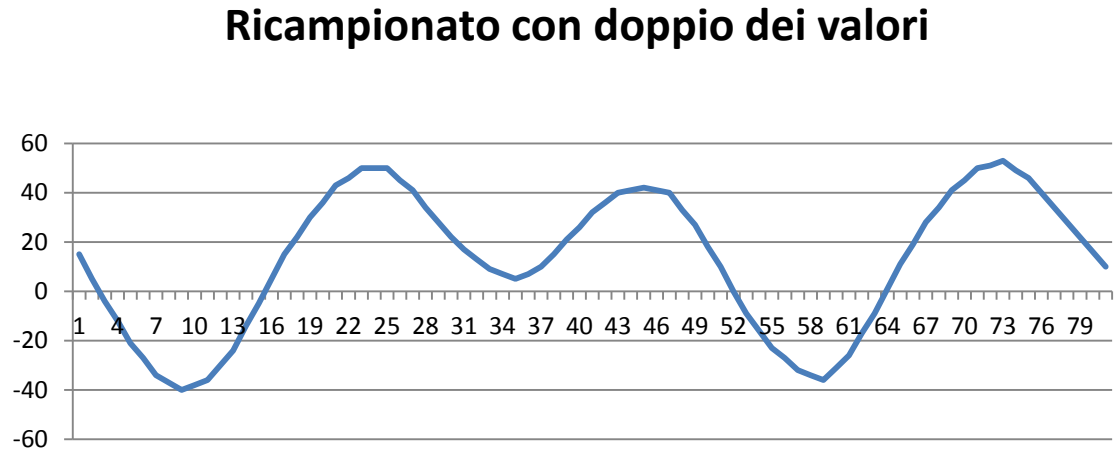
$$S_k(\mathrm{x}) = a_k + b_k(\mathrm{x} - x_{k-1}) + c_k(\mathrm{x} - x_{k-1})^2 + d_k(\mathrm{x} - x_{k-1})^3$$

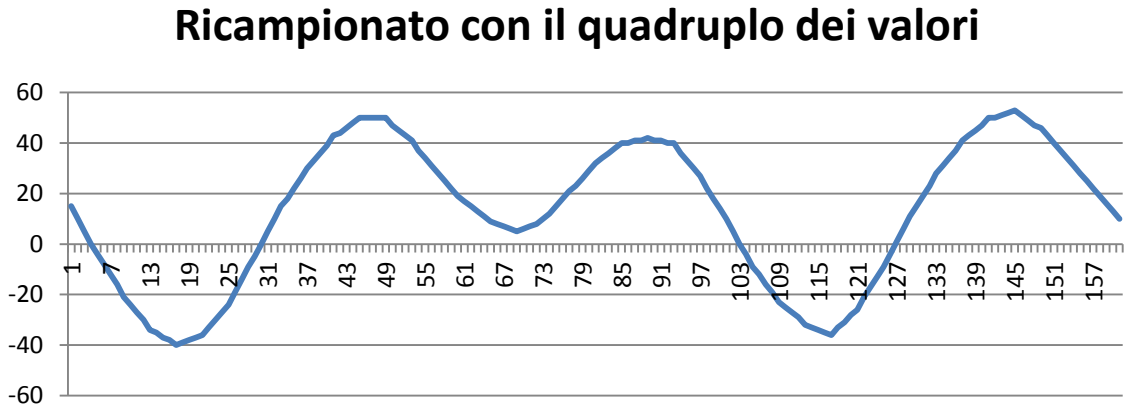
Si scrivono per ogni i ($1 \le i \le n$) le parabole cubiche, cioé funzioni del tipo $y = a_i + b_i x + c_i x^2 + d_i x^3$ passanti per i due punti (x_{i-1}, y_{i-1}) (x_i, y_i). Ne esistono ∞^2 per ogni i ($i = 1, \ldots, n$), quindi $2n$ dei $4n$ coefficienti sono arbitrari. Imponendo che sia le derivate prime che quelle seconde coincidano nei punti $x_1, \ldots x_{n-1}$ si hanno altre $2n-2$ condizioni lineari; rimangono ancora due scelte arbitrarie ed è uso imporre che la prima e l'ultima parabola cubica abbiano un flesso rispettivamente in x_0 e in x_n (spline naturale).

Si ha pertanto una funzione definita a pezzi sugli intervalli $[x_{i-1}, x_i]$ che è evidentemente continua e derivabile due volte in $[x_0, x_n]$ e soddisfa i dati.

Nei sequenti grafici vengono riportate le simulazioni di ricampionamento.







Come si può notare si hanno inizialmente 40 valori di marea, poi portati a 80 e 160.

2.0.4 Previsione:

Come già detto, la marea può essere intesa come un processo stocastico, cioè un sistema che evolve nel tempo seguendo leggi aleatorie. Supponiamo di voler modellizzare la dinamica di un punto che si muove su di una retta con una legge probabilistica. Possiamo introdurre un processo stocastico come la collezione delle variabili casuali $\{x_t\ , t\ \in T\}$ dove t è un parametro e T è l'insieme dei possibili valori di t. Di solito con t indica il tempo, quindi un processo stocastico è una famiglia di variabili casuali dipendenti dal tempo. Le variabili casuali X_t sono definite sull'insieme X, detto spazio degli stati; parlando di maree si dovrebbe dire che le variabili appartengono ad un insieme insieme continuo, ma dato che i mareografi producono dati ad intervalli di tempo regolari si parla di processo stocastico discreto. Gli elementi $x_t\ \in X$, ossia i valori che possono assumere le variabili casuali x_t, sono detti stati del sistema e rappresentano i possibili risultati. Nel caso della marea allora si parla di processo aleatorio continui a parametro discreto.

Per effettuare la previsione a breve termine è stata utilizzato il processo di estrapolazione su di una curva spline cubica naturale.
A differenza dell'interpolazione, l'estrapolazione è il processo che permette di calcolare il valore di informazioni esterne ad un insieme discreto di dati noti. In sostanza, dato un piano cartesiano sul quale siano stati tracciati i punti (x_i,y_i) corrispondenti all'insieme di valori noti, si vuole trovare il valore y_w corrispondente ad un valore x_w maggiore (o minore) di ciascun x_i .

La costruzione della Spline varia a seconda della distanza temporale in cui si vuole effettuare la previsione. Volendo a fare una previsione a distanza *t* , prendo gli ultimi *n* valori con con passo temporale *t* e da questi ricavo la Spline.

Vengono proposti 3 esempi grafici, rispettivamente con una previsione a passo temporale t = 1, 2 e 3 e n = 6; Il pallino verde rappresenta la previsione, i pallini blu i dati in DB e i pallini rossi i dati presi per la costruzione della Spline.

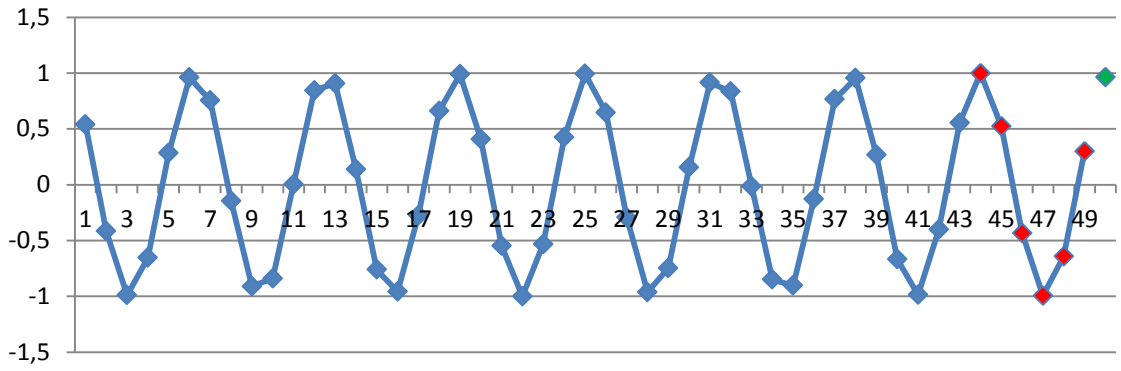


FIGURA 9 PASSO TEMPORALE 1

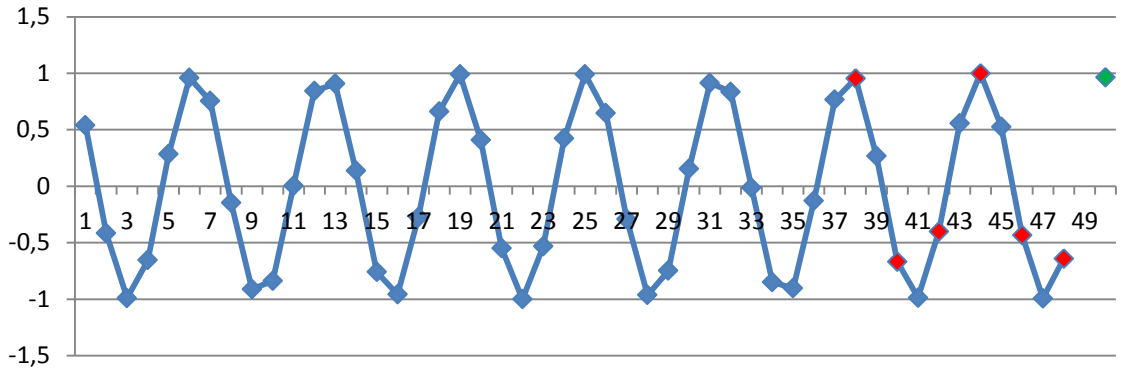


FIGURA 11 PASSO TEMPORALE 2

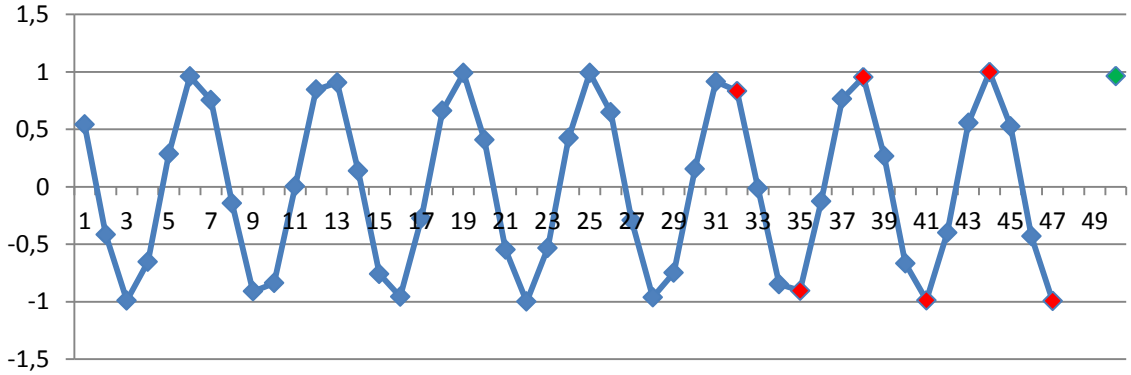


FIGURA 10 PASSO TEMPORALE 3

2.1 Analisi del sistema

"L'architettura di un sistema descrive la sua forma e struttura: quali sono i suoi componenti ed il modo in cui interagiscono"
Jerrold Grochow (vicepreside del Massachusetts Institute of Technology)

Architettura del sistema di monitoraggio, previsione e supporto decisionale

L'intero sistema di raccolta dati ed analisi dei dati sarà collocato, presso l'Arsenale di Venezia in cui sarà realizzata la sala operativa del sistema di gestione e rappresentazione delle misure e delle previsioni. L'attività di gestione del sistema MOSE e della laguna di Venezia coinvolge gran parte dell'Arsenale Nord e le bocche di porto.

L'architettura generale del sistema di monitoraggio meteomarino, del sistema di previsione di livello e del sistema di supporto alle decisioni è formata dai seguenti macrocomponenti:

Strutture distribuite geograficamente sul territorio :

- Rete di monitoraggio meteomarino, a sua volta suddivisa in:
  - Sensori di rilevamento;
  - Reti di trasmissione dati;
- Previsioni meteorologiche.

Strutture del Centro di Comando e Controllo :

- Banche dati:
  - Server di acquisizione e preprocessamento dati;
  - Server di archiviazione dati;
- Sistema di previsione dei livelli di marea;
  - Server di modellazione dati;
  - Server di presentazione dati;
- Sistema di supporto alle decisioni.

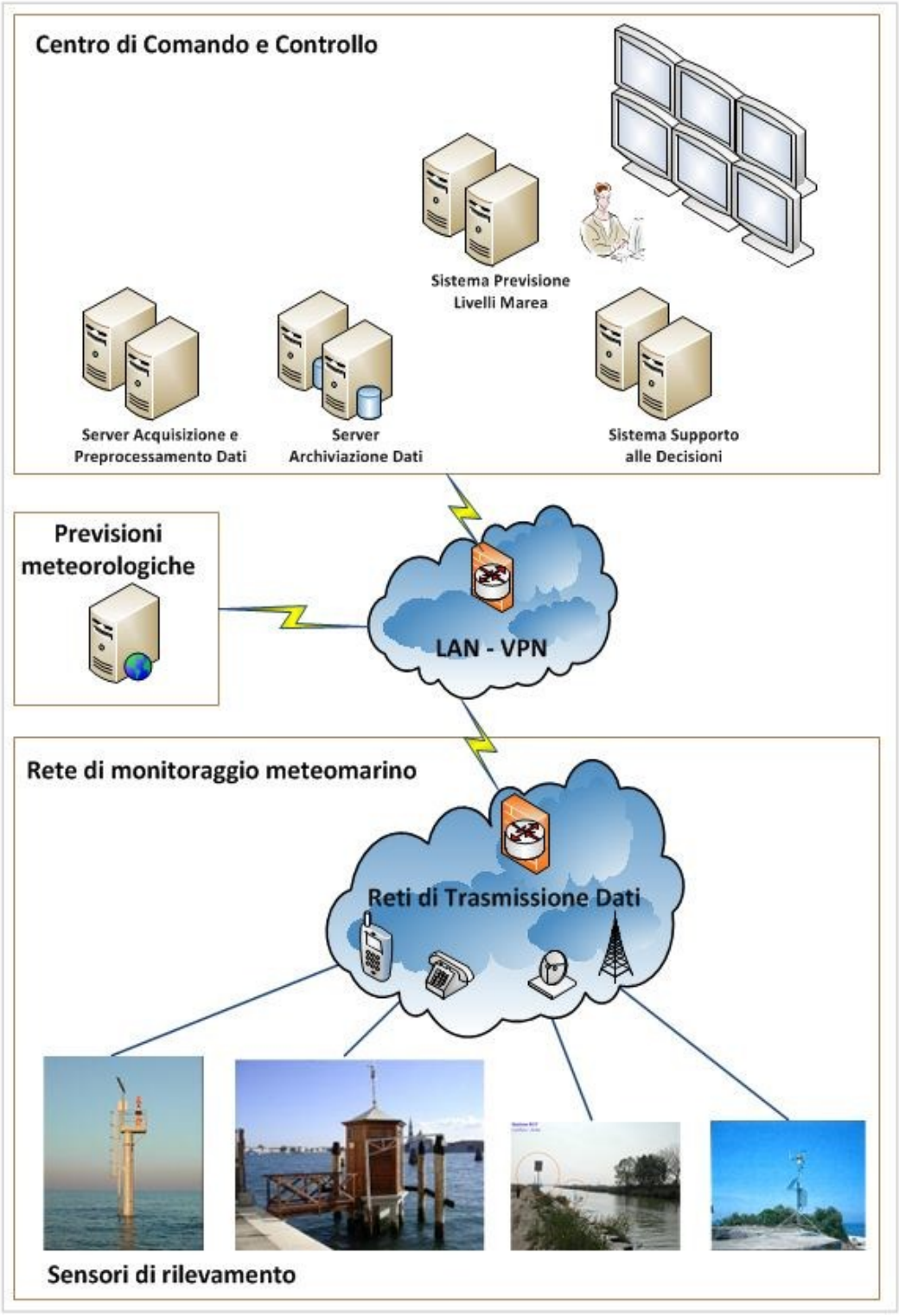


FIGURA 12 ARCHITETTURA DI SISTEMA

Ai fini di acquisire tutti i dati necessari all'esecuzione dei modelli di previsione è stata realizzata a livello lagunare e di area di mare circostante le bocche di porto una rete di monitoraggio meteomarino composta da:

- stazioni equipaggiate con sensori di misura e sistemi di acquisizione ed elaborazione del dato;

- reti di trasmissione dei dati misurati verso una centrale.

Nel loro complesso le tipologie di rete coinvolte sono:

Reti di monitoraggio mareografico;

Reti di monitoraggio flussometrico alle foci dei fiumi del bacino scolante;

Reti di monitoraggio ondametrico;

Reti di monitoraggio flussometrico alle bocche di porto;

Reti di monitoraggio anemometrico e pluviometrico.

Tre stazioni sono state realizzate nei pressi delle bocche di porto, al di fuori dell'area d'influenza del flusso e del riflusso di marea, quattro stazioni sono state realizzate nei pressi delle bocche di porto lato laguna, una è stata collocata all'interno della bocca di porto di Lido, lato mare, una è stata collocata presso l'Arsenale di Venezia e una in laguna Nord presso Valle Dogà.

La struttura di supporto della strumentazione varia in funzione delle localizzazioni; le stazioni in mare e quella nei pressi di San Nicolò sono state realizzate su palo in acciaio, mentre le rimanenti in laguna sono state installate su piattaforme in calcestruzzo o agganciate a banchine e/o manufatti esistenti.

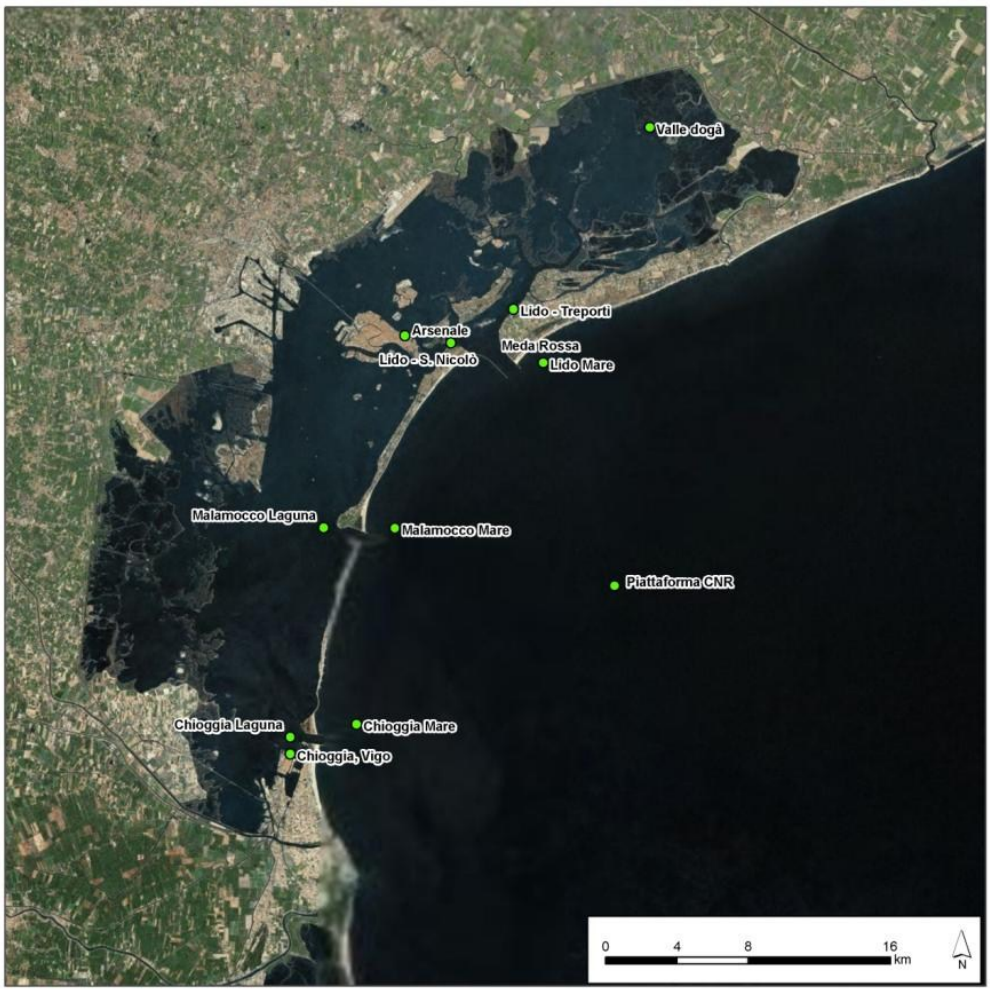


FIGURA 13 LAGUNA DI VENEZIA CON MAREOGRAFI

Le stazioni mareografiche sono dotate d'impianti di segnalazione luminosa (diverse per le stazioni su palo da quelle su piattaforma), strumentazione mareografica, sistemi di acquisizione e trasmissione dati.

Il dato di livello viene acquisito mediante lettura istantanea ogni 120 s da tutte le stazioni. La trasmissione dei dati dalle stazioni alla centrale acquisizione dati avviene mediante modem GPRS ad un server dedicato ogni 15 minuti.

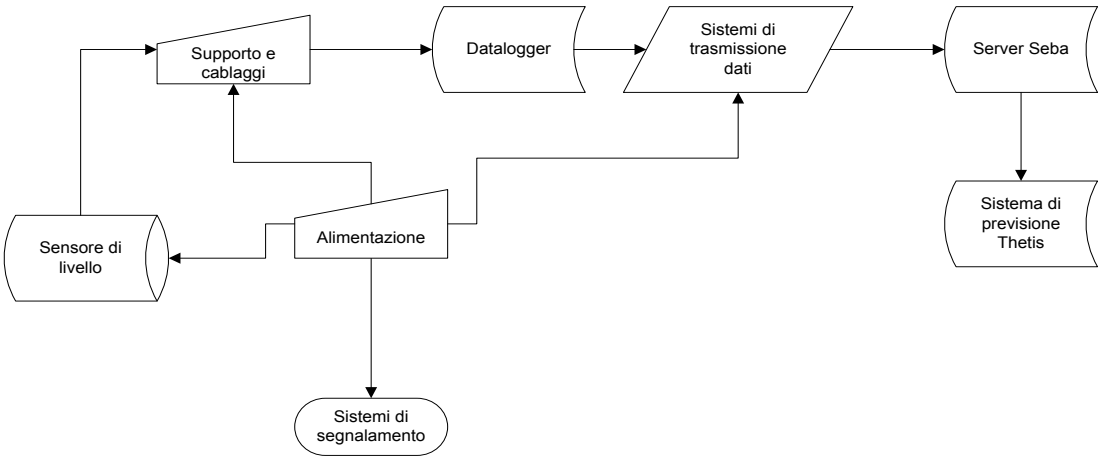


FIGURA 14 SCHEMA LOGICO DEL SISTEMA

Le reti di monitoraggio meteomarino gestite utilizzano modalità diverse di trasmissione dei dati che confluiscono nei Server Database al servizio dei sistemi di raccolta dati, dei sistemi di previsione di livello e dei sistemi di supporto alle decisioni.

Reti di trasmissione dati

I sistemi fornitori di dati esterni posso essere:
- reti di monitoraggio meteomarino;
- sistemi di previsione meteorologiche.

Le modalità di fornitura dati per le reti di monitoraggio metomarino sono:

- Scarico tramite collegamento diretto punto-punto su linea telefonica

- La centralina collegata allo strumento viene chiamata mediante su rete a telefonica analogica o rete cellulare GSM;

- Una volta stabilita la connessione i dati misurati dallo strumento vengono ricevuti e salvati in un'area di memorizzazione temporanea.

- Scarico tramite collegamento ad internet e sito ftp:

- La centralina collegata allo strumento viene connessa ad internet tramite rete cellulare GPRS/UMTS;

- Il software di controllo della centralina invia il file con i dati misurati dallo strumento ad un sito web o ftp di transito;

- La procedura di acquisizione si connette anch'essa via internet al sito web o ftp di transito e scarica il file e lo salva

- Scarico tramite collegamento su ponte radio e rete privata geografica:

- La centralina collegata allo strumento è collegata ad una rete di ponti radio basati su tecnologia HiperLAN o Wi-MAX;

- Viene configurata una rete privata geografica (WAN Wide Area network) in modo tale che alle centraline associate allo strumento siano associati degli indirizzi IP raggiungibili dalla centrale;

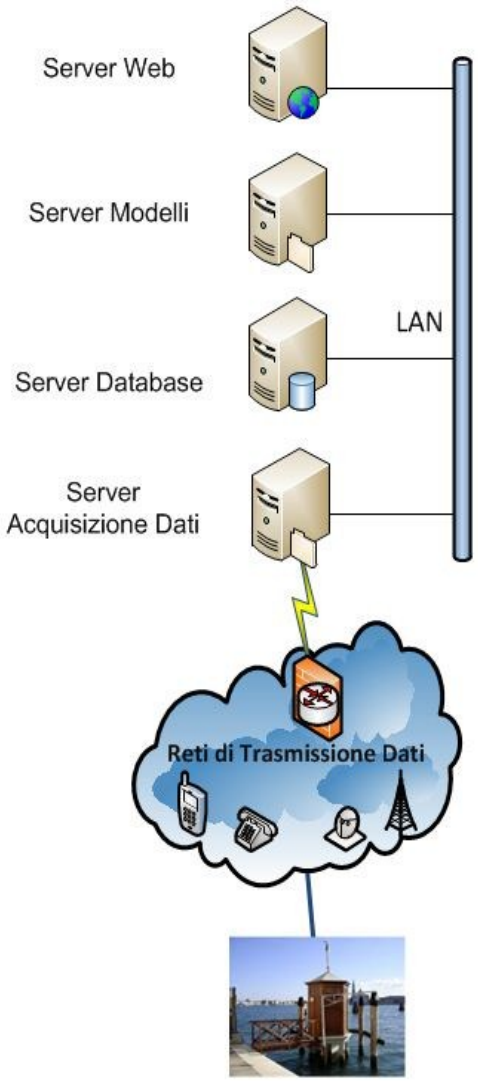
- Le procedure di acquisizione colloquiano direttamente con le centraline come se fossero in rete locale, ed effettualo le chiamate per lo scarico dei dati misurati salvandoli in un area di memorizzazione temporanea; in alternativa potrebbero essere le centraline stessa ad inviare i dati misurati verso la centrale.

- Una volta stabilita la connessione i dati misurati dallo strumento vengono ricevuti e salvati in un'area di memorizzazione temporanea.

Strutture archiviazione ed elaborazione dati

Il modello logico che rappresenta il sistema di monitoraggio meteomarino e previsione di livello è basato sui seguenti livelli:

- livello dati dal campo: ovvero i sensori che misurano diverse grandezze fisiche o chimiche, li memorizzano su un datalogger locale e li trasferiscono alla centrale mediante una rete di trasmissione dati;
- livello i preprocessamento dati: ovvero un insieme di server e procedure che gestiscono la ricezione dei dal campo e li trasferiscono, dopo opportune trasformazioni, nei vari database;
- livello di memorizzazione dati: ovvero i database server dedicati alla memorizzazione sia dei dati dal campo che dei risultati dei vari modelli di previsione;
- livello di elaborazione dati: ovvero i server su cui girano i vari software di modellazione
- statistica e deterministica e che generano sia le interfacce utente per la presentazione dei risultati dei modelli di previsione che quella per il monitoraggio e controllo dell'intero sistema;
- livello di presentazione: ovvero i personal computer con browser internet su cui sono visualizzati i risultati dei modelli e le interfacce di monitoraggio e controllo.



Le caratteristiche hardware dei server sono le seguenti:

Processore	2 Intel XEON
Memoria	2 GB
Dischi fissi	C:\ 48 GB (Programmi) D:\ 224 GB (Dati)
Sistema operativo	Microsoft Server 2003 R2 SP2

FIGURA 5 STRUTTURA ARCHIVIAZIONE ED ELABORAZIONE DATI

Sitema di monitoraggio dello stato dei sistemi

Nell'ambito delle varie attività in corso di mantenimento delle reti meteomarini, dei sistemi di previsione acque alte e decisionali, delle relative infrastrutture informatiche (database, server, connessioni, pagine WEB) è emersa la necessità di ampliare il monitoraggio dello stato dei sistemi, in parte già realizzato tramite software autoprodotto Thetis con VB .NET, che ha già esperienza di uso dello stesso nell'ambito dei sistemi di monitoraggio e nella gestione del traffico (autobus) nella città di Roma.

Si è scelta la formula di acquisire un software per il monitoraggio dei sistemi (strumenti dotati di indirizzo IP statici, database, server, modelli operativi), addestrando dei componenti interni del Centro sistemi di previsioni e Modelli per la predisposizione e programmazione delle regole più idonee di monitoraggio e per l'impostazione dei messaggi di allerta via mail ed SMS. L'obiettivo è ridurre le cause di guasto e la durata degli stessi.

È stato inizialmente fatto uno screening delle necessità di controllo ed è emerso che le aree da controllare in modalità automatica sono quelle riportate di seguito:

Controllo dello stato dei Server
Controllo dello stato del servizio ftp su Web Server
Controllo dello stato dei siti ftp esterni
Controllo dello stato dei servizi di DB
Controllo dello stato del servizio Internet e disponibilità di banda
Controllo dello stato del server di posta elettronica
Controllo dello stato servizio WEB su WEB Server
Controllo dello stato degli script di scaricamento dati
Controllo dello stato degli script di scaricamento dati
Controllo sull'ultimo dato scaricato
Controllo sulle anomalie sugli ultimi valori scaricati in DB
Controllo sullo stato degli script sistema di previsione di caricamento su DB
Controllo sullo stato del sistema di supporto alle decisioni

Il Database

Le serie temporali dei dati misurati e previsti vengono archiviati in opportune tabelle della banca dati del sistema di supporto alle decisioni in un formato il più possibile standardizzato:

- stazione di riferimento;
- tipologia di parametro;
- data e ora di riferimento del dato;
- valore del dato.

Per i dati di previsione vengono inoltre fornite altre informazioni per identificare correttamente e in modo univoco la data e l'ora di emissione della previsione e precisamente:

- data e ora di emissione della previsione meteorologica (FctDate);
- ritardo della emissione della previsione di livello rispetto all'emissione della previsione meteo (Delay);

Il sistema di supporto alle decisioni per la gestione delle dighe mobili a difesa di Venezia e della sua laguna necessita di una banca dati rifornita di:

- Dati di previsione per la gestione della fase preparatoria della chiusura e per la gestione delle allerte e dei preavvisi (previsioni meteorologiche e di livello);
- Dati di misura in tempo reale per la gestione operativa del sistema di previsione acque alte;
- Dati di misura in tempo reale per la gestione operativa del modello di supporto alle decisioni per la gestione delle paratoie mobili (in particolare livelli di marea misurati in laguna e in mare, velocità e direzione del vento, precipitazione, portata dai fiumi).

In particolare sono presenti i seguenti database su supporto MS SQL Server:

- ASTRO: database contenete le maree astronomiche generate per i punti di interesse al fine di determinare i contributi meteorologici in diversi punti del bacino lagunare e del mare Adriatico; i dati sono archiviati con passo di 30 minuti in orario GMT;
- WATERFORECAST: database principale per l'operatività del modello di previsione acque alte; contiene i dati di misura in tempo reale e le previsioni di livello e meteorologiche ECMWF e Cosmo-I7 del SMREMR;

Nel database **waterforecast** di SQL Server sono contenute le seguenti tabelle di dati misurati in tempo reale:

measdata: tabella funzionale al modello di supporto alle decisioni;

formato tabella:

StationID

ParamID

TimeStepID

Val

contenuto tabella:

livello di marea: mareografi ICPSM passo 5 minuti

livello di marea: mareografi MAV-Thetis passo 2 minuti

vento (velocità e direzione) alla Piattaforma “Acqua Alta” stazione di misura ICPSM passo 5 minuti

dati meteorologici (vento, pressione, temperatura aria) della rete METAR passo 30 minuti

precipitazioni stazioni ARPAV passo 60 minuti

orario dei dati: GMT

frequenza di aggiornamento: 30-120 minuti (a seconda del tipo di dato)

Tabella delle stazioni mareografiche della rete Thetis che vengono archiviate in tempo reale nella tabella measdata del database waterforecast.

StationID	Code	Name	Est Gauss-Boaga	Nord Gauss-Boaga
65	LIM	Lido Mare	2320756	5033412
66	SNI	San Nicolò	2315584	5034525
67	TRP	Treporti	2319089	5036410
63	MAM	Malamocco Mare	2312432	5024130
64	MAL	Malamocco Laguna	2308418	5024149
61	CHM	Chioggia Mare	2310272	5013124

2.2 Analisi del software

Il normale ciclo di vita di un'applicazione consiste nell'esecuzione sequenziale di numerosi blocchi di codice richiamati dal Thread principale, cioè un'entità funzionale elementare del sistema operativo in grado di eseguire codice.

Il software esegue principalmente cinque tipi di operazioni:

Reading : Lettura dei dati dal DataBase
Filling : Riempimento dei "Buchi"
Resampling : Ricampionamento a trama più fitta
Forecasting : Previsione futura
Writing : Scrittura dei dati nel DataBase

Queste operazioni vengono effettuate su i dati storici già presenti nel database e poi ripetutte ciclicamente quando un nuovo dato viene rilevato da un mareografo e salvato in DB.
Ovviamente quando viene eseguito del Thread principale il metodo chiamante risulta bloccato fintanto che la routine invocata non giunge al termine.
Si tratta di un approcio con diverse limitazioni, quindi per poter eseguire operazioni richieste su i dati di ogni mareografo è stata utilizzata la tecnica del **multi-threading**. Un classica applicazione scritta in modo normale non dà la possibilità di servire più richieste contemporaneamente, infatti sarebbe costretta a finire l'esecuzione di una richiesta prima di poterne servire un'altra. Scrivendo applicazioni che sfruttano il multi-threading è invece possibile suddividere il classico processo in più thread, ovvero suddividere il processo principale in diversi flussi di codice concorrenti. Nel nostro caso nel nostro matodo *main* viene creato un Thread per ogni mareografo, e ogniuno di questi verrà eseguito in parallelo.

Il Sofware è composto dalle seguenti classi:
- DataReader : classe che implementa i metodi per leggerere i dati da Data Base.
- DataWriter : classe che implementa i metodi per scrivere i dati sul Data Base.
- ForecastData : classe che implementa i metodi per le previsioni di marea
- ProcessData : classe che implementa i metodi per eleborare i dati di marea
- Record : questa classe costruisce oggetti di tipo Record, questo oggetto contiente il dato di marea come tipo Double e la data di salvataggio come tipo Date
- Station : questa classe costruisce oggetti di tipo Stazione e implementa i metodi per chiamare le classi per l'elaborazione dei dati di marea.

Connessione al Data Base

Il software utilizza la libreria JDBC (Java Database Connectivity), un'interfaccia completamente Java utilizzata per eseguire istruzioni SQL. In questo caso il driver JDBC funge da ponte, per accedere al database attraverso driver ODBC (Open Database Connectivity, una API standard per la connessione ai DBMS), che deve essere presente e configurato sul server.
Viene riportato lo spezzone di codice del costruttore della classe DataReader per connettersi al DB.

```
private Connection con;
private Statement sta;

public DataReader(String url, String user, String password)
throws ClassNotFoundException,
       InstantiationException,
       IllegalAccessException,SQLException {
        //Carica il driver JDBC
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver").newInstance();
        System.out.println("SQL  JDBC driver loaded ok.");
        //Ottiene la connessione al Data base
        con = java.sql.DriverManager.getConnection(url, user, password);
        //Inizializza l'oggetto Statement per eseguire le query
        sta = con.createStatement();
    }
```

Scaricamento dei dati

Una volta connessi al DB è possibile eseguire operazioni di scrittura e lettura su di esso.

Viene riportatolo spezzone di codice del metodo **getData** della classe DataReader per scarica i dati.

```
private ResultSet res;

    public String[] getData(String query) {
        String[] tmp = new String[100];
        int i = 0;
        try {
            //Esegue la query e salva il contenuto in un oggetto ResulSet
            res = sta.executeQuery(query);
            //Inserisce il contenuto del ResultSet in un array di Stringhe
            while (res.next()) {
                if (i >= tmp.length) {
                    tmp = (String[]) Utils.resizeArray(tmp, tmp.length * 2);
                }
                tmp[i++] = res.getString(1);
            }
            //Chiude il ResultSet
            res.close();
        } catch (SQLException ex) {
            Logger.getLogger(DataReader.class.getName()).log(Level.SEVERE, null, ex);
        }
        tmp = (String[]) Utils.resizeArray(tmp, i);
        return tmp;
    }
```

Un esempio di query utilizzata è la seguente:

```
SELECT  val*100 FROM MEASDATA WHERE ParamId = 3 AND STATIONID = 3 order by TIMESTEPID desc
```

Questa query estrae da DB i valori di marea misurati, invece le due query seguenti estraggono i valori temporali

```
SELECT REPLACE(STR(DATEPART(YYYY,timeStepId),4),' ','0')+'-'+
REPLACE(STR(DATEPART(mm,timeStepId),2),' ','0')+'-'+
REPLACE(STR(DATEPART(dd,timeStepId),2),' ','0')
FROM MEASDATA WHERE ParamId = 3 AND STATIONID = 3 order by timestepid desc

SELECT REPLACE(STR(DATEPART(HH,timeStepId),2),' ','0')+':'+
REPLACE(STR(DATEPART(mi,timeStepId),2),' ','0')+':'+
REPLACE(STR(DATEPART(ss,timeStepId),2),' ','0')
FROM MEASDATA WHERE ParamId = 3 AND STATIONID = 3 order by timestepid desc
```

La prima estrae i dati con il formato “YYYY-mm-dd”, la seconda invece con il formato “HH:mi:ss”. Questi due valori, assieme al dato misurato, serviranno per creare l’oggetto Record.

L’oggetto Record

Per sfruttare a pieno le potenzialità di Java e sviluppare un software più leggibile e facilmente implementabile si è usufruito della programmazione orientata ad oggetti creando il tipo di dato Record. Secondo il paradigma di programmazione ad oggetti un’oggetto è un’entità in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi. È particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare. La programmazione ad oggetti prevede di raggruppare in una zona circoscritta del codice sorgente la dichiarazione delle strutture dati e delle procedure che operano su di esse. Le classi costituiscono dei modelli astratti, che a tempo di esecuzione sono istanziati per creare oggetti; questi sono dotati di attributi (dati) e di metodi (procedure) secondo quanto dichiarato dalle rispettive classi.

Per il salvataggio e l’elaborazione dei dati del database è stato necessario creare una classe Record che crea l’omonimo oggetto.
Il record non è altro che un oggetto che contiente al suo interno il valore misurato (o elaborato) e il valore temporale associato.

```
private double value;

private SimpleDateFormat sdf;
private java.util.Date date;


public Record(double value, String date , String time) {
        try {
            this.value = value;
            sdf = new SimpleDateFormat("yyy-MM-dd HH:mm:ss");
            this.date = sdf.parse(date + " " + time);
                    } catch (ParseException ex) {
            Logger.getLogger(Record.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

Il costruttore prende in ingresso un valore double, che viene inserito direttamente nel campo *value* del Record, e due String che rappresentano la data e il tempo associato, queste vengono concatenate e convertite in un tipo Date.
Il dato Date è molto utile dato che contiente un medoto per la conversione della data in un dato

di tipo Long, cioé il numero di millisecondi trascorsi dal 1 gennaio 1970 alle 00:00:00 alla data interessata. Grazie al dato Long risulta più facile trovare la differenza temporale tra due Record

```
public static Long timeStep(Record rec1, Record rec2) {

   Date time1;
   Date time2;
   long diff = 0;
   try {
       time1 = rec1.getTime();
       time2 = rec2.getTime();
       diff = Math.abs(time2.getTime() - time1.getTime());
   } catch (NullPointerException e) {
   }
   return diff;
}
```

Scrittura dei dati

Viene riportatolo spezzone di codice del metodo **toDatabase** della classe DataReader per inserire i dati nel data base.
I parametri d'ingresso del matodo sono i dati, contenuti in un array di Record, la tabelle e l'identificativo della stazione dove inserire i dati.

```
public void toDatabase(Record[] data, String table, int stationId) throws SQLException {
        String query = null;
        int i = 0;
        for (i = 0; i < data.length; i++) {

                //Preparazione della query
                query = "INSERT INTO " + table + " VALUES ( " + stationId + " , ";
                query += data[i].getValue() + " , CAST(' ";
                query += data[i].getDateInt() + " ";
                query += data[i].getTime() + " ' AS datetime));";
                //Esecuzione dell'update
                int updateCount = sta.executeUpdate(query);

              }
}
```

Filling

Questa operazione viene eseguita su i dati storici; data la presenza di dati *null*, dovuti a errori di misura, è stato necessario riempire queste mancanze per dare una continuità alla serie di dati. Le tecniche di riempimento utilizzate sono due:

- Nel caso in cui la mancanza di dati sia minore od uguale a 6 Record viene utilizzata l'interpolazione polinomiale.
  Vengono presi i 3 dati precedenti ai valori *null* e i 3 dati successivi, dati questi punti si trova la curva polinomiale di grado 6 e con questa interpolo gli *n* dati mancanti.

```
public static double polyInterpolation(double x, double fi[]) {
        //Grado della funzione interpolante
        int n = fi.length - 1;
        //Creo e inizzailizzo l'array rappresentante l'ascissa, a elementi
        //equidistanti
        double xi[] = new double[fi.length];
        for (int i = 0; i < xi.length; i++) {
            xi[i] = i;
        }


        double dp[][] = new double[n + 1][];
        double dm[][] = new double[n + 1][];

        //Assegno la prima colonna delle correzioni
        dp[0] = (double[]) fi.clone();
        dm[0] = (double[]) fi.clone();

        // Find the closest point to x
        //Trovo il punto più vicino a x
        int j0 = 0, k0 = 0;
        double dx = x - xi[0];
        for (int j = 1; j <= n; ++j) {
            double dx1 = x - xi[j];
            if (Math.abs(dx1) < Math.abs(dx)) {
                j0 = j;
                dx = dx1;
            }
        }
        k0 = j0;

        //Stima ricorsivamente il resto delle correzioni
        for (int i = 1; i <= n; ++i) {
            dp[i] = new double[n - i + 1];
            dm[i] = new double[n - i + 1];
            for (int j = 0; j < n - i + 1; ++j) {
                double d = dp[i - 1][j] - dm[i - 1][j + 1];
                d /= xi[i + j] - xi[j];
                dp[i][j] = d * (xi[i + j] - x);
                dm[i][j] = d * (xi[j] - x);
            }
        }

        //Aggiorna l'intepolazione con le correzioni
        double f = fi[j0];
        for (int i = 1; i <= n; ++i) {
            if (((dx < 0) || (k0 == n)) && (j0 != 0)) {
                j0--;
                f += dp[i][j0];
                dx = -dx;
            } else {
                f += dm[i][j0];
                dx = -dx;
                k0++;
            }
        }
        return f;
    }
```

Aumentando il grado del polinomio la curva seguirà più fedelmente i valori e si otterranno dati più affidabili. Se aumentiamo troppo il grado però ricadiamo nel problema di overfing:, la funzione approssimante segue perfettamente i dati descrivento anche errori casuali o rumore. Un modello che è stato Overfit avrà generalmente scarsa prestazioni predittive, dato che può esagerare con le piccole oscillazioni nei dati.

- Nel caso in cui gli *n* dati mancanti siano maggiori di 6 l'interpolazione non è più affidabile e si utilizzano i dati della "*stazione amica*".
  La *stazione amica* è la stazione che ha maggior correlazione con quella presa in esame. Trovata la *stazione amica* vengono presi gli *n* punti precedenti alla serie di NULL, e con questo set di dati si fa la correlazione con ogni set di *n* dati contigui della *stazione amica.* Una volta trovato il set di della stazione amica si prendono gli *n* Record succesivi e si sostituiscono ai dati NULL della serie originale, togliendo un offset dovuto dall'altezza del mare.

```
private Record[] values
private void fillSubset(int start, int length, Record[] bestFriend) {
    double tmp = -10;
    int j = 0;
    Record[] arrTmp = Arrays.copyOfRange(values, 0, length);
    //Trova il subset con maggior correlazione
    for (int i = 1; i < start; i++) {
        if (tmp < Utils.getCorrelation(
                Arrays.copyOfRange(bestFriend, i, i + length),
                Arrays.copyOfRange(values, start, start + length))) {
            tmp = Utils.getCorrelation(
                    Arrays.copyOfRange(bestFriend, i, i + length),
                    Arrays.copyOfRange(values, start, start + length));
            arrTmp = Arrays.copyOfRange(bestFriend, i, i + length);
            j = i;
        }
    }
    for (int i = start + length; i < values.length - length; i++) {
        if (tmp < Utils.getCorrelation(
                Arrays.copyOfRange(bestFriend, i, i + length),
                Arrays.copyOfRange(values, start, start + length))) {
            tmp = Utils.getCorrelation(
                    Arrays.copyOfRange(bestFriend, i, i + length),
                    Arrays.copyOfRange(values, start, start + length));
            arrTmp = Arrays.copyOfRange(bestFriend, i, i + length);
            j = i;
        }
    }

    //Sostituisco il subset trovato con maggior correlazione con l'originale
    for (int k = j; k < j + length; k++) {
        values[start + length + k - j].setValue(arrTmp[k - j].getValue());
    }
}
```

Resampling

Questa operazione viene eseguita sia sui dati storici che su i dati in tempo reale.

Data la non omogeneità dei tempi di acquisizione dei dati da parte delle stazioni di rilevamento si è voluto portare la risoluzione dei dati a passo minore o uguale a 5 minuti.

L'algoritmo che esegue il ricampionamento è ricorsivo, prende in ingresso un'array *n* di Record e ritorna in uscita un array di *n* * *m* Record.

Presi due Record che distano tra loro un tempo *t* maggiore di 5 minuti trovo *m* tale che dividendo *t* in *m* frazioni il nuovo step temporale sia minore o uguale a 5 minuti.

```
public Record[] resample(int n) {
        //Chiama il metodo di ricampionamento ricorsivo
        valuesRes = resampleP(values, n + 1);
        return valuesRes;
    }

    private Record[] resampleP(Record[] x, int n) {
        //Caso base
        if (n <= 1) {
            return x;
        }
        //Trova il fattore primo piu piccolo
        int spf = Utils.smallestPrimeFactor(n);

        Record[] out = new Record[x.length * (spf)];
        Cubic[] c = null;
        NatCubic nc = new NatCubic();

        long dateStep = 0l;
        long timeStep = 0l;

        for (int i = 0; i < out.length - spf;) {
            //Se il mudulo dell'indice è zero allora inserisco nell'array di
            //output il valore originale
            if (i % spf == 0) {
                out[i] = x[i / spf];
                i++;
            } else {
                //Altrimenti inserisco il nuovo Record con il valore intepolato
                Record[] tmp = null;
                int r = 10;

                for (int t = 10 + spf; t > 0;) {
                    if (x.length < (i - 1) / (spf) + t) {
                        t--;
                    } else {

                        tmp = Arrays.copyOfRange(x, (i - 1) / (spf),
                                (i - 1) / (spf) + t);
                        try {
                            c = nc.calcNaturalCubic(r, tmp);
                            break;
                        } catch (ArrayIndexOutOfBoundsException e) {
                            r--;
                        }

                    }
                }

                for (double j = 0; j <= spf; j++) {

                    timeStep = Utils.timeStep((i - 1) / (spf), x);

                    Format formatter;
                    formatter = new SimpleDateFormat("yyyy-MM-dd");

                    String newDate = formatter.format(new Date(x[(i - 1) /
                            (spf)].getDate().getTime() + ((long) j + 1) *
                            timeStep / (spf)));
                    formatter = new SimpleDateFormat("HH:mm:ss");
                    String newTime = formatter.format(new Date(x[(i - 1) /
                            (spf)].getTime().getTime() + ((long) j + 1) *
```

```
                        timeStep / (spf)));
                out[i + (int) j] = new Record(c[0].eval((j + 1) / (spf + 1)),
                        newDate, newTime);
            }
            i += spf - 1;
        }
    }
    return resampleP(out, n / (spf));
}
```

Forecasting

Questa operazione viene eseguita su i dati in tempo reale.

Utilizzando un set di 6 Record viene fatta una previsione a breve termine.

```
public static Record forecastValue(Record[] values) {
    Cubic[] c;
    NatCubic nc = new NatCubic();
    Record[] val = new Record[6];

    int y = 0;
    int step = 0;
    int pos = values.length - 1;
    val[y] = values[values.length - 1];

    //Partendo dall'ultimo valore, prendo i valori precedenticon intevalli
    //maggiori di forecstTimestep
    for (int i = 0; i < values.length; i++) {
        if (Utils.timeStep(values[values.length - i - 2],
                values[pos]) >= forecstTimestep) {
            val[y++] = values[values.length - i - 2];
            step = i;
            pos = values.length - i - 2;
        }
        if (y >= val.length) {
            break;
        }
    }

    // una volta creato l'array di valori la Spline Cubica Naturale
    c = nc.calcNaturalCubic(val.length - 1, val);
    Format formatter;

    //Creo la stringa contenente il valore temporale del dato previsto
    //con formato HH:mm:ss
    formatter = new SimpleDateFormat("HH:mm:ss");
    String newTime = formatter.format(new Date(System.currentTimeMillis()));
    int j = 0;
    int i = 0;
    Long day = 0l;

    while (true) {
        try {
            newTime = formatter.format(
                    new Date(values[values.length - 1 - j].getTime().getTime()
                    + forecstTimestep));
            break;
        } catch (NullPointerException e) {
            j++;
        } catch (ArrayIndexOutOfBoundsException e) {
            break;
        }
    }

    //Creo un'altra stringa contenente il valore temporale del dato previsto
    //con formato yyyy-MM-dd
    formatter = new SimpleDateFormat("yyyy-MM-dd");
    String newDate = formatter.format(new Date(System.currentTimeMillis()));
    while (true) {
        try {
            newDate = formatter.format(
                    new Date(values[values.length - 1 - i].getDate().getTime()
                    + day));
            break;
        } catch (NullPointerException e) {
            i++;
        } catch (ArrayIndexOutOfBoundsException e) {
            i++;
            break;
        }
    }

    //ritorno il nuovo oggetto Record previsto
    return new Record(c[0].eval(interCoefficient), newDate, newTime);
}
```

Per decidere quanto lunga potrà essere la previsione basterà scegliere un opportuno valore per la variabile forecstTimestep.

Per avere più di una previsione è stato impkementato il metodo

```
public static Record[] multipleForecast(Record[] values) {
        Record[] newVals = new Record[3];
        forecstTimestep = 0;
        for (int i = 0; i < newVals.length; i++) {
            forecstTimestep += 300000;
            newVals[i] = ForecastData.forecastValue(values);
        }
        return newVals;
    }
```

Questo metodo chiama 3 volte il metodo per la previsione dei valori, con valori di previsione di 5, 10 e 15 minuti.

3 Conclusioni e Sviluppi futuri

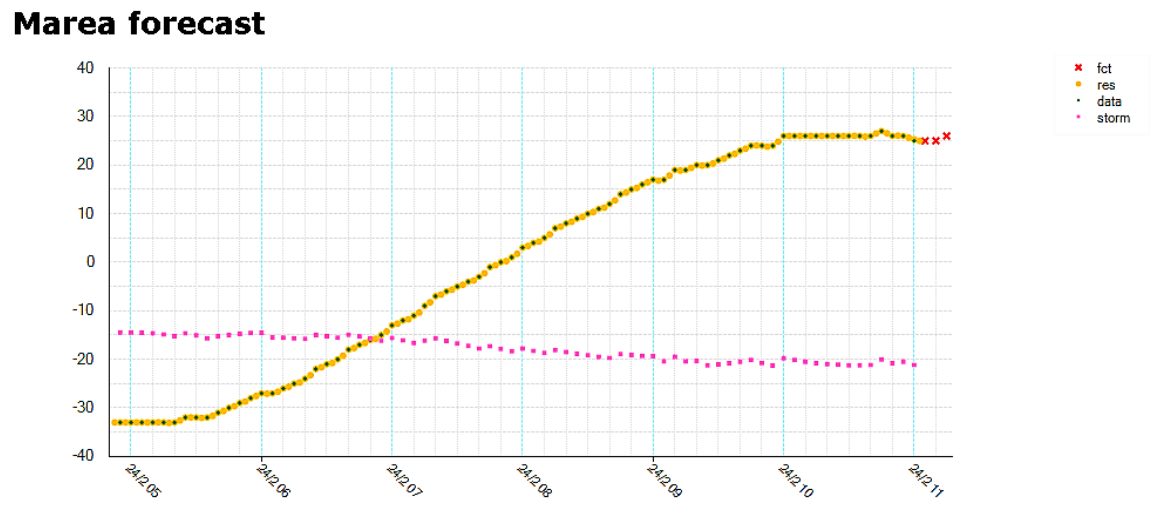


FIGURA 16 GRAFICO DEI DATI PRODOTTI

L'immagine propone il grafico dei risultati finali, plottati tramite browser web dal sito di Thetis, dove si possono vedere in verde i dati originale, in giallo i dati ricampionati, in rosa il contributo metereologico e in rosso la previsione di marea a 5, 10 e 15 minuti in avanti.

Il software, sviluppato con approccio prototipale, pur producendo dati accettabile può avere ampli margini di sviluppo, come l'analisi autoregressiva per la previsione.

Posso ritenermi pienamente soddisfatto del tirocinio svolto in Thetis, grazie al quale ho potuto prender mano con le varie problematiche relative all'organizzazione lavorativa per la realizzazione di un software, inoltre avuto la possibilità capire l'organizzazione di una grande azienda e la suddivisione dei vari compiti. Quest esperienza mi ha fatto approfondire la mia conoscenza sulla programmazione Java, soprattutto tramite l'interfacciamento con un database.

La maggiore difficoltà affrontata è stata principalmente nel dover sviluppare in nel tempo del tirocinio, dovendo sia impementare il software, sia studiare la teoria per poterlo sviluppare.

Bibliografia

Roger S. Pressman, *Principi di Ingegneria del software* – McGraw-Hill Companies, 2000

Cay Horstmann, *Concetti di informatica e fondamenti di Java* – Apogeo, 2010

Daniele Bocchicchio, *C# 4* –Hoepli, 2010

Sheldon Ross, *Calcolo delle probabilità* – Apogeo, 2004

Wikipedia, L'enciclopedia libera - www.wikipedia.org

Thetis, gestione progetti e conoscenza - www.thetis.it

Consorzio Venezia Nuova - www.consorziovenezianuova.com

Html - www.html.it

Comune di Venezia - www.comune.venezia.it