UNIVERSITA' DEGLI STUDI DI CATANIA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

GRILLO STEFANIA

SVILUPPO DI UN'APPLICAZIONE WEB-BASED PER IL MONITORAGGIO DI DEPOSITI DI SCORIE RADIOATTIVE

ELABORATO FINALE

RELATORE:
Prof.ssa V. CARCHIOLO
CORRELATORE:
Prof. P. FINOCCHIARO
Dott. Ing. G. VECCHIO

ANNO ACCADEMICO 2011/2012

RINGRAZIAMENTI

Ringrazio per il grande aiuto fornitomi in questi mesi di lavoro, il professor Paolo Finocchiaro e l'ingegner Vecchio, e la mia relatrice prof.ssa V. Carchiolo.

Ringrazio tutti coloro che in questi anni mi sono stati accanto, nel bene e nel male, e mi hanno sempre sostenuto. La mia tesi è dedicata a loro, la mia famiglia, il mio ragazzo e i miei amici e colleghi di studio e non solo. Grazie a loro il mio percorso universitario non è stato solo pieno di ore di studio, ma anche di momenti di relax e divertimento che rimarranno nel mio cuore ora e sempre.

INDICE DEI CONTENUTI

1 INTRODUZIONE

1.1 AMBITO DI SVILUPPO

Il lavoro di tesi è stato svolto presso l'Istituto Nazionale di Fisica Nucleare nella sede di Catania, nell'ambito del recupero di informazioni riguardanti lo smaltimento di rifiuti radioattivi e di eventuali condizioni di sicurezza dei depositi di scorie attualmente presenti in Italia e non solo.

Le scorie radioattive[1] vengono prodotte in seguito a dei processi nucleari, come ad esempio reazioni di fissione, usate nei processi industriali e in campo medico. A differenza dei rifiuti chimici, la radioattività diminuisce col tempo, anche se sfortunatamente lo smaltimento potrebbe essere più difficoltoso del previsto e richiedere oltre che dei tempi prolungati dei mezzi specifici per lo smaltimento.

A questo proposito possono essere individuati dei livelli di classificazione in base alle difficoltà e al tempo necessario per smaltirli:

- HLW-High Level Waste: livello alto
- LILW-Low and Intermediate Level Waste: livello basso e intermedio

Sistemi di monitoraggio di scorie radioattive sono, ovviamente, già presenti nei depositi delle centrali nucleari, ma fino ad ora si è trattato soltanto di controlli di tipo ambientale, mentre nel monitoraggio proposto nel progetto dell'INFN di Catania, si andrebbe a controllare più nello specifico ogni singolo bidone. A livello tecnico sarebbe già possibile realizzare un controllo di questo tipo, ma il tutto richiederebbe una spesa

eccessiva. Gli strumenti utilizzati attualmente necessiterebbero, infatti, una rete molto fitta di rilevatori che potesse controllare interamente i bidoni o comunque i vari contenitori di rifiuti nucleari; cosa che richiederebbe un numero molto elevato di rilevatori, essendo questi abbastanza piccoli. Inoltre il collegamento ad un sistema centrale risulterebbe difficoltoso, dal momento che questi rilevatori non sono predisposti per una connessione veloce e soprattutto economica.

Le principali caratteristiche che un sistema di sensori di questo tipo deve avere, sono:

- Una rete di rilevatori installata in ogni piattaforma
- Flessibilità e robustezza meccanica tale da permettere una modifica della forma a seconda della diversa disposizione, senza necessariamente creare dei sensori appositi per ognuna
- Forma e misura dei sensori lineari devono essere di almeno 1-2 metri in modo da estendersi per intero almeno su una delle dimensioni del bidone
- Alta sensibilità
- Affidabilità
- Basso costo

Il sistema dovrebbe essere in grado di controllare, tramite un qualche strumento robotico, lo stato dei bidoni e tenere traccia via via delle misurazioni effettuate. Inoltre, dovrebbe permettere di capire se determinate fibre o sensori hanno cessato di funzionare ed eventualmente sostituirle con altre funzionanti.

Per l'immagazzinamento delle informazioni, è vantaggioso l'utilizzo di una ridondanza nei collegamenti tra i sensori e il

sistema di controllo centrale, per evitare che malfunzionamenti dei cavi impediscano la trasmissione di dati. E', inoltre, necessario l'uso di batterie, sfruttato nel caso in cui si possa verificare una temporanea assenza di alimentazione elettrica.

E' altresì indispensabile la presenza di un software di alto livello che permetta, tramite una *graphical user interface*, di interagire con l'hardware che si occupa dell'acquisizione dei dati e con un database che tenga traccia dei dati storici via via rilevati dai sensori. Per questioni di sicurezza, anche per il database, contenente le informazioni rilevate, si consiglia una ridondanza nel salvataggio dei dati, scegliendo preventivamente di memorizzarli, in molteplice copia, in memorie di massa differenti. Le informazioni in esso contenute, devono poter essere accessibili in ogni momento, in altre parole il database deve poter essere interrogato dall'utente nel momento in cui sia necessario estrarre informazioni riguardanti una data postazione, o un singolo bidone di una piattaforma, e il tutto deve poter essere fatto tramite il software sopra citato, dotato di una GUI che sia *user-friendly*. Così strutturato, il sistema può dare un utile contributo alla risoluzione di errori legati all'inefficienza dell'uomo, a guasti di una qualche attrezzatura o ad operazioni non andate a buon fine per qualsivoglia motivo.

1.2 IL PROGETTO DMNR

Il progetto DMNR[1] (Detector Mesh for Nuclear Repositories) ha da poco iniziato a studiare la possibilità di un sistema integrato per il monitoraggio on-line sul campo, fornendo una mappa tridimensionale della radioattività prodotta dai rifiuti. Ad oggi, esistono sul mercato delle soluzioni che potrebbero essere adottate per questo scopo (come i contatori

Geiger-Muller), ma che risulterebbero avere un costo proibitivo al momento dell'applicazione sul campo. Il DMNR è invece una soluzione economica e quindi innovativa che si basa sull'utilizzo di sensori realizzati con materiale a basso costo collegati ad una elettronica di front-end anch'essa dai prezzi contenuti. Il sistema sfrutta la potenzialità delle fibre ottiche di essere un ottimo mezzo trasmissivo per inviare a dei microcontrollori FPGA le informazioni rilevate dai bidoni cui i sensori ottici sono applicati. Lo scopo primario del sistema è ricavare informazioni utili come la temperatura, i "conteggi", il livello di attività per rappresentare l'andamento della radioattività di ogni contenitore.

Pertanto la soluzione proposta in questo progetto, si basa su elementi efficienti e a basso costo (sia per la parte sensoristica, che per quella elettronica).

1.2.1 OBIETTIVI DEL PROGETTO

L'obiettivo che il team dell'INFN di Catania si è posto, è stato quello di realizzare un prototipo di sistema che più si avvicini a quello descritto prima, e che quindi vada a rispecchiare al meglio le caratteristiche principali richieste da un sistema di monitoraggio online. Il sistema suddetto, deve avere come scopo principale, quindi, quello di andare a verificare lo stato dei contenitori e di andare a registrare queste informazioni in modo che siano disponibili per i controlli che periodicamente devono essere fatti all'interno di un deposito di scorie radioattive.

In particolar modo vediamo illustrate le caratteristiche di cui al punto precedente, relative, stavolta, al progetto DMNR:

Rete di rilevatori installata nella piattaforma	**Sì**
Robustezza e flessibilità	Sì, grazie alle fibre ottiche
Forma e misura	Sì
Efficienza geometrica	Sì, le fibre possono essere facilmente modificate
Alta sensibilità	Sì
Affidabilità	Sì, il sistema è molto semplice
Basso costo	Sì

Tabella 1: Caratteristiche progetto DMNR

Si noti come il progetto in esame va a rispecchiare le richieste generali di un sistema di controllo di depositi nucleari.

1.2.2 Le Soluzioni adottate

Come già accennato, si è scelto di utilizzare come sistema di rilevatori, che rispetti la lista generale di caratteristiche richieste, delle fibre ottiche. In modo da sfruttare la potenzialità di queste ultime di essere un ottimo mezzo trasmissivo per inviare a dei microcontrollori FPGA le informazioni rilevate dai bidoni cui i sensori ottici sono applicati.

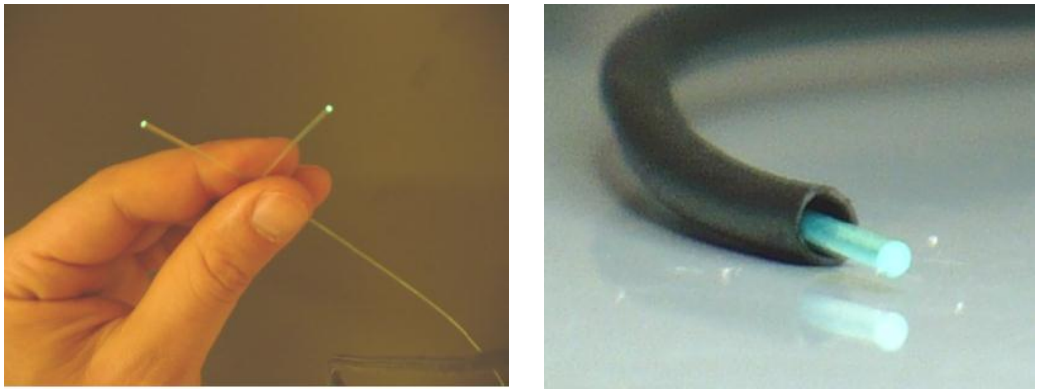


Figura 1 Fibra ottica utilizzata. A destra è raffigurata la fibra con un involucro isolante.

Nel prototipo proposto dall'INFN di Catania, è stato scelto di lavorare con le fibre ottiche perché in grado di essere manipolate con estrema facilità. E' possibile, infatti, adattarle sia alla forma del recipiente da monitorare, sia alla lunghezza dello stesso. Tutto questo grazie alla grande flessibilità delle fibre ottiche e alla caratteristica di mantenere intatte le proprietà intrinseche, indipendentemente dalla lunghezza del cavo, rendendo possibile un accorciamento diretto della fibra a seconda delle esigenze.

Ogni rilevatore è caratterizzato da una fibra ottica di 1-2 m di lunghezza e 1 mm di diametro, adattata alla circonferenza e alla lunghezza del bidone (*Figura 2*).

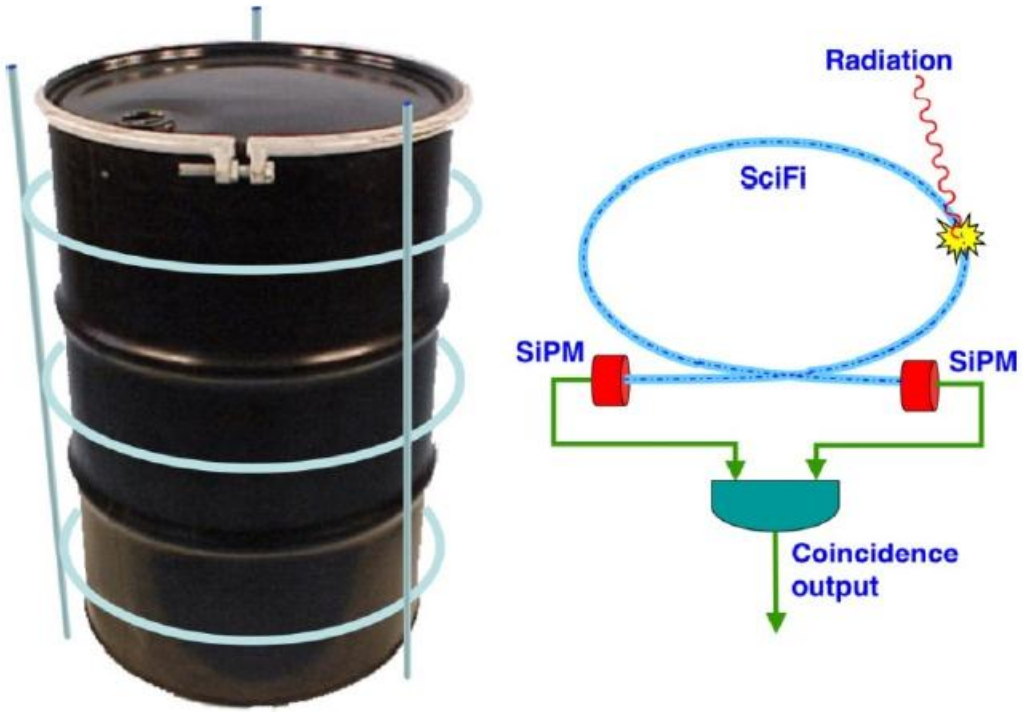


Figura 2 Possibile disposizione delle fibre attorno ad un bidone di scorie radioattive.

Questa particolare rete di rilevatori si occupa dell'acquisizione dei dati al più basso livello, come già accennato, queste informazioni saranno poi trasmesse ad un microprocessore FPGA che avrà il compito di elaborarle e salvarle

in un formato tale da essere poi letto, modificato e immagazzinato da un sistema di front-end dotato di GUI. In quest'ultimo *step* verrà realizzato un software, che, collegato alla rete, riuscirà ad interagire col Database contenente le informazioni, e soprattutto riuscirà a gestire l'immagazzinamento dei dati di basso livello, cosa che costituisce lo scopo ultimo del progetto.

Lo schema generale di questi collegamenti viene mostrato in *Figura 3*.

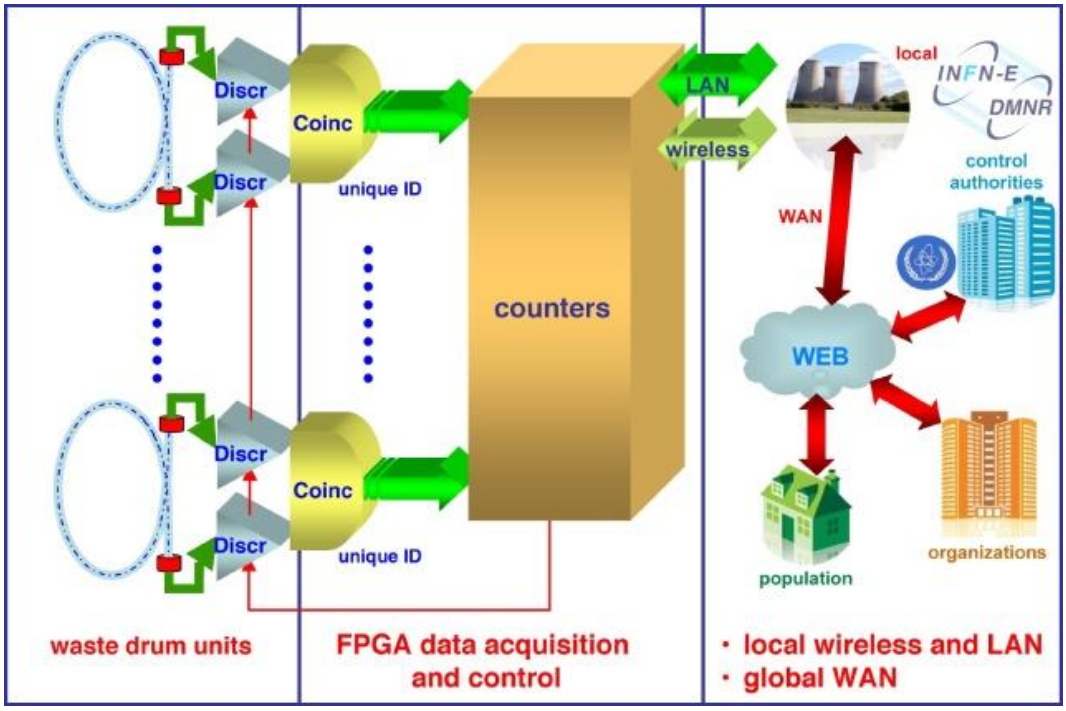


Figura 3 Schema generale di un sistema di monitoraggio dell'acquisizione dati e trasmissione.

Come si nota dal prospetto, lo *step* finale è rappresentato dal *sitoweb*, accessibile da qualsiasi browser; tale applicazione permette l'interazione con il database, e soprattutto è in grado di elaborare i file creati dal microprocessore, e di memorizzarli. Di questo ci siamo occupati col nostro team.

Verranno, quindi, di seguito descritte le specifiche di front-end dell'applicazione.

2 SPECIFICHE DELL'APPLICAZIONE

Le specifiche del servizio offerto dall'applicazione rispondono all'esigenza di dover elaborare, analizzare, immagazzinare e presentare dati provenienti da un sistema di sensori preposti alla raccolta d'informazioni su bidoni contenenti materiale radioattivo. L'obiettivo è dunque monitorare l'attività di ogni singolo bidone, la stabilità termica e meccanica e rendere disponibili questi dati in tempo reale. E' necessario dunque immagazzinare queste informazioni in un archivio elettronico e offrire la possibilità di poter consultare in qualsiasi momento dati storici e attuali. E' importante inoltre poter consultare la storia cronologica dell'intero impianto e visualizzare grafici riguardanti l'attività radioattiva di ogni singolo bidone.

2.1 TECNOLOGIE UTILIZZATE

2.1.1 PHP (Hypertext PreProcessor) 5.3.2[2][3] - Php è un linguaggio di *Scripting general-purpose* particolarmente indicato, per la sua alta dinamicità, per lo sviluppo web. L'elaborazione di codice php sul server produce codice html da inviare al browser del client che ne fa richiesta. Dalla versione 5, supporta la programmazione ad oggetti, mentre nelle versioni precedenti era essenzialmente un linguaggio procedurale. Php si integra perfettamente con il DB di Oracle, MySQL e fornisce un'API per interagire con Apache.. La flessibilità di un linguaggio a tipizzazione debole e con supporto alla *OOP (Object Oriented*

Programming), come php, mi ha permesso un veloce apprendimento già nelle prime settimane di lavoro.

2.1.2 PHPMYADMIN 3.3.2[4] - Applicazione scritta in php utile alla gestione del DB MySQL tramite browser. Permette di creare un DB, popolarlo e interrogarlo. Fornisce la possibilità di creare un backup dei dati. In phpMyAdmin l'interazione col DB si differenzia tra utente e amministratore, quest'ultimo tramite interfaccia grafica può gestire gli utenti. Al primo utilizzo abbiamo impostato tutti i privilegi per il nostro *host* così da poter non solo interrogare il DB ma gestire anche la creazione/eliminazione delle tabelle, la modifica della loro struttura interna e soprattutto le relazioni che intercorrono tra di esse. Per poter collegare le tabelle tra loro tramite le chiavi esterne è stato necessario creare delle tabelle di tipo InnoDB. La funzionalità più importante offerta da InnoDB è la possibilità di creare chiavi esterne. Tramite l'interfaccia grafica (*Figura 4*) intuitiva di phpMyAdmin abbiamo creato le relazioni tra le tabelle. Altri aspetti positivi di InnoDB sono le ottime performance in termini di prestazioni e utilizzo della CPU specialmente quando si ha a che fare con una grande quantità di dati. Uno degli inconvenienti è che InnoDB comprime i record molto meno rispetto a MyISAM. Questo significa che la memoria e lo spazio su disco richiesti da InnoDB sono maggiori. Un'altra mancanza di questo tipo di tabelle è l'assenza della possibilità di dichiarare un campo come indice FULLTEXT, cosa che avrebbe permesso ricerche molto precise, ordinando anche i risultati in base al grado di attinenza con la ricerca. Nonostante quindi, le tabelle MyISAM siano caratterizzate da questa particolare ricerca e da una compressione dei file che rende meno pesante il DB,

nel nostro caso, avendo utilizzato una versione di mySQL superiore alla 4, e soprattutto avendo a che fare con relazioni tra le tabelle di tipo n a n, è più utile avere delle tabelle InnoDB.

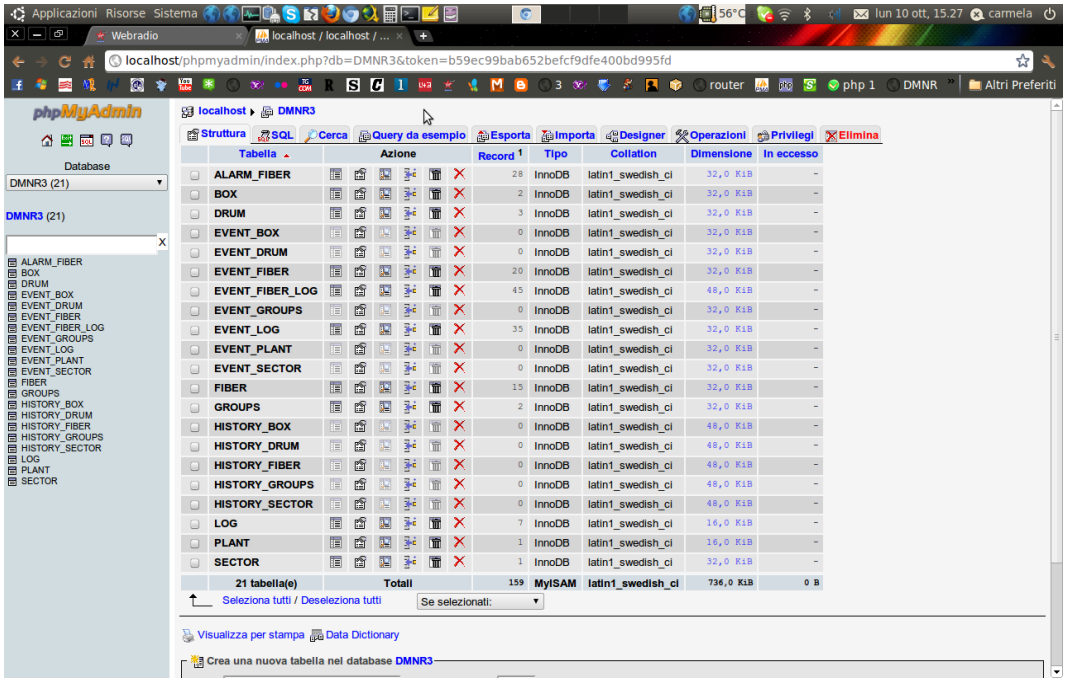


Figura 4 Gestione del database tramite GUI phpMyAdmin

2.1.3 pCHART 2.1.1[5] - Framework gratuito *class oriented* scritto in php che permette di realizzare dei grafici rappresentanti serie di dati singole o multiple. I dati che pChart usa per costruire i grafici possono essere facilmente estrapolati da un DB MySQL tramite *query* ordinarie integrate in php. Un vantaggio importante di questo *framework* è di essere *free* ed *open source*. La velocità di *rendering* nell'ultima versione è stata ottimizzata elaborando un algoritmo di *aliasing* ad alte prestazioni che rende più nitida la visione dei grafici. Nello specifico, per velocità di *rendering*, s'intende il processo di "resa" ovvero di generazione di un'immagine a partire da una descrizione matematica di una scena tridimensionale, interpretata da algoritmi che definiscono il colore di ogni punto dell'immagine. La descrizione è data in un linguaggio o in una

struttura dati e deve contenere la geometria, il punto di vista, le informazioni sulle caratteristiche ottiche delle superfici visibili e sull'illuminazione.

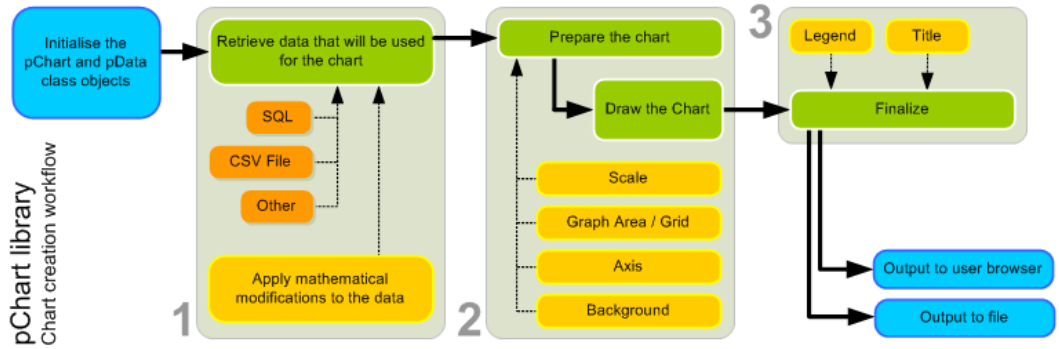


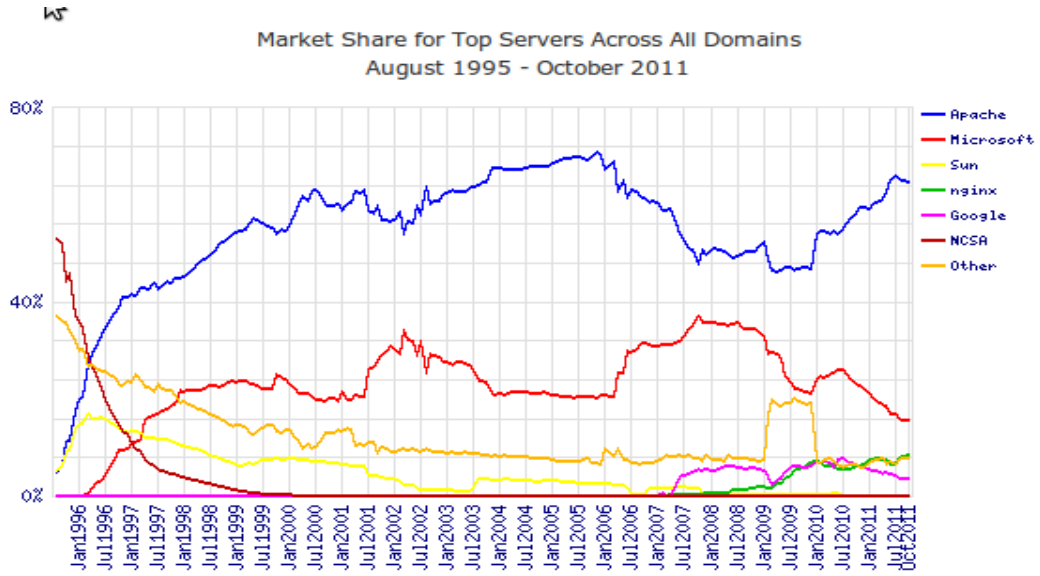
Figura 5 Fasi creazione grafico pChart

I vantaggi di pChart sono che è una libreria stand-alone, che la grafica è altamente personalizzabile e che dà la possibilità di creare numerosi tipi di grafico. Allo stesso tempo presenta degli svantaggi non trascurabili come il fatto che i grafici non sono interattivi ed è necessario dimensionare ogni oggetto contenuto nell'immagine. Quindi l'elaborazione di un grafico è spesso molto laboriosa.

2.1.4 MySQL: 5.1.41-3ubuntu12.10[6] - MySQL è un RDBMS (*relational database management system*) che gira come server di gestione di diversi database. SQL sta per *Structured Query Language*, è un linguaggio d'interrogazione per database progettato per leggere, modificare e gestire dati memorizzati in un sistema di gestione di basi di dati.

2.1.5 APACHE HTTP SERVER Apache/2.2.14 (Ubuntu)[7] - Apache è la piattaforma server più usata al mondo (Stima analizzata su 315,605,335 siti web: a Ottobre 2011 è utilizzata dal 64.67% di essi[8]). Apache HTTP Server offre un server sicuro, efficiente ed estensibile (perché ha

architettura modulare). Fornisce servizi HTTP in accordo con i moderni standard HTTP.



Developer	September 2011	Percent	October 2011	Percent	Change
Apache	315,605,335	65.05%	326,008,432	64.67%	-0.38

Figura 6 Statistica "Market Share for Top Server Across All Domains" da netcraft.com

La soluzione **APACHE + PHP + MYSQL** offre vantaggi quali:

-*economicità:* poiché sono tutti applicativi open source,

-*facile realizzazione:* installazioni rapide e molto semplici, configurazione di default adatta agli scopi e ampia documentazione sul web.

Avendo inoltre a disposizione delle macchine Apple-Mac è stato di grande aiuto l'utilizzo di un software che non richiedeva alcuna configurazione particolare:

2.1.6 MAMP[9] -*Mamp* è una piattaforma open-source, il cui nome è acronimo di:

-**M**ac Os X, il sistema operativo;

-**A**pache, il *web server*;

-**M**ySql, sistema di management del DB;

-**P**hp, Perl o anche Phyton per lo sviluppo web dell'applicazione.

Questa piattaforma fa quindi riferimento ad un set di programmi liberi, comunemente utilizzati insieme, per far girare un sito web dinamico sul sistema operativo Mac Os X della Apple. Mac OS X 10.5 (e successivi) include un motore PHP ed è predisposto a MySQL. Se questi componenti vengono utilizzati insieme, la combinazione di essi è un'ottima piattaforma tecnologica per i server applicativi. Il software Mamp permette di utilizzare tutti questi elementi in un unico pacchetto, sfruttando un'interfaccia utente per avviare i server Apache e MySql e utilizzare phpMyAdmin, fornendo un utile strumento per lo sviluppo di applicazioni web dinamiche.

2.1.7 JAVASCRITP E AJAX (Asynchronous Javascript and XML)[10][11][12] -

Javascript è un linguaggio di *scripting* orientato agli oggetti utilizzato per realizzare pagine web dinamiche. Il codice scritto in Javascript inserito nell'applicazione in php, viene eseguito sul client a differenza di php stesso, cosa che ha il vantaggio di non sovraccaricare il server con le richieste del client. L'impiego di AJAX è stato particolarmente utile nel *rendering* dei grafici, poiché permette che i dati richiesti al server

dal *web browser* vengano scambiati in *background*, evitando l'aggiornamento manuale dell'intera pagina per visualizzare la nuova richiesta del client. Generalmente il flusso di dati è racchiuso in due passaggi alla volta, richiesta dell'utente (Link, form o refresh) e risposta da parte del server, per poi passare, eventualmente, a una nuova richiesta da parte dell'utente. Come si può notare dalla *Figura 7* , il terzo ed

inevitabile incomodo è l'attesa che intercorre tra la richiesta dell'utente e la risposta da parte del server.

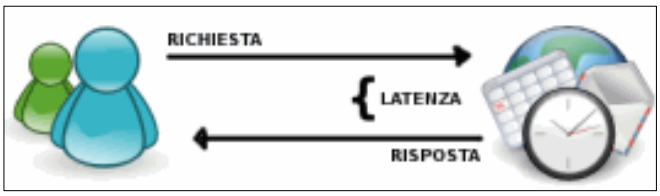


Figura 7 Rappresentazione di una richiesta al server con latenza e relativa risposta.

Con l'aggiunta di AJAX si perde questa linearità e mentre l'utente è all'interno della stessa pagina, le richieste sul server possono essere numerose e completamente indipendenti.

Nulla, infatti, vieta, teoricamente, di effettuare decine di richieste simultanee al server per operazioni differenti, con o senza controllo da parte del navigatore.

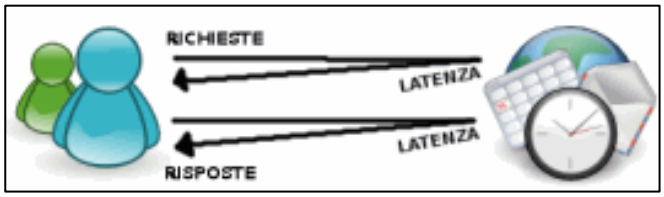


Figura 8 Richieste multiple al server, con latenza e risposta

Ciò che resta del vecchio flusso, il tempo d'attesa, passa spesso in secondo piano ed in molti tipi di interazione è praticamente impercettibile.

Per quanto riguarda invece la parte grafica, sono stati utilizzati:

2.1.8 CSS (Cascading Style Sheets) e HTML (HyperText Markup Language)[13] - Le pagine web che compongono l'applicazione sono dei documenti di tipo XHTML 1.0 e ognuna di esse include un unico CSS che contiene la definizione degli stili di formattazione scelti per i *tag html*. La scelta di usare i fogli di stile mi ha permesso di separare il contenuto delle pagine web dalla loro presentazione grafica, rendendo più veloce la modifica di quest'ultima.

2.2 SCOPO DELL'APPLICAZIONE

L'obiettivo è la realizzazione di un'applicazione web per la gestione dei depositi di rifiuti nucleari. In particolare in questa relazione farò riferimento alla parte di front-end dell'applicazione software preposta ad interfacciarsi e ad interagire con l'utente. Sono state, infatti, utilizzate le tecnologie sopra elencate per creare un'applicazione *web-based* in linguaggio php, che fosse in grado di essere utilizzata da un qualsiasi utente indipendentemente dalla macchina utilizzata per accedervi. Tutto questo è stato possibile grazie agli strumenti usati per realizzarla, per la parte grafica, come già accennato, si è scelto di usare il mezzo sicuramente più diffuso in questo momento per la creazione di sitiweb, HTML e Css. Sicuramente più innovativa è stata la scelta di AJAX, sfruttata in questo progetto per permettere agli utenti di osservare gli andamenti dei conteggi riscontrati nelle misurazioni ed eventualmente prendere nota dei possibili allarmi presenti in un dato bidone, o in una fibra di un bidone specifico. Queste visualizzazioni sono rese possibili dal fatto che è intrinseca nell'applicazione una gestione del DB con mySQL, tramite la quale è possibile dinamicamente andare a verificare l'andamento delle fibre, visualizzando in un grafico i dati aggiornati, effettivamente presenti nel database, il tutto realizzato in un'interfaccia utente *user-friendly*, in modo tale da non richiedere necessariamente conoscenze informatiche per l'utilizzo dell'applicazione.

Verranno a questo punto descritte nel dettaglio le procedure che mi hanno permesso di realizzare l'applicazione, dalla creazione del database, alla stesura del codice per la creazione del *sito-web.*

3 REQUISITI FUNZIONALI

I requisiti funzionali ci descrivono i servizi offerti dal sistema progettato, in particolar modo il progetto presenta un'architettura SW di tipo *Client-Server*. La parte di cui mi sono occupata è il lato Server che fornisce, su richiesta, servizi ad altri sottosistemi, in una rete distribuita. Il nostro sistema prevede una distinzione tra l'utente, che in questo caso si comporta da utente amministratore, e il sistema che svolge le azioni più interne dei casi d'uso. Useremo appunto uno *Use Case Diagram* per descrivere le richieste cui il Server può rispondere.

3.1 ATTORI:

- User;
- System**.**

Processi / Entità	User	System
Grafico	R	R M
Gerarchia	R M I	R M I D
File di log	R M I	R M I D

3.2 CASI D'USO:

- **INSERIMENTO FILE DI LOG:**

-Inserimento, modifica ed elaborazione di file di log;

- **VISIONE GRAFICI E ALLARMI:**

-Visualizzazione grafica, per tipologia, dell'andamento giornaliero dei dati rilevati, ed eventuali dati in allarme;

- **GESTIONE GERARCHIA:**

-Inserimento ed eliminazione di elementi appartenenti all'impianto.

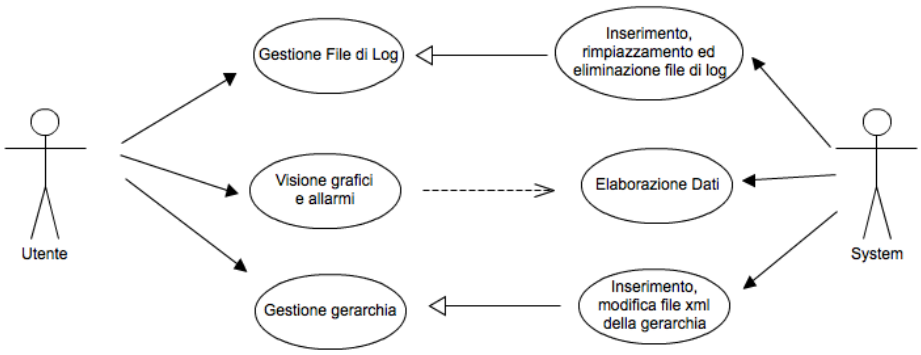


Figura 9 Use Case Diagram generale.

Mi occuperò, adesso, delle due parti dell'applicazione in maniera distinta, ovvero, la parte di interazione con l'utente che è caratterizzata da un'interfaccia utente e la parte riguardante la creazione e gestione del DB, con relativa interpretazione dei dati presenti in un file di log opportunamente compilato dall'elettronica di back- end.

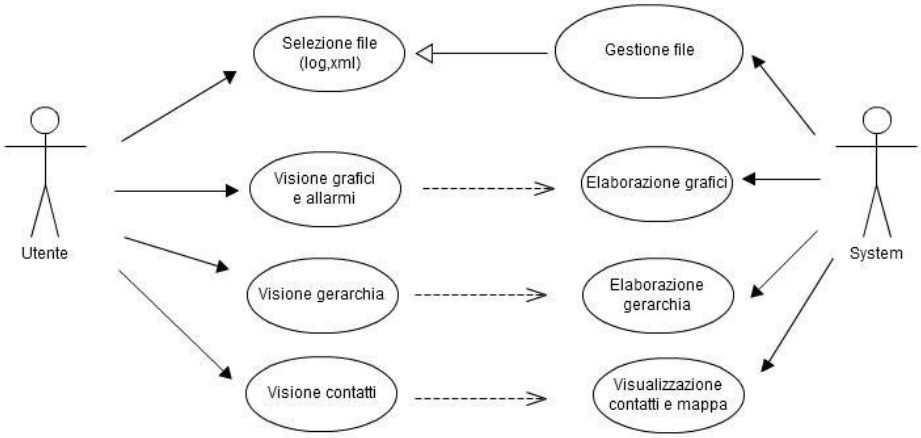


Figura 10 Use Case Diagram della parte grafica

Il sito, dotato di un'interfaccia utente, permette, a chiunque si colleghi, di selezionare un file di log, all'interno del proprio PC, che si voglia inserire, modificare, rimpiazzare o eliminare dal DB. Sarà il sistema ad occuparsi di estrapolare i dati utili e inserirli dopo averli elaborati all'interno del DB.

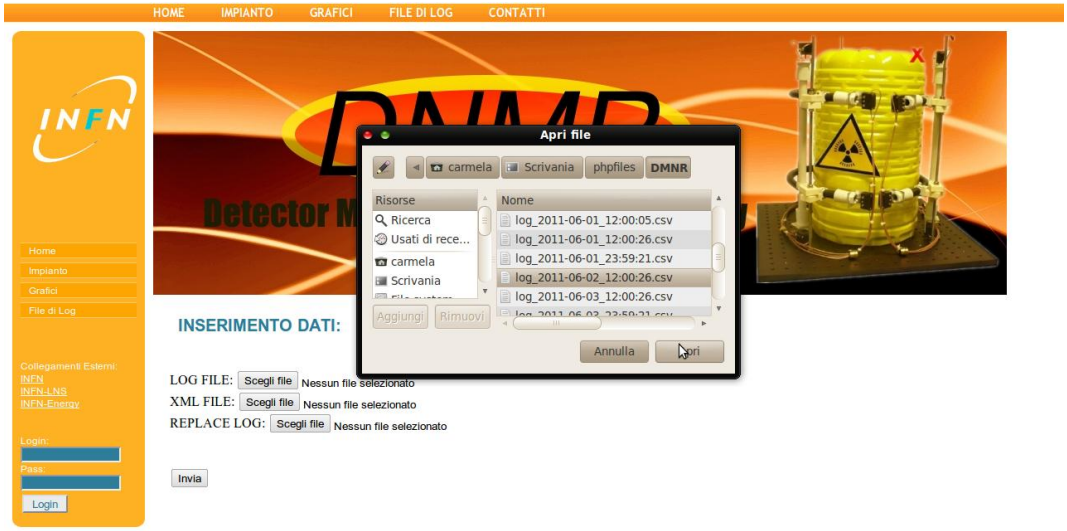


Figura 11 Form per la scelta del file di log o xml

Dal medesimo *form* è inoltre possibile scegliere un file xml, contenente la gerarchia generale dell'impianto su cui si stanno svolgendo i controlli, ed anche in questo caso il sistema si occuperà di interpretare i dati e memorizzarli. L'utente può, altresì, visualizzare l'andamento dei conteggi effettuati. Specificando la tipologia, l'utente può scegliere se visualizzare il grafico di un bidone, o di una sola fibra di un dato bidone.

Figura 12 Form scelta tipologia del grafico

Il sistema in questo caso d'uso si occupa di andare a prendere i valori dei conteggi e rappresentarli graficamente,

verificando se per quel bidone, o per quella fibra, esistono dei dati in allarme da segnalare all'utente, in modo tale che, se richiesto, possa prenderne visione. Nel caso della visualizzazione per bidone (*Figura 13*), se il sistema trova un allarme, sotto il menu laterale a sinistra apparirà una combo-box sulla quale sarà possibile selezionare le fibre per cui si vogliono visualizzare gli allarmi, nel caso in cui non si abbia alcun allarme, tale elemento apparirà vuoto.

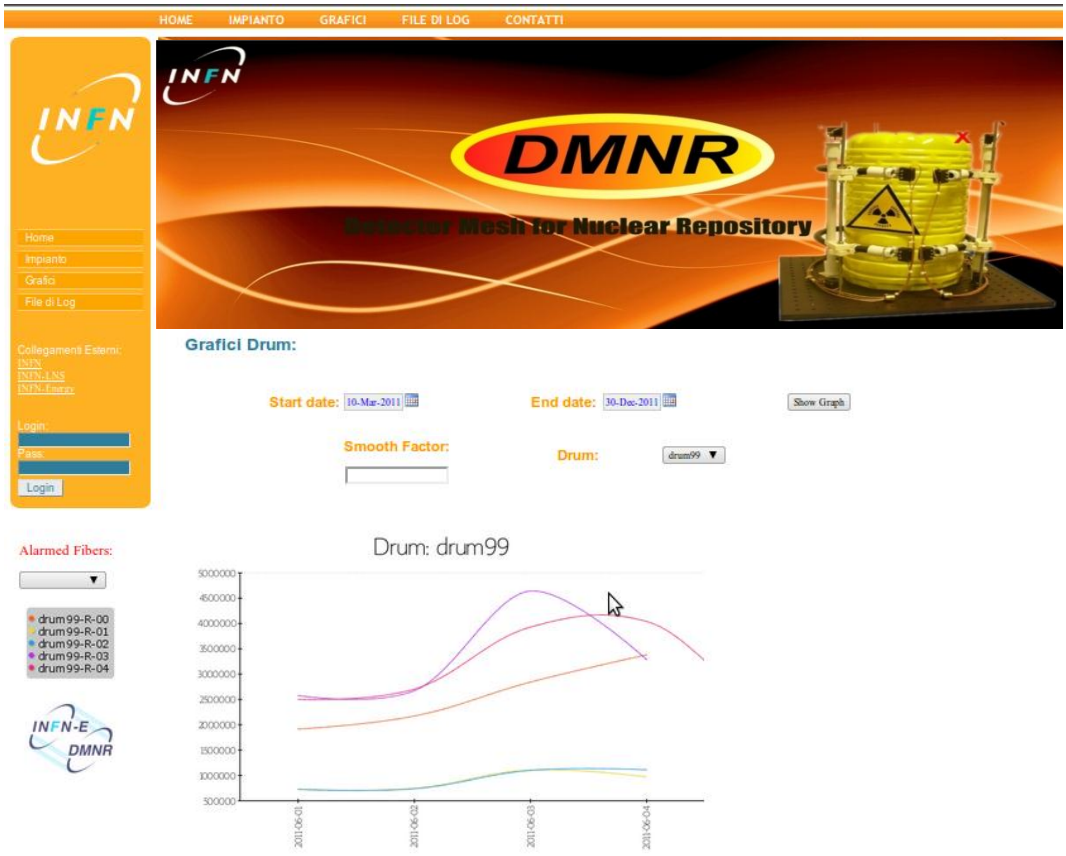


Figura 13 Esempio grafico per bidone

Mentre, nel caso in cui si visualizzi il grafico di un'unica fibra, se essa stessa è in allarme, apparirà un bottone che riporterà alla pagina dei dettagli relativi agli allarmi, se no non apparirà nulla.

La pagina che permette di visionare tutti i dettagli degli allarmi registrati per quella data fibra, fornisce una possibilità di scelta dei dettagli da visualizzare a seconda della data e dell'ora in cui è stato riscontrato un evento allarmante, come si nota dalla *Figura 14*.

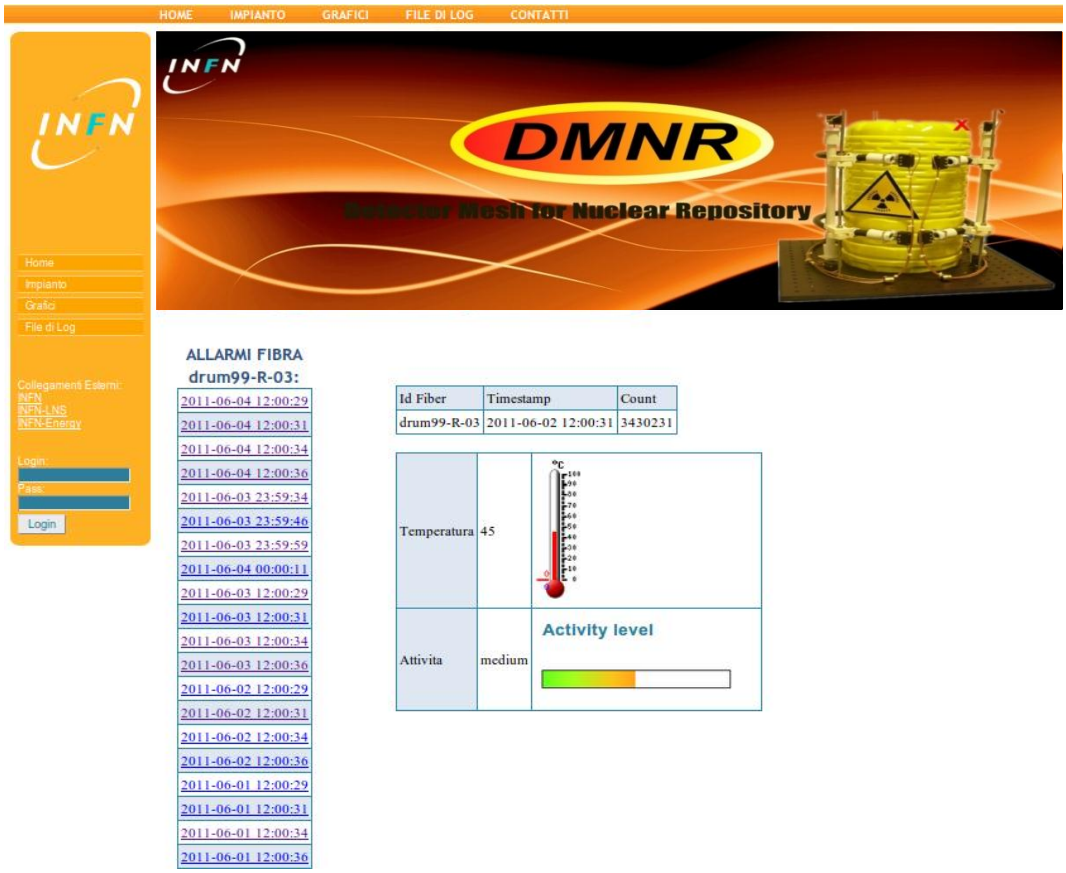


Figura 14 Visualizzazione dettagliata allarmi relativi ad una fibra

E' presente inoltre una pagina dei contatti, molto semplice, che racchiude le informazioni utili all'utente per mettersi in contatto con l'INFN, e una mappa interattiva su cui si può visualizzare l'indirizzo di riferimento. La mappa è stata inserita utilizzando uno script reperito in rete (www.phpclasses.org) che sfrutta una classe *"EasyGoogleMap.class.php"*[16] scritta in php capace di generare il codice HTML e Javascritp necessario per mostrare una mappa su una pagina web con opportuni *markers*.

Tramite questa classe ho potuto definire la grandezza della mappa (altezza e larghezza), il livello di zoom e, inoltre, ho inserito l'indirizzo come informazione ulteriore su una piccola finestra associata alla locazione.

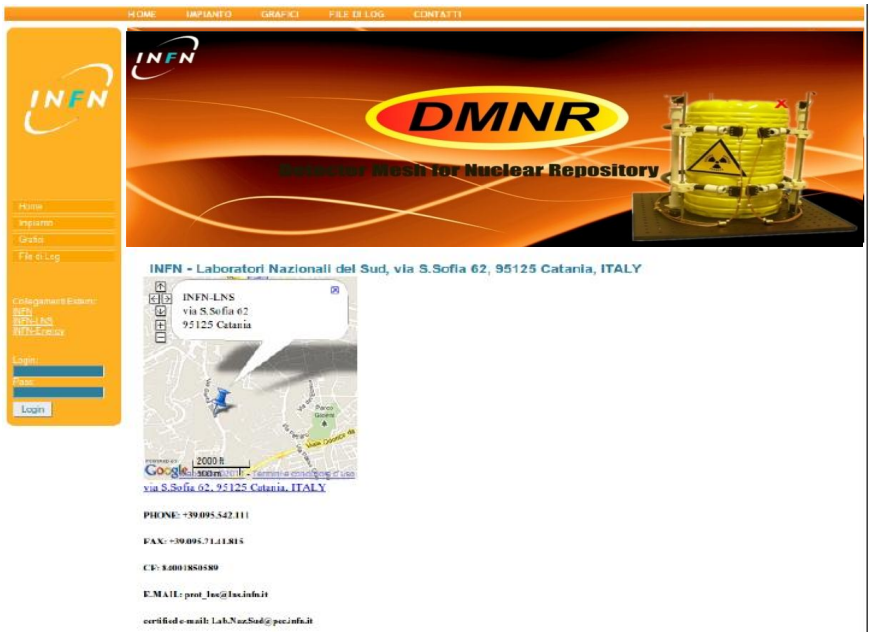


Figura 15 Pagina dei contatti

La pagina "impianto" in cui è possibile visionare la gerarchia, è una pagina di presentazione dell'organizzazione generale di un impianto di stoccaggio di scorie, arricchita da un menù laterale dinamico apri-chiudi realizzato in javascritp che va a rintracciare la gerarchia direttamente dal DB sottostante, per poi visualizzarla in una forma gradevole per l'utente.

Figura 16 Visione impianto compatta

Figura 17 Visione impianto estesa

Adesso verrà trattata la parte della creazione e gestione del database, di cui viene mostrato lo *use case diagram*:

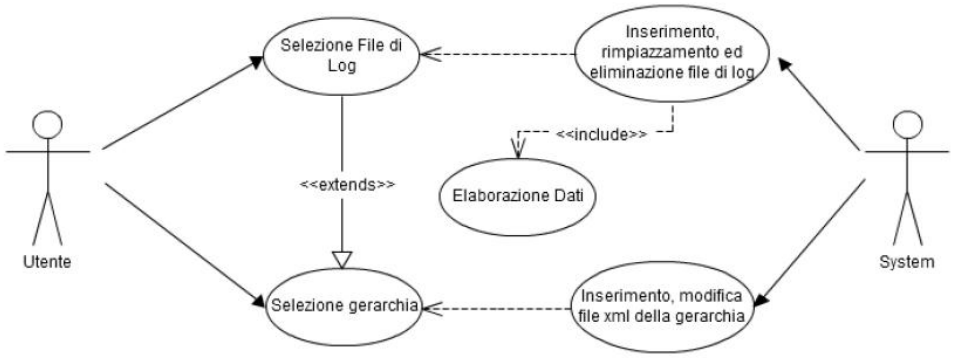


Figura 18 Use Case Diagram gestione database

Il server può, in seguito ad una **selezione** fatta dall'utente sul **file di log** da utilizzare, gestire **l'inserimento, la modifica, il rimpiazzamento o l'eliminazione dal DB** del contenuto suddetto **file.** In particolar modo, il file di Log, si presenta come un file di testo (*Figura 19*) in cui sono memorizzate

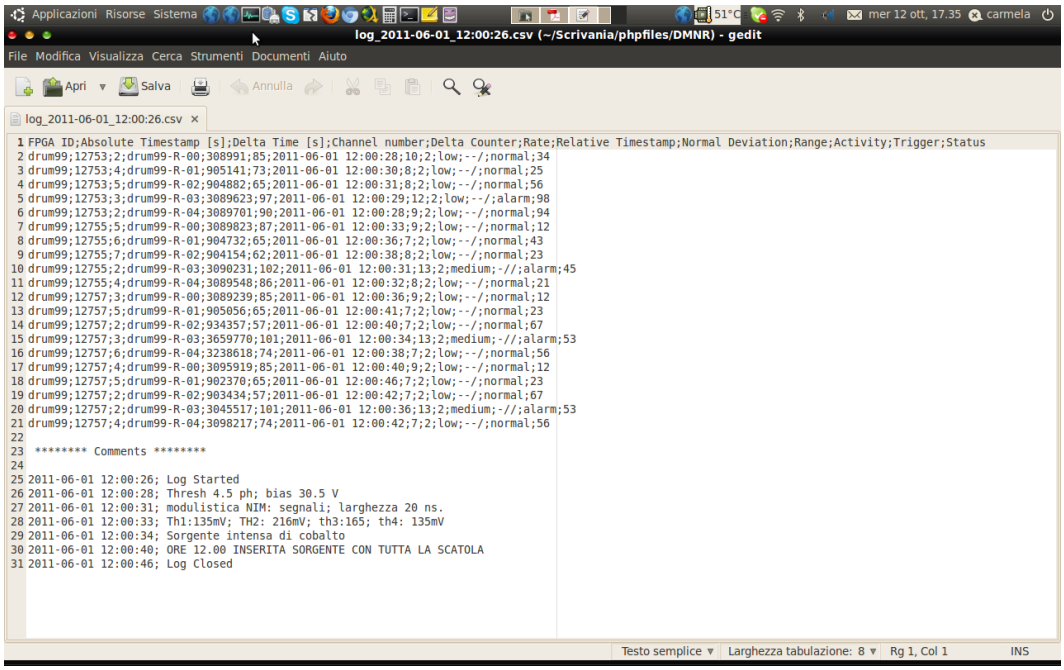


Figura 19 File di log

le informazioni raccolte dall'elettronica di back-end; l'applicazione estrapola tali dati e dopo aver eseguito una media dei valori giornalieri per ogni fibra, li immagazzina all'interno del DB (*vedi* **Appendice A.1**). Il codice in php si occupa di andare a recuperare non solo le informazioni memorizzate all'inizio del file, che costituiscono i veri e propri dati numerici da memorizzare, ma anche quelle testuali scritte alla fine del file. In particolar modo, l'elettronica di *back-end* va a registrare anche delle descrizioni di ciò che è stato riscontrato in determinati orari, sotto forma di commenti, compreso l'ora d'inizio e fine log; tutte queste informazioni verranno poi memorizzate in opportune tabelle del DB chiamate LOG e EVENT_LOG (*vedi* **Appendice A.2** *e* **A.3**).

Come si nota dalla *Figura 18*, un altro caso d'uso riguarda l'inserimento del file *.xml,* contenente la gerarchia dell'impianto presente all'interno del deposito, a partire dalla piattaforma, seguendo poi con il settore, i *box,* i bidoni (*drum) e* le fibre, ciascuno dotato di un identificativo univoco. L'inserimento del file *.xml* è stato considerato necessario poiché, in questo modo, andando a memorizzare i nomi di ciascun componente presente in deposito, è possibile confrontare la veridicità dei dati del file di log, evitando di immagazzinare dettagli relativi a una fibra o ad un bidone inesistente. Anche in questo caso sono presenti delle apposite tabelle nel DB in cui la gerarchia va memorizzata, ovvero: *Fiber*, per le fibre; *Drum*, per i bidoni; *Box*; *Sector*, e infine, *Plant*, per la piattaforma. Per la lettura e interpretazione del file *.xml* sono state adottate diverse configurazioni, scegliendone poi una più consona al linguaggio di programmazione utilizzato, in modo tale da non creare troppi problemi nella raccolta dei dati contenuta in esso.

Viene di seguito riportato un esempio di file *.xml* con la configurazione scelta:

```
<?xml version="1.0"?>
<Plant>
 <Sector ID="sector01">
  <Group ID="group01" Location="0">
    <Box ID="box01" Location="0,0,0">
      <Drum ID="drum99" Location="0,0" RFibres="5" VFibres="5" Queue_size="20">
        <Fibres>
      <Fibre ID="drum99-R-00" Queue_size="20" />
          <Fibre ID="drum99-R-01" Queue_size="20" />
          <Fibre ID="drum99-R-02" Queue_size="20" />
          <Fibre ID="drum99-R-03" Queue_size="20" />
          <Fibre ID="drum99-R-04" Queue_size="20" />
          <Fibre ID="drum99-V-00" Queue_size="20" />
          <Fibre ID="drum99-V-01" Queue_size="20" />
          <Fibre ID="drum99-V-02" Queue_size="20" />
          <Fibre ID="drum99-V-03" Queue_size="20" />
          <Fibre ID="drum99-V-04" Queue_size="20" />
        </Fibres>
      </Drum>
      <Drum ID="0" Location="0,1" RFibres="0" VFibres="0" Queue_size="0">
        <Fibres>
        </Fibres>
      </Drum>
      <Drum ID="0" Location="1,0" RFibres="0" VFibres="0" Queue_size="0">
        <Fibres>
        </Fibres>
      </Drum>
      <Drum ID="0" Location="1,1" RFibres="0" VFibres="0" Queue_size="0">
        <Fibres>
        </Fibres>
      </Drum>
    </Box>
    <Box ID="box02" Location="0,0,1">
      <Drum ID="drum05" Location="0,0" RFibres="10" VFibres="20" Queue_size="20">
        <Fibres>
          <Fibre ID="drum05-R-00" Queue_size="20" />
          <Fibre ID="drum05-R-01" Queue_size="20" />
          <Fibre ID="drum05-R-02" Queue_size="20" />
          <Fibre ID="drum05-R-03" Queue_size="20" />
          <Fibre ID="drum05-R-04" Queue_size="20" />
          <Fibre ID="drum05-R-05" Queue_size="20" />
          <Fibre ID="drum05-R-06" Queue_size="20" />
          <Fibre ID="drum05-R-07" Queue_size="20" />
          <Fibre ID="drum05-R-08" Queue_size="20" />
          <Fibre ID="drum05-R-09" Queue_size="20" />
          <Fibre ID="drum05-V-00" Queue_size="20" />
          <Fibre ID="drum05-V-01" Queue_size="20" />
          <Fibre ID="drum05-V-02" Queue_size="20" />
          <Fibre ID="drum05-V-03" Queue_size="20" />
          <Fibre ID="drum05-V-04" Queue_size="20" />
          <Fibre ID="drum05-V-05" Queue_size="20" />
          <Fibre ID="drum05-V-06" Queue_size="20" />
          <Fibre ID="drum05-V-07" Queue_size="20" />
          <Fibre ID="drum05-V-08" Queue_size="20" />
          <Fibre ID="drum05-V-09" Queue_size="20" />
          <Fibre ID="drum05-R-05" Queue_size="20" />
          <Fibre ID="drum05-R-06" Queue_size="20" />
          <Fibre ID="drum05-R-07" Queue_size="20" />
          <Fibre ID="drum05-R-08" Queue_size="20" />
          <Fibre ID="drum05-R-09" Queue_size="20" />
          <Fibre ID="drum05-V-00" Queue_size="20" />
          <Fibre ID="drum05-V-01" Queue_size="20" />
          <Fibre ID="drum05-V-02" Queue_size="20" />
          <Fibre ID="drum05-V-03" Queue_size="20" />
          <Fibre ID="drum05-V-04" Queue_size="20" />
          <Fibre ID="drum05-V-05" Queue_size="20" />
          <Fibre ID="drum05-V-06" Queue_size="20" />
          <Fibre ID="drum05-V-07" Queue_size="20" />
          <Fibre ID="drum05-V-08" Queue_size="20" />
          <Fibre ID="drum05-V-09" Queue_size="20" />
          <Fibre ID="drum05-V-10" Queue_size="20" />
          <Fibre ID="drum05-V-11" Queue_size="20" />
          <Fibre ID="drum05-V-12" Queue_size="20" />
          <Fibre ID="drum05-V-13" Queue_size="20" />
          <Fibre ID="drum05-V-14" Queue_size="20" />
          <Fibre ID="drum05-V-15" Queue_size="20" />
          <Fibre ID="drum05-V-16" Queue_size="20" />
          <Fibre ID="drum05-V-17" Queue_size="20" />
          <Fibre ID="drum05-V-18" Queue_size="20" />
          <Fibre ID="drum05-V-19" Queue_size="20" />
        </Fibres>
      </Drum>
      <Drum ID="drum01" Location="0,1" RFibres="3" VFibres="0" Queue_size="20">
        <Fibres>
          <Fibre ID="drum01-R-00" Queue_size="20" />
          <Fibre ID="drum01-R-01" Queue_size="20" />
          <Fibre ID="drum01-R-02" Queue_size="20" />
        </Fibres>
      </Drum>
      <Drum ID="drum02" Location="1,0" RFibres="2" VFibres="3" Queue_size="20">
        <Fibres>
          <Fibre ID="drum02-R-00" Queue_size="20" />
          <Fibre ID="drum02-R-01" Queue_size="20" />
          <Fibre ID="drum02-V-00" Queue_size="20" />
          <Fibre ID="drum02-V-01" Queue_size="20" />
          <Fibre ID="drum02-V-02" Queue_size="20" />
        </Fibres>
      </Drum>
      <Drum ID="drum03" Location="1,1" RFibres="2" VFibres="0" Queue_size="20">
        <Fibres>
          <Fibre ID="drum03-R-00" Queue_size="20" />
          <Fibre ID="drum03-R-01" Queue_size="20" />
        </Fibres>
      </Drum>
    </Box>
  </Group>
  <Group ID="group02" Location="1">
  </Group>
 </Sector>
</Plant>
```

Figura 20 File .xml utilizzato per la gerarchia

3.3 SWIMLANE DEI CASI D'USO

3.3.1 VISIONE GRAFICI E ALLARMI

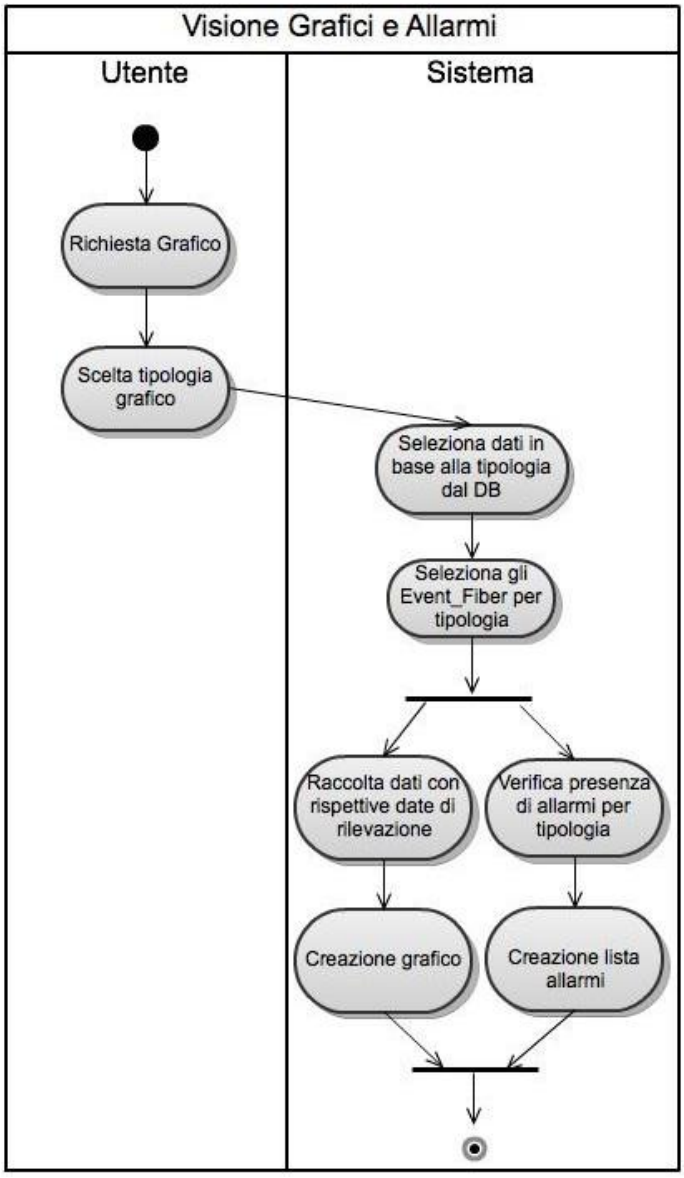


Figura 21 Swimlane Visione grafici e allarmi.

What: L'utente richiede di visualizzare il grafico di una fibra o di un intero bidone;

Who: Utente, Sistema;

What can go wrong: Il sistema non trova, in EVENT_FIBER, la fibra per la quale si vogliono graficare i dettagli e visualizza una finestra di errore.

3.3.2 INSERIMENTO FILE DI LOG

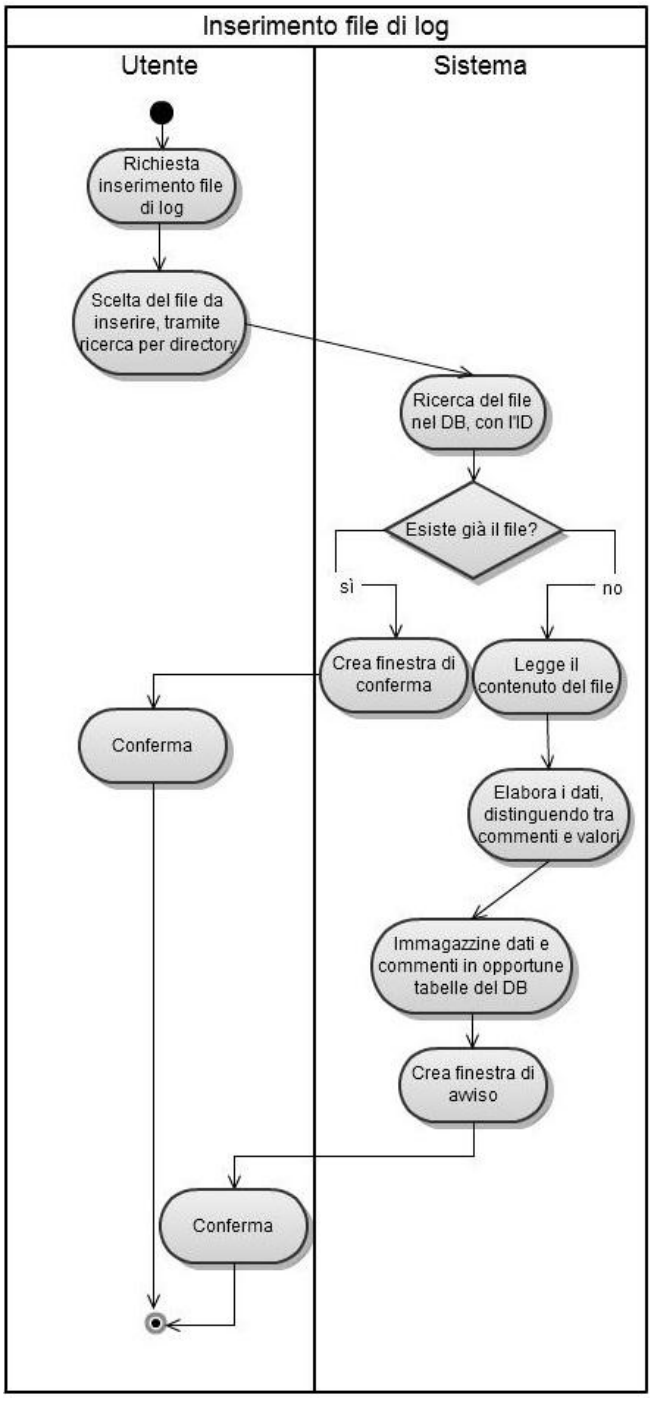


Figura 22 Swimlane Inserimento file di log

What: Inserimento file di log nel database;

Who: Utente, Sistema;

What can go wrong: Il file potrebbe già esistere, errore opportunamente gestito dal programma.

3.3.3 RIMPIAZZAMENTO FILE DI LOG

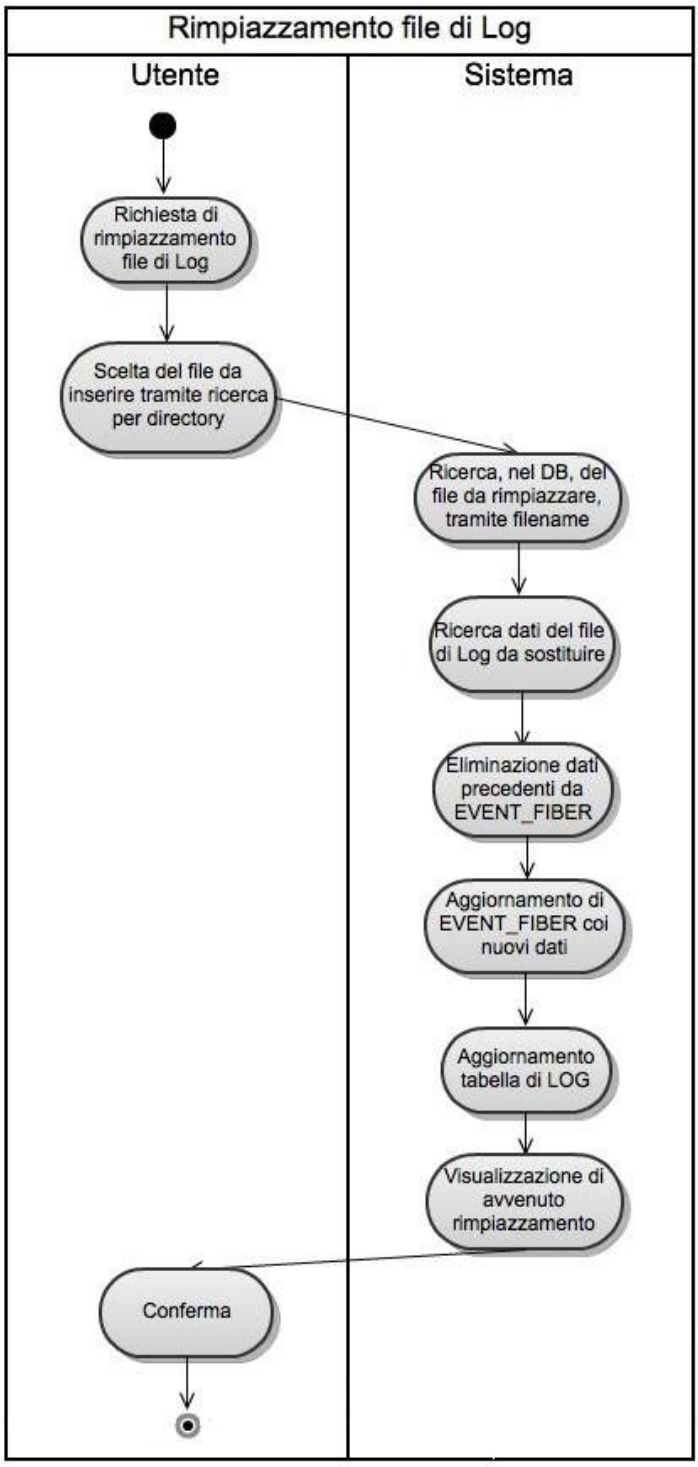


Figura 23 Swimlane Rimpiazzamento file di log.

What: Rimpiazzamento di un file di log;

Who: Utente, Sistema;

What can go wrong: Non viene trovato il file da rimpiazzare.

3.3.4 GESTIONE GERARCHIA

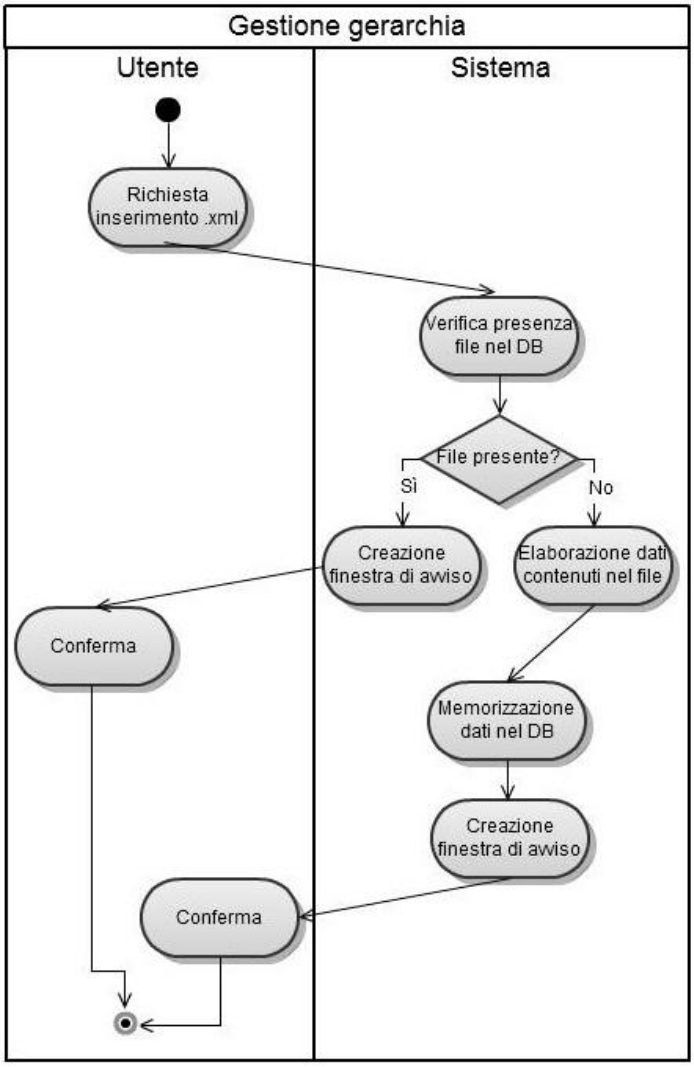


Figura 24 Swimlane Gestione gerarchia

What: Gestione gerarchia;

Who: Utente, Sistema;

What can go wrong: La gerarchia può già esistere nel DB, errore opportunamente gestito dal codice.

4 PROGETTAZIONI DEL DATABASE

Seguono l'*Entity Relationship Model* e le relative tabelle che illustrano la struttura del database in dettaglio. L'ERM può essere concettualmente suddiviso in diversi gruppi di tabelle, ognuna delle quali svolge un compito ben preciso. Le tabelle in viola raccolgono lo storico del sistema, quelle in blu rappresentano gli elementi base della gerarchia i cui eventi in dettaglio sono contenuti nelle tabelle in grigio. L'unica tabella in rosso è quella relativa agli allarmi di una singola fibra.

4.1 ENTITY RELATION MODEL:

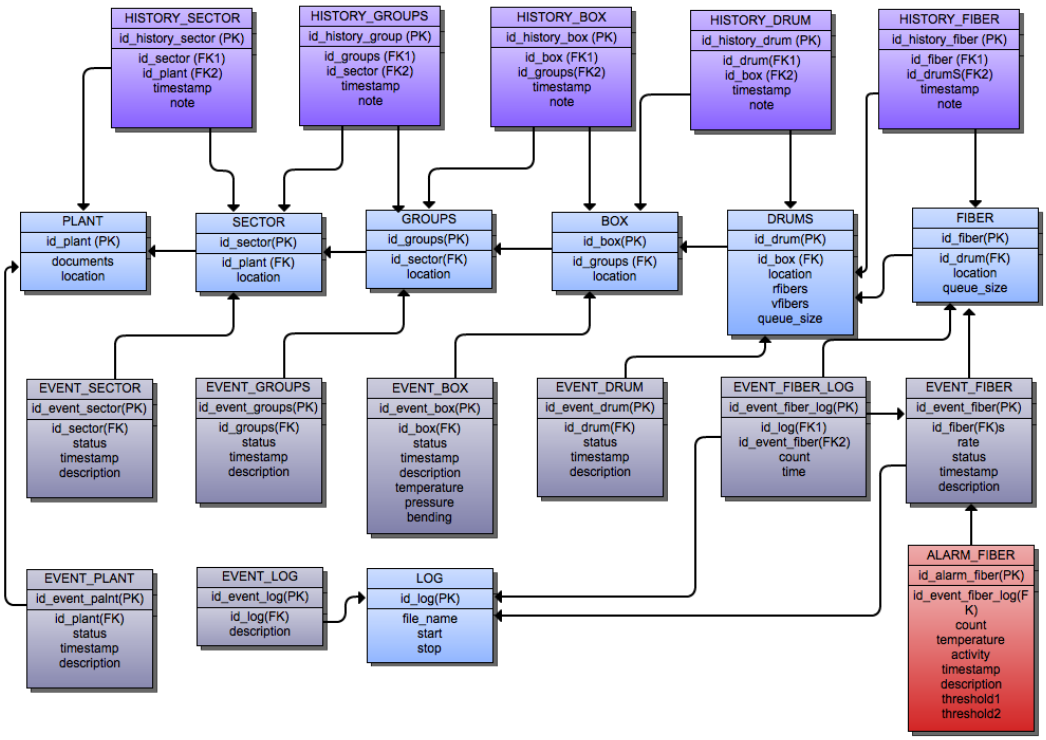


Figura 25 ERM

4.2 TABELLE

ALARM_FIBER

Campo	Tipo	Descrizione
id_alarm_fiber	int(11)	Chiave primaria
id_event_fiber_log	varchar(50)	Foreign key
count	int(11)	
temperature	double	
Activity	varchar(50)	
threshold1	double	
threshold2	double	
timestamp	datetime	
description	varchar(200)	

BOX

Campo	Tipo	Descrizione
id_box	varchar(50)	Chiave primaria
id_groups	varchar(50)	Foreign key
location	varchar(10)	

DRUM

Campo	Tipo	Descrizione
id_drum	varchar(50)	Chiave primaria
id_box	varchar(50)	Foreign key
location	varchar(10)	
rfibres	int(11)	
vfibres	int(11)	
queue_size	int(11)	

EVENT_BOX

Campo	Tipo	Descrizione
id_event_box	int(11)	Chiave primaria
id_box	varchar(50)	Foreign key
status	varchar(50)	
timestamp	datetime	
description	varchar(200)	
temperature	double	
pressure	double	
bending	double	

EVENT_DRUM

Campo	Tipo	Descrizione
id_event_drum	int(11)	Chiave primaria
id_drum	varchar(50)	Foreign key
status	varchar(50)	
timestamp	datetime	
description	varchar(200)	

EVENT_FIBER

Campo	Tipo	Descrizione
id_event_fiber	varchar(50)	Chiave primaria
id_fiber	varchar(50)	Foreign key
rate	double	
status	varchar(50)	
timestamp	datetime	
description	varchar(200)	

EVENT_FIBER_LOG

Campo	Tipo	Descrizione
id_event_fiber_log	varchar(50)	Chiave primaria
id_log	varchar(50)	Foreign key
id_event_fiber	varchar(50)	Foreign key
count	bigint(20)	
time	bigint(20)	

EVENT_GROUPS

Campo	Tipo	Descrizione
id_event_groups	int(11)	Chiave primaria
id_groups	varchar(50)	Foreign key
status	varchar(50)	
timestamp	datetime	
description	varchar(200)	

EVENT_LOG

Campo	Tipo	Descrizione
id_event_log	int(11)	Chiave primaria
id_log	varchar(50)	Foreign key
description	varchar(200)	

EVENT_PLANT

Campo	Tipo	Descrizione
id_event_plant	int(11)	Chiave primaria
id_plant	varchar(50)	Foreign key
Status	varchar(50)	
timestamp	datetime	
description	varchar(200)	

EVENT_SECTOR

Campo	Tipo	Descrizione
id_event_sector	int(11)	Chiave primaria
id_sector	varchar(50)	Foreign key
status	varchar(50)	
timestamp	datetime	
description	varchar(200)	

FIBER

Campo	Tipo	Descrizione
id_fiber	varchar(50)	Chiave primaria
id_drum	varchar(50)	Foreign key
location	varchar(10)	
queue_size	int(11)	

GROUPS

Campo	Tipo	Descrizione
id_groups	varchar(50)	Chiave primaria
id_sector	varchar(50)	Foreign key
location	varchar(10)	

HISTORY_BOX

Campo	Tipo	Descrizione
id_history_box	int(11)	Chiave primaria
id_box	varchar(50)	Foreign key
id_groups	varchar(50)	Foreign key
timestamp	datetime	
note	varchar(200)	

HISTORY_DRUM

Campo	Tipo	Descrizione
id_history_drum	int(11)	Chiave primaria
id_drum	varchar(50)	Foreign key
id_box	varchar(50)	Foreign key
timestamp	datetime	
note	varchar(200)	

HISTORY_FIBER

Campo	Tipo	Descrizione
id_history_fiber	int(11)	Chiave primaria
id_fiber	varchar(50)	Foreign key
id_drum	varchar(50)	Foreign key
timestamp	datetime	
note	varchar(200)	

HISTORY_GROUPS

Campo	Tipo	Descrizione
id_history_groups	int(11)	Chiave primaria
id_groups	varchar(50)	Foreign key
id_sector	varchar(50)	Foreign key
timestamp	datetime	
note	varchar(200)	

HISTORY_SECTOR

Campo	Tipo	Descrizione
id_history_sector	int(11)	Chiave primaria
id_sector	varchar(50)	Foreign key
id_plant	varchar(50)	Foreign key
timestamp	datetime	
note	varchar(200)	

LOG

Campo	Tipo	Descrizione
id_log	varchar(50)	Chiave primaria
file_name	varchar(50)	
start	datetime	
stop	datetime	

PLANT

Campo	Tipo	Descrizione
id_plant	varchar(50)	Chiave primaria
documents	varchar(200)	
location	varchar(200)	

SECTOR

Campo	Tipo	Descrizione
id_sector	varchar(50)	Chiave primaria
id_plant	varchar(50)	Foreign key
location	varchar(50)	

5 ARCHITETTURA DEL SISTEMA

5.1 ARCHITETTURA HARDWARE

L'architettura hardware del sistema è composta dalla parte di back-end (Sensori ed elettronica di controllo) e la parte di front-end (Software per l'acquisizione, analisi e immagazzinamento dei dati). Il lavoro descritto in questo documento riguarda principalmente la progettazione e lo sviluppo del front-end dell'applicazione. Infatti, per descrivere il nostro sistema partiremo da una routine di acquisizione dei dati scritta in C++ che preleva le informazioni dall'elettronica di back-end, seguendo la cadenza di un intervallo di tempo prestabilito. Questi dati vengono poi inviati e mostrati in tempo reale su una console locale (realizzata in Microsoft WPF) tramite un socket in ascolto su una particolare porta del sistema. Dalla routine di acquisizione dei dati, viene realizzato un file di log dove vengono immagazzinati gli stessi dati in tempo reale che mostriamo sulla console. Questi file di log a loro volta verranno memorizzati nel database Oracle MySQL, tramite apposite funzioni dello script php. Quello che andiamo ad immagazzinare non è però il file di log nella sua interezza, ma soltanto un riassunto giornaliero delle fibre che contiene, in modo da non appesantire troppo il database relazionale. La logica di gestione e l'immagazzinamento del log, l'inserimento e il recupero dei dati è quindi interamente contenuta nell'HTTP Server Apache, che espone i servizi verso l'esterno. Per permettere la fruibilità verso l'esterno di tutta questa serie di dati lo script è stato racchiuso all'interno di un'applicazione Web dinamica realizzata tramite l'utilizzo di php e javascript. Essa è accessibile da un qualsiasi browser web.

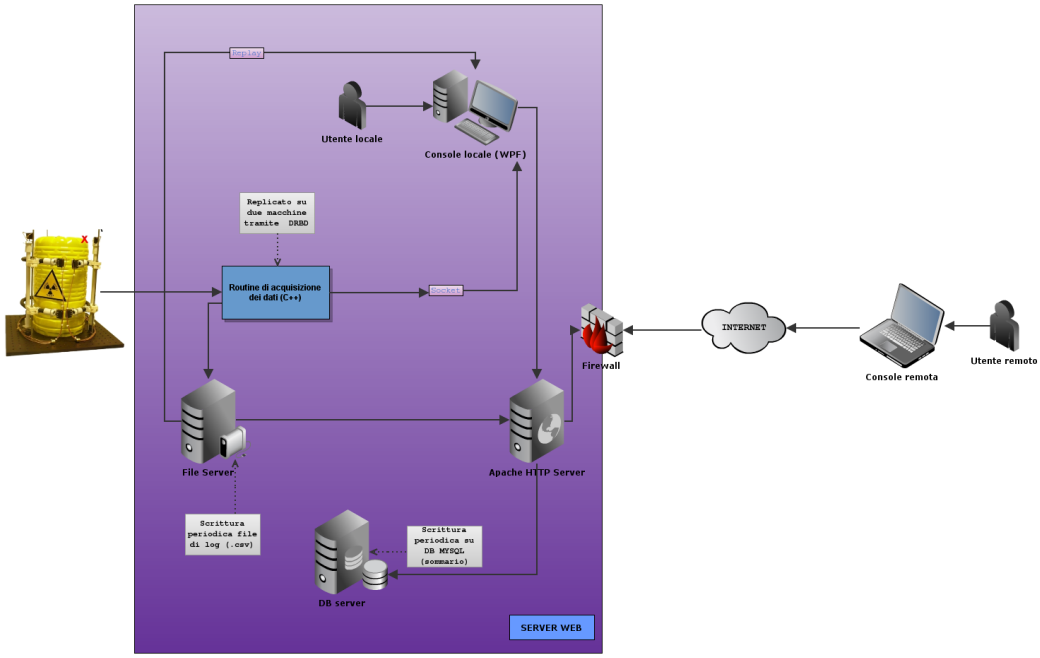


Figura 26 Architettura HW

5.2 ARCHITETTURA SOFTWARE

La logica dell'applicazione si può assimilare a quella classica a livelli. Tali livelli si scambiano dati e svolgono un compito ben preciso, essi sono (*Figura 27*):

- **PRESENTATION LAYER**

- **BUSINESS LOGIC LAYER**

- **DATA ACCESS LAYER**

Il Presentation Layer si occupa di gestire le interazioni con l'utente, raccogliere le sue richieste per poi passarle allo strato sottostante e presentare i servizi offerti da quest'ultimo

all’esterno. Nella nostra applicazione i confini tra Presentation Layer e Business Logic Layer sono debolmente demarcati, dal momento che il linguaggio in questione è php, linguaggio che permette sia di interfacciarsi con l'utente sullo strato più esterno, attraverso l'integrazione con html, che di realizzare dei veri e propri algoritmi di elaborazione più interni. Il Data Access Layer invece risulta ben definito.

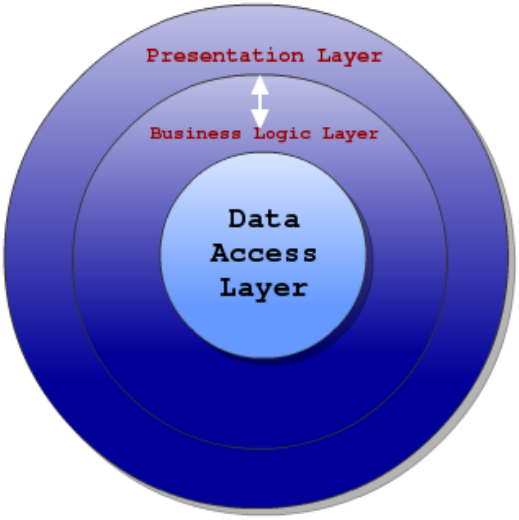


Figura 27 Architettura SW.

In tabella sono presenti alcuni dei file facenti parte di PL e BLL. Per il primo livello vi sono file che si occupano esclusivamente dell'interazione diretta con i file di log e il DB, per poi fornire i dati ricavati da essi ai livelli superiori.

PRESENTATION LAYER	BUSINESS LOGIC LAYER
index.html	index.php
layout.html	indexF.php
ajaxGraph.php	getuserF.php
formLog.php	controlAlarm.php
chooseGraph.php	
showHierarchy.php	
showAlarms.php	
viewDetails.php	

Alcuni dei file dell'applicazione divisi per strato.

6 DIMENSIONAMENTO

6.1 DEFINIZIONE

Il dimensionamento consiste nella definizione delle caratteristiche qualitative e quantitative che le componenti del sistema devono possedere per permettere di realizzare i requisiti informativi, ottimizzando costi/benefici e rispettando le esigenze degli utilizzatori.

6.1.1 PROCESSI DI ARRIVO E DI SERVIZIO

Il processo di arrivo[14] descrive la modalità statistica con cui sono presentate le richieste di servizio al sistema; il modello più comunemente usato è quello esponenziale ovvero si assume che il processo sia di tipo *Poissoniano*.

Il processo di servizio, invece, descrive le modalità con cui sono serviti i pacchetti, ovvero i tempi di occupazione della risorsa trasmissiva di un link (C bit/s).

6.1.2 IL SISTEMA M/M/1

Supponiamo di lavorare con un sistema del tipo rappresentato in figura:

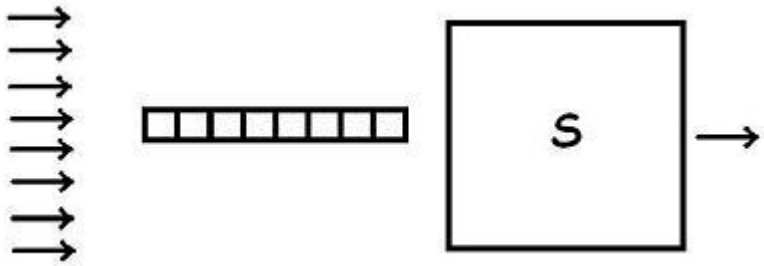


Figura 28 Sistema gestito tramite la teoria delle code.

Questo schema rappresenta una situazione molto frequente nel reale, anche se modellizzata. Ad esempio una

qualsiasi coda alla posta o al supermercato, rappresenta un sistema gestito con la teoria delle code.

Per modello di coda M/M/1[15] si intende un particolare tipo di coda caratterizzato da processi di arrivo esponenziali, un solo server, tempi esponenziali di servizio, coda infinita, nessun abbandono in corso e modalità di servizio *first come first served* (FCFS).

Per tempi esponenziali s'intende un processo di *Poisson* caratterizzato dall'intensità λ. Dove, per processo di *Poisson* s'intende un processo stocastico che simula il manifestarsi di eventi che siano indipendenti l'uno dall'altro e che accadano continuamente nel tempo. I processi di questo tipo rappresentano un esempio di *Catene di Markov* ed è proprio per questo motivo che consideriamo un processo di *Poisson*, per il sistema a coda M/M/1. Possiamo, perciò, dire che i processi di un sistema di questo tipo godono della *proprietà di Markov*, ovvero nel momento in cui si determina il passaggio ad uno stato di sistema, la probabilità di transizione dipende esclusivamente dallo stato di sistema immediatamente precedente e non dal *"come"* si sia giunti a tale stato. Questa proprietà è detta anche *condizione di assenza di memoria*.

6.2 APPLICAZIONE AL PROGETTO

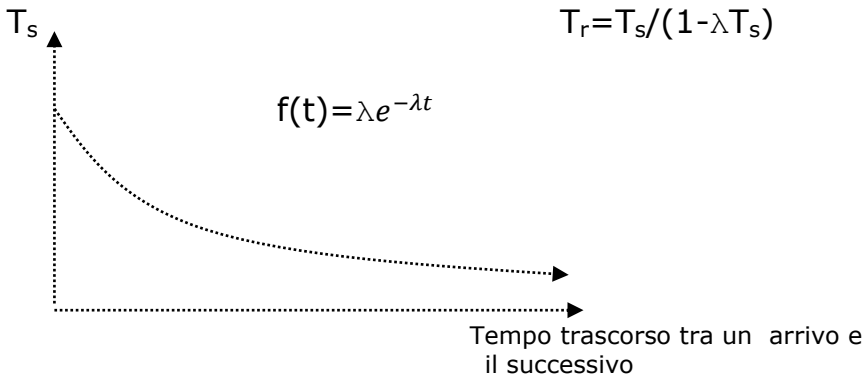
Analizzerò adesso l'applicazione web realizzata, assumendo come ipotesi di lavoro quelle appena descritte, nonché:

1) Sistema in esame di tipo *Markoviano* M/M/1:

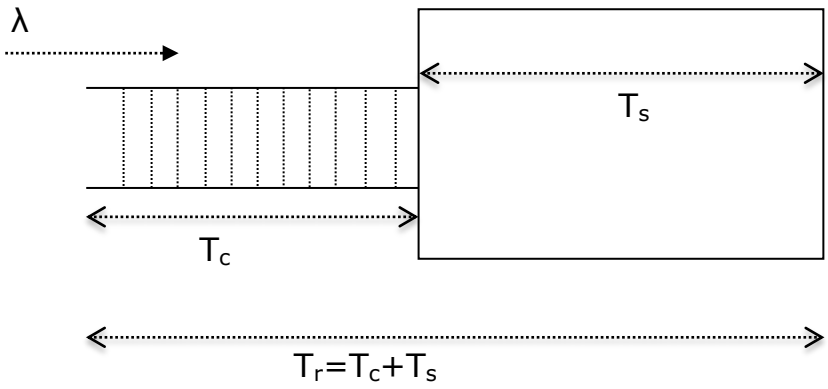
- il flusso d'ingresso (λ) e i tempi di servizio (T_s) hanno distribuzione esponenziale (*Poissoniano*);
- le richieste sono servite da un solo server (n=1);

2) I flussi dei dati uscenti e i tempi di risposta di ogni nodo saranno di tipo *Poissoniano*, cioè avranno distribuzione esponenziale.[1]

La rete di servizi può quindi essere considerata come un sistema M/M/1 il cui funzionamento è rappresentato dal grafico:



Il tempo di risposta, T_r, di cui è stata data la relazione con il tempo di servizio e λ flusso d'ingresso, può essere schematizzato in maniera molto intuitiva dal grafico:



Come s'intuisce dal grafico il tempo di risposta T_r è dato dalla somma del tempo di attesa in coda T_c e del tempo di servizio T_s.

Prima di andare a vedere qual è il tempo di risposta della nostra rete, è opportuno andare a fare delle considerazioni sul flusso in ingresso al sistema, relativo, chiaramente, al client. Non vi è

[1] Teorema di Jackson: *se* *λ d'ingresso e T_s sono di tipo M, allora lo saranno anche λ d'uscita e T_r tempo di risposta.*

alcuna limitazione riguardo agli accessi al sito, che possono eseguire in qualsiasi momento, ma poiché nelle ore notturne, il numero di clienti che accede al sito sarà sicuramente più basso rispetto a quello relativo al resto della giornata, il server potrà gestire facilmente ogni richiesta. Per tale motivo, non consideriamo 24 ore della giornata per i nostri calcoli, ma ipotizziamo che gli accessi avvengano solo tra le 8.00 e le 00.00; consideriamo cioè solo le 16 ore di maggiore traffico. Supporremo infine un numero di ingressi giornalieri pari a 150.

$$\lambda_{UTENTE} = \frac{150 accessi}{960\,\min} = 0{,}156\, accessi/\min$$

$$\rho_{UTENTE} = \frac{0{,}156}{60\,\sec} = 0{,}026 accessi/s$$

I flussi in ingresso nei singoli nodi del sistema quindi sono:

$$\lambda_{HD} = \lambda_{CPU} = \lambda_{WAN} = \lambda_{UTENTE} = 0{,}0026/s$$

Adesso possiamo quindi calcolare il tempo di servizio relativo alla rete che stiamo utilizzando, ovvero consideriamo una rete di tipo WAN *wide area network,* dato che comunque si tratta di una rete utilizzata dall'intera azienda e che quindi potrebbe essere considerata una *wide area*. Per calcolare i dati relativi alla rete WAN, ipotizziamo che la connessione alla rete avvenga tramite un modem a 8Mbps, cioè $8*10^6$bps (bit per secondo), che equivale a una velocità di trasferimento di byte/s pari a:

$$V\hat{e}locit\hat{a}_{WAN} = \frac{8000000 bps}{8 bit} = 1000000 byte/s = 10^6 byte/s$$

Il tempo di servizio è legato alla velocità della rete per via di un coefficiente detto *path length* che consideriamo di 1024

byte, che andremo a moltiplicare per due, in quanto i dati viaggiano nei due sensi (da/verso client).

Esso verrà quindi calcolato nel seguente modo:

$$T_{S\ WAN} = \frac{2 \cdot path\ length}{Velocit\hat{}_{WAN}} = \frac{2 \cdot 1024byte}{10^6 \frac{byte}{s}} \approx 0{,}002s$$

Grazie a questo valore, possiamo calcolare il tempo di risposta sapendo che la relazione che intercorre tra i due:

$$T_{R\ WAN} = \frac{T_{S\ WAN}}{1+(\lambda_{WAN} \cdot T_{S\ WAN})} = \frac{0{,}002\,\mathrm{sec}}{1+(0{,}0026 \cdot 0{,}002)} \approx 0{,}002s$$

Ricaviamo, così, il coefficiente di utilizzazione della WAN (ρ_{wan}) è:

$$\rho_{WAN} = \lambda_{WAN} \cdot T_{S\ WAN} = 0{,}0026 \cdot 0{,}002 = 0{,}5 \cdot 10^{-6}$$

Per la formula finale del tempo di risposta, abbiamo bisogno di calcolare i tempi relativi all'HD ed alla CPU, prendiamo prima di tutto in considerazione l'HD, facciamo delle supposizioni partendo dal caso peggiore, è chiaro che utilizzando degli *hardware* migliori si va anche a migliorare la qualità globale del sistema. Ipotizziamo di avere un HD che abbia le seguenti caratteristiche:

- Tempo di accesso (TSEEK): 9ms
- Tempo di rotazione (TROT): 8ms
- Velocità di trasferimento (VLINEA): 300 MB/s
- Numero di byte che compone un dato (LMSG): 1025 byte.

Assumendo queste caratteristiche il tempo di servizio dell'hard-disk sarà:

$$T_{S\ HD} = T_{SEEK} + \frac{T_{ROT}}{2} + \frac{L_{MSG}}{V_{LINEA}} = 0{,}009s + \frac{0{,}008s}{2} + \frac{1024byte}{300 \cdot 10^6 \frac{byte}{s}} = 0{,}013s$$

Mentre il tempo di risposta e il coefficiente di utilizzazione saranno pari a:

$$T_{R\ HD} = \frac{T_{S\ HD}}{1+(\lambda_{HD}\cdot T_{S\ HD})} = \frac{0{,}013s}{1+(0{,}0021\cdot 0{,}013)} \approx 0{,}013s$$

e $$\rho_{HD} = \lambda_{HD}\cdot T_{S\ HD} = 0{,}0021\cdot 0{,}13 = 0{,}273\cdot 10^{-3}$$

Calcoliamo ora invece i dati relativi alla CPU, consideriamo un processore Intel® Core™ 2 Duo con frequenza di clock ad esempio di f_{ck}=1,83GHz e supponiamo esegua un'istruzione a ogni ciclo di clock, quindi:

$$V_{CPU} = \frac{1{,}83\cdot 10^{9}}{10^{6}} = 1830MIPS$$

Calcoliamo il tempo di servizio della CPU considerando che la richiesta sia composta da 150000 istruzioni (L_{IST}):

$$T_{S\ CPU} = \frac{L_{IST}}{V_{CPU}} = \frac{150\cdot 10^{3}\,istruzioni}{1{,}83\cdot 10^{9}\,\frac{istruzioni}{s}} = 81\mu s$$

E vediamo quanto vale il tempo di risposta della CPU:

$$T_{R\ CPU} = \frac{T_{S\ CPU}}{1+(\lambda_{CPU}\cdot T_{S\ CPU})} = \frac{81\mu s}{1+(0{,}0026\cdot 81\cdot 10^{-6})} \approx 80\mu s$$

e il coefficiente di utilizzazione della CPU:

$$\rho_{CPU} = \lambda_{CPU}\cdot T_{S\ CPU} = 0{,}0026\cdot 80\cdot 10^{-6} = 0{,}208\cdot 10^{-6}$$

Il teorema che ci permette adesso di assemblare tutti i valori appena calcolati per ricavare il tempo risposta totale, è il *teorema di Jackson* che dice:

Il tempo di risposta in una rete di servizi esponenziali con uno o più serventi si calcola:

1) *Risolvendo il sistema di equazioni che fornisce per ogni nodo il flusso totale entrante;*
2) *Calcolando il T_r di ciascun nodo con le formule M/M/n;*

3) *Infine sommano i T_r dei nodi attraversati dalla transazione;*
4) *Può essere calcolato come cifra di merito il T_r medio dato dalla somma di tutti i T_r diviso il numero di nodi.*

$$T_R = T_{R\ WAN} + T_{R\ CPU} + T_{R\ HD} = 0{,}002 + 0{,}208 \cdot 10^{-6} + 0{,}013 = 0{,}015s$$

$$T_{Rmedio} = \frac{T_{R\ WAN} + T_{R\ CPU} + T_{R\ HD}}{3} = \frac{0{,}002 + 0{,}208 \cdot 10^{-6} + 0{,}013}{3} = \frac{0{,}015}{3} = 0{,}005s$$

In conclusione, abbiamo visto come tutti i coefficienti di utilizzazione calcolati per i vari nodi risultano essere minori del valore critico $\rho_{critico}$=0,3/0,4. Inoltre essendo un sistema Markoviano possiamo calcolare il tempo di risposta nel 90% e nel 95% dei casi:

$$T_{90\%} = 2{,}31 \cdot T_R = 2{,}31 \cdot 0{,}015\text{sec} = 0{,}03s$$

$$T_{95\%} = 3 \cdot T_R = 3 \cdot 0{,}015\text{sec} = 0{,}05s$$

Entrambi i tempi sono sotto i 2-3 secondi, quindi il nostro sistema ha delle performance accettabili.

7 CONCLUSIONI E PROSPETTIVE

Facendo un riepilogo del periodo di tesi svolto all'INFN, una parte consistente del lavoro ha riguardato l'adattamento del codice php alla tipologia del file di log. Ogni qual volta si sono presentate delle modifiche nello strato sottostante, e quindi nel file di log proveniente da questo, spesso, è stata riscontrata la necessità di cambiare anche l'interfaccia con l'utente o, cosa più complessa, la rappresentazione grafica dei dati.

Un'altra importante considerazione fatta durante il lavoro, è stata quella di cercare di realizzare il miglior immagazzinamento dei dati possibile, cercando di evitare sprechi inutili di memoria di massa. Essendo molto elevata la mole di dati ricevuti giornalmente, sarebbe stato oneroso salvarla nella sua interezza, non essendo comunque tutta di rilevante importanza. In primo esame si era deciso di registrare sul DB tutti i dati provenienti dalle fibre, ma ad un certo punto si è compreso che la scelta migliore era registrare solo gli eventi più importanti e una media globale dei dati per evitare sprechi di spazio, ridondanza di informazioni e difficoltà nella rappresentazione grafica.

Il progetto così come è stato realizzato, prevede una parte del database predisposta a contenere i dati storici, ovvero tutti quegli elementi che, passata una certa data, vengono considerati da archiviare, in questo modo la parte coi dati attuali rimane sempre aggiornata, ma allo stesso tempo le informazioni meno recenti non vengono perse e rimangono disponibili per una consultazione futura da parte degli utenti dell'INFN. Ovviamente essendo un progetto in via di sperimentazione al momento dello

sviluppo dell'applicazione non si disponeva ancora di informazioni da archiviare, ma si è deciso comunque di settare questa parte, in modo tale che nel momento in cui si raggiungano tali condizioni, possano essere archiviate senza problemi. A livello di database un'altra importante considerazione da fare riguarda la parte più ampia della gerarchia, ovvero, noi ci siamo occupati del livello più basso dell'impianto totale, supponendo che vi fosse un'unica piattaforma su cui si stanno svolgendo i controlli. Chiaramente si tratta di una sperimentazione, nel momento in cui il progetto verrà realizzato in un deposito nucleare reale, le piattaforme potrebbero essere parte di un settore, e l'organizzazione potrebbe risultare più complessa (*Figura 29*), ma il DB è stato progettato in vista di depositi di grandi dimensioni, per cui tutto ciò non risulta essere un problema.

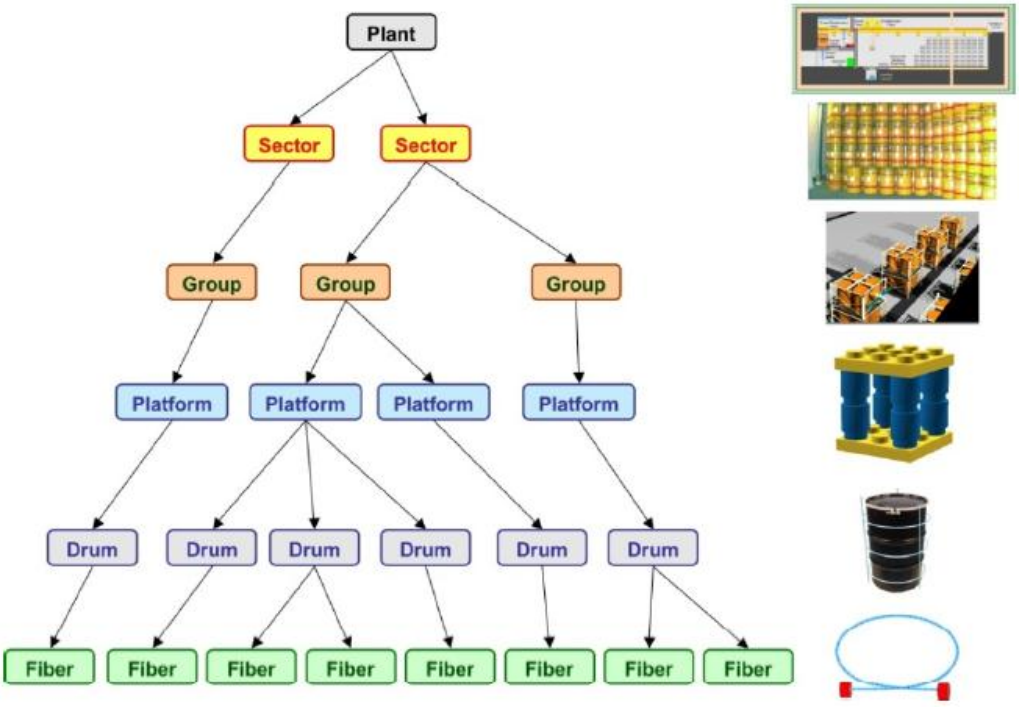


Figura 29 Gerarchia generale

Per quanto riguarda invece la parte di elaborazione dati e grafica, sono state create, come visto nei capitoli precedenti, delle pagine per la visualizzazione dell'andamento dei conteggi rilevati per tutte le fibre di un bidone oppure per una singola fibra, selezionata dopo aver scelto il bidone al quale appartiene. Un possibile miglioramento funzionale potrebbe essere apportato inserendo delle pagine di confronto tramite le quali si possa scegliere di andare a visualizzare nella stessa pagina o grafici di più fibre di uno stesso bidone o di bidoni differenti, in modo tale da fare dei confronti direttamente sul grafico, oppure ancora più bidoni contemporaneamente. Si sarebbe comunque trattato soltanto di andare a replicare la pagina relativa al grafico, e inserirla nello stesso form in cui attualmente è presente una sola rappresentazione, sia per il bidone che per la fibra singola.

Alla fine dell'esperienza il bilancio delle conoscenze che ho acquisito è senz'altro positivo. Innanzitutto è stato importante lavorare in gruppo, poiché spesso è stato utile il confronto per arrivare più velocemente ad una soluzione. Ma anche l'aver lavorato autonomamente è stato utile al fine di sviluppare conoscenze che spaziano dalla programmazione in un linguaggio, per me nuovo, come php, all'applicazione di molti concetti finora relegati ad un aspetto solo teorico che ho appreso durante gli studi universitari.

Per quanto riguarda l'applicazione e i suoi eventuali sviluppi futuri, si potrebbero come primo obiettivo aggiungere delle accortezze che rendano il sito-web ancora più performante, come ad esempio, per quanto riguarda i requisiti, potrebbe essere utile l'aggiunta della distinzione tra utente e amministratore. Ad oggi, non è stata richiesta questa distinzione perché avrebbe previsto una divisione dei ruoli in tutto ciò che

può essere fatto nell'applicazione, si è quindi deciso di creare un sito-web che venisse realizzato in vista di uso fatto da un utente amministratore, ma che prevedesse al suo interno una predisposizione ad essere modificato in tale senso.

Successivamente, si potrebbe pensare di progettare un sistema ad alta affidabilità per tutelare i dati presenti sul DB e sui dispositivi di archiviazione tramite qualche software che realizzi i cosiddetti "high availability (HA) clusters" (una delle possibili soluzioni potrebbe essere: DRDB accoppiato a Linux-HA). Quest'ultima soluzione è stata elaborata e in parte sperimentata, dalla collega tirocinante che ha lavorato in team con noi, Carmela Greco.

8 BIBLIOGRAFIA

[1] Paolo Finocchiaro, "DMNR: A new concept for real-time online monitoring of short and medium term radioactive waste", 1022 Nova Science Publishers, Inc.

[2] Documentazione php, http://www.php.net/docs.php

[3] Php Wikipedia, http://it.wikipedia.org/wiki/PHP

[4] Documentazione phpMyAdmin, http://localhost/phpmyadmin/Documentation.html

[5] Documentazione pChart, http://pchart.sourceforge.net/documentation.php

[6] MySQL 5.1 Reference Manual, http://dev.mysql.com/doc/refman/5.1/en/index.html

[7] Documentazione Apache HTTP SERVER Apache/2.2.14, http://httpd.apache.org/docs/2.2/

[8] Statistiche di Netcraft, http://news.netcraft.com/

[9] Mamp, http://it.wikipedia.org/wiki/MAMP http://www.mamp.info/en/index.html

[10] Javascript Wikipedia, http://it.wikipedia.org/wiki/JavaScript

[11] AJAX Wikipedia, http://it.wikipedia.org/wiki/AJAX

[12] Guida Ajax, http://javascript.html.it/guide/leggi/95/guida-ajax/

[13] Rachel Andrew, "Guida ai CSS: 101 trucchi, segreti e soluzioni", 2007, HOEPLI Informatica

[14] Modello di coda M/M/1, http://net.infocom.uniroma1.it/corsi/TMR/lucidi/4_MM1_TMR_2011.pdf

[15] Teoria delle code, http://www.fabriziomondo.com/blog/2011/03/25/il-modello-di-coda-mm1/

[16] EasyGoogleMaps, http://www.phpclasses.org/package/3801-PHP-Display-world-location-using-Google-maps.html

INDICE DELLE FIGURE

APPENDICE A

A.1 CODICE DEL FILE *readData.php*

Di seguito è presentato il codice che riguarda l'elaborazione dei dati presenti nel file di log e la loro memorizzazione nel DB.

```
<?php

include ("FiberClass.php");
include ("AlarmFiberClass.php");

function readData($link,$filename,$id_log){

$array_fibers=array();
$array_alarms=array();
$normal='normal';
$count_value=0;
$status='normal';
$found=FALSE;

//apertura file
$handlefile=fopen($filename,"r");
if($handlefile){

//salta la prima riga
$substring=fgets($handlefile,4096);

//lettura riga per riga
while($substring[0]!="\n"){
      $found=FALSE;
      $buffer = fgets($handlefile, 4096);
      $substring=explode(";", $buffer);
      $id_drum = $substring[0];
      if(isset($substring[2]))
            $d_time = $substring[2];
      if(isset($substring[3]))
            $id_fiber_r = $substring[3];
      if(isset($substring[4]))
            $instant_count= $substring[4];
      if(isset($substring[6]))
            $timestamp= $substring[6];
      if(isset($substring[9]))
            $activity= $substring[9];
      if(isset($substring[12]))
            $temperature= $substring[12];
      if(isset($substring[11]))
            $alarm_value= trim($substring[11]);
            $is_alarm=strcasecmp($alarm_value,$normal);
```

```
//controllo che la fibra esista sul db
$result= mysql_query("SELECT * from FIBER WHERE
id_fiber='$id_fiber_r' AND id_drum='$id_drum'",$link);
$control_df=mysql_fetch_array($result);

if($control_df[0]){

//inserisco il primo elemento se l'array è vuoto
$n=count($array_fibers);
             if($n==0){
                  $fiber=new Fiber($id_fiber_r,$timestamp);
                  $fiber->increment($instant_count);
                  $fiber->incrementDTime($d_time);
                        if($is_alarm){
                              $fiber->setStatus();
                              $alarm_fiber=new
AlarmFiber($id_fiber_r,$activity,$temperature,$timestamp,$instant_
count);

      array_push($array_alarms,$alarm_fiber);
                              }

                  array_push($array_fibers,$fiber);
                  }

           else
                  {
                  $num_fibers=count($array_fibers);

            for($i=0;$i<$num_fibers;$i++){
                  $is_there=strcasecmp($array_fibers[$i]-
>id_fiber,$id_fiber_r);
                  $BUFFERTIME1=explode(" ",$array_fibers[$i]-
>timestamp);
                  $DATE1=$BUFFERTIME1[0];

                  $BUFFERTIME2=explode(" ",$timestamp);
                  $DATE2=$BUFFERTIME2[0];
                  $same_date= strcasecmp($DATE1,$DATE2);



                        if($is_there==0){ //la fibra esiste
già  nell'array delle fibre
                              if($same_date==0){
                                    $found=TRUE;
                                    $ind=$i;
                              }
                        }

                  }

                  if($found){
                  $array_fibers[$ind]->increment($instant_count);
                  $array_fibers[$ind]->incrementDTime($d_time);
                        if($is_alarm){
```

```
                                    $array_fibers[$ind]-
>setStatus();
                                    $alarm_fiber=new
AlarmFiber($id_fiber_r,$activity,$temperature,$timestamp,$instant_
count);

      array_push($array_alarms,$alarm_fiber);
                        }
                  }

                  else{

                        $new_fiber=new
Fiber($id_fiber_r,$timestamp);
                        $new_fiber->increment($instant_count);
                        $new_fiber->incrementDTime($d_time);

                        if($is_alarm){
                              $new_fiber->setStatus();
                              $alarm_fiber=new
AlarmFiber($id_fiber_r,$activity,$temperature,$timestamp,$instant_
count);

      array_push($array_alarms,$alarm_fiber);
                              }

                        array_push($array_fibers,$new_fiber);
                        }
}

}

}


$num_fibers1=count($array_fibers);

echo "INSERIMENTO DI ".$num_fibers1 ." FIBRE";
      for($j=0;$j<$num_fibers1;$j++){
            $ID_FIBER=$array_fibers[$j]->id_fiber;
            $TIME=$array_fibers[$j]->timestamp;
            $BUFFER=explode (" ",$TIME);
            $DATE=$BUFFER[0];
            $count_value=$array_fibers[$j]->sum_count;
            $DELTA_TIME=$array_fibers[$j]->delta_time;
            $ID_EVENT_FIBER=$ID_FIBER."_".$DATE;
            mysql_query("insert into EVENT_FIBER_LOG
(id_event_fiber,id_log,count_temp,time_temp)values('$ID_EVENT_FIBE
R','$id_log','$count_value','$DELTA_TIME')");
            $TIME=$array_fibers[$j]->timestamp;


//controllo che quella fibra non esista già sul db per quel giorno
$control=mysql_query("SELECT * from EVENT_FIBER WHERE
id_event_fiber='$ID_EVENT_FIBER'");
$control_res=mysql_fetch_array($control);
            if($control_res[0]){
                  $old_count=$control_res[2];
                  $new_count=$old_count + $count_value;
```

```
                  mysql_query("update EVENT_FIBER SET
count='$new_count' WHERE id_event_fiber='$ID_EVENT_FIBER'",$link);
                  $old_time=$control_res[6];
                  $new_time=$old_time + $d_time;
                  mysql_query("update EVENT_FIBER SET
time='$new_time' WHERE id_event_fiber='$ID_EVENT_FIBER'",$link);
                  }
            else
            $insert=mysql_query("INSERT INTO
EVENT_FIBER(id_event_fiber,id_fiber,timestamp,count,time) values
('$ID_EVENT_FIBER','$ID_FIBER','$TIME','$count_value','$DELTA_TIME
')");
                  }

            if($array_alarms[0]){
            $num_alarms=count($array_alarms);
                  for($j=0;$j<$num_alarms;$j++){
                        $ALARM=$array_alarms[$j];
                        $id_fiber_al=$ALARM->id_fiber;
                        $alarm_date=$ALARM->timestamp;
                        $buffer_date=explode(" ",$alarm_date);
                        $data=$buffer_date[0];
                        $search=mysql_query("SELECT id_event_fiber
from EVENT_FIBER WHERE id_fiber='$id_fiber_al' AND
timestamp='$data'");
                        $row=mysql_fetch_array($search);
                        $ID_EVENT_FIBER=$row[0];
                        $ACTIVITY=$ALARM->activity;
                        $COUNT=$ALARM->count;
                        $TIMEST=$ALARM->timestamp;
                        $TEMPERATURE=$ALARM->temperature;
                        $insert_al=mysql_query("insert into
ALARM_FIBER(id_event_fiber,count,activity,timestamp,temperature)
values
('$ID_EVENT_FIBER','$COUNT','$ACTIVITY','$TIMEST','$TEMPERATURE')"
);
                  }
            }

}
if(!$insert)
      echo "Errori in event_fiber: ".mysql_error();
if(!$insert_al)
      echo "Errori in alarm_fiber: ".mysql_error();
}

?>
```

A.2 CODICE DEL FILE *readLog.php*

Il file *readLog.php* è quello che si occupa di prendere il file dalla directory indicata dall'utente, verifica che già non esista nel database e chiama la funzione presente nel precedente file *readData.php* per far sì che i valori vengano memorizzati dopo essere elaborati.

```
<?php

include ("openDB.php");
include ("saveFromXML.php");
include ("readComments.php");
include ("readData.php");
include ("replaceLog.php");
include ("index.html");


$link=openConnection();
$count=mysql_query("SELECT * FROM DRUM",$link);
$num_rows= mysql_num_rows($count);

//permette di salvare i nuovi bidoni e le nuove fibre quando
arriva un nuovo file XML

if($num_rows==0){
      saveFromXML($link,$_POST['xml']);
      echo '<script Language="javaScript">
                alert("File .xml properly put!");
                </SCRIPT> ';
}

$filename = $_POST['filename'];
$filename2= $_POST['filename2'];

$handlefile=fopen($filename,"r");

$handlefile2=fopen($filename2,"r");

if($handlefile){

$logExists= mysql_query("SELECT * FROM LOG where
file_name='$filename'",$link);
$logExists1=mysql_fetch_array($logExists);
if(!$logExists1[0]){
      $id_log=readComments($filename,$handlefile,$link);

      if($id_log){
            readData($link,$filename,$id_log);
            echo '<script Language="javaScript">
                alert("File properly put!");
                </SCRIPT> ';
            }
```

```
}
else  {
      echo '<script Language="javaScript">
                  alert("File already exists!");
                  </SCRIPT> ';

      }
}

if($handlefile2){
$success=replaceLog($filename2,$link);
if($success){echo '<script Language="javaScript">
                  alert("File replaced!");
                  </SCRIPT> ';}
else{echo '<script Language="javaScript">
                  alert("Error in replacing file!");
                  </SCRIPT> ';}



}

mysql_close($link);
fclose($handlefile);

?>
```

A.3 CODICE DEL FILE *readComments.php*

Quest'altro file, anch'esso richiamato dal precedente *readLog.php,* si occupa invece di leggere i commenti finali del file di log e di memorizzarli opportunamente.

```
<?php

function readComments($filename,$handlefile,$link){

if($handlefile){

//SALTA LA PRIMA PARTE DEL LOG
      $buffer = fgets($handlefile, 4096);

      while($buffer[0] != " ")
                  $buffer = fgets($handlefile, 4096);
      }

//LEGGE I COMMENTI
            for($i=0; $i<1; $i++)
                  $buffer = fgets($handlefile,4096);
```

```
            while(!feof($handlefile)){
                  $buffer1= fgets($handlefile,4096);
                  $substr=explode(";",$buffer1);
                  if($substr[0]!=NULL){
                              $s=count($substr);
                              //date time
                              $timestamp=  $substr[0];

                              //date
                              $substr2=explode(" ",$substr[0]);
                              $newdata = $substr2[0];

                              //creo l'id_log associato a quel
giorno
                              $start=' Log Started';
                              $stop=' Log Closed';
                        if($s<3){
                              $newstr =
str_ireplace("\n","",$substr[1]);

      $firstLine=strcasecmp($newstr,$start);
                              $endLine=strcasecmp($newstr,$stop);
                              if($firstLine==0){
                                          $timestamp1=$timestamp;
                                          $id_log=
'log_'.$timestamp1;
                                          mysql_query("insert into
LOG(id_log,file_name) VALUES ('$id_log','$filename')",$link);
                              }


                              if ($endLine==0)
                                          $timestamp2=$timestamp;

                              $description = $timestamp . " " .
$substr[1];
                              if($firstLine && $endLine)
                              mysql_query ("insert into
EVENT_LOG(description,id_log)
VALUES('$description','$id_log')",$link);
                              }


                        else{
                              $str= "";
                              for($n=2; $n<$s; $n++)
                                    $str= $str . $substr[$n];

                              $newstr=str_ireplace("\n","",$str);


      $firstLine=strcasecmp($newstr,$start);
                              $endLine=strcasecmp($newstr,$stop);
                              if($firstLine==0)
                                    $timestamp1=$timestamp;

                              if ($endLine==0)
                                    $timestamp2=$timestamp;
```

```
                                $description = $timestamp . " " .
$str;
                                if($firstLine && $endLine)
                                mysql_query ("insert into
EVENT_LOG(description,id_log)
VALUES('$description','$id_log')",$link);
                            }
                    }
            }
mysql_query("update LOG SET start='$timestamp1' WHERE
start=0",$link);
mysql_query("update LOG SET stop='$timestamp2' WHERE
stop=0",$link);
echo "ID LOG GENERATO ".$id_log. "\n";
return $id_log;
}
?>
```