ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA GESTIONALE

Dipartimento di Informatica

TESI DI LAUREA

in
Ingegneria Gestionale

PROGETTAZIONE E SVILUPPO DI UN'INTERFACCIA WEB PER LA GESTIONE DI UN IMPIANTO FOTOVOLTAICO DI MEDIE DIMENSIONI

CANDIDATO
Brinis Stefano

RELATORE:
Chiar.mo Prof. Grandi Fabio

Anno Accademico 2012/13

Sessione II

Indice

1. Introduzione

Con questa tesi vado a descrivere la creazione di un'interfaccia web per la gestione di un impianto fotovoltaico di medie dimensioni. Mi focalizzerò comunque di più sugli aspetti tecnici informatici della creazione del sito. Riguardo all'impianto fotovoltaico in se, per burocrazie e tecnologie, faccio invece solamente dei richiami, dove necessario.

L'obiettivo, infatti, è di dimostrare le potenzialità del web odierno utilizzando i vari linguaggi disponibili per gestire attività più o meno complesse tramite interfacce semplici, complete e che svolgono egregiamente quello che in passato era relegato esclusivamente ai software.

L'evoluzione del web, dei suoi linguaggi di programmazione, e delle connessioni di rete, ne ha modificato completamente l'utilizzo. Se inizialmente era stato ideato per la visualizzazione di semplici pagine fisse, dove poter inserire testi, immagini e collegamenti, ora permette la creazione di piattaforme che permettono di ottenere quei risultati che una volta erano possibili solo installando un applicativo specifico nel proprio pc. Basti pensare ai servizi multimediali per la visualizzazione di filmati o per l'ascolto della musica (Google Music, Grooveshark), alle applicazioni da ufficio (Google Documenti) e molti altri. Il web, che di fronte a questa evoluzione è stato rinominato in *web 2.0,* permette tramite la sola installazione del browser di usufruire di sempre più contenuti, trasformando il pc in un terminale e i software in strumenti più intelligenti e facilmente condivisibili.

L'impianto fotovoltaico preso in esame è di proprietà della mia famiglia e situato nel comune di Cona (VE). La potenza nominale è di 100 kwp. L'investimento iniziale non trascurabile ne ha reso necessaria una gestione che andasse oltre le semplici letture prese manualmente. Ci siamo quindi affidati ad una ditta che ha provveduto all'installazione di un PLC (controllore logico programmabile) che ha il compito dell'elaborazione e del salvataggio in locale di tutti i parametri utili ai fini dell'analisi dell'impianto. A questo è stato aggiunto un router 3G WIFI per permetterne la connessione da remoto. Questi due componenti hanno reso quindi superflua la lettura manuale permettendo di evitare errori o sviste e garantendo allo stesso tempo una precisione indiscutibile. Con tali condizioni si rende quindi possibile un preciso monitoraggio dell'impianto.

I dati possono essere ottenuti in due modalità

- Dato in tempo reale tramite un software venduto dalla ditta stessa
- Backup delle letture e dei parametri quotidiani tramite file csv (comma-separated values)

Purtroppo il software è stato creato per permettere solamente una lettura dei parametri senza strumenti di analisi quali grafici, report, indici di prestazione, interrogazioni su periodi impostati, etc.

Tali incombenze sono dunque delegate al proprietario dell'impianto.

Nonostante l'apparentemente semplice funzionamento di un impianto di produzione di energia elettrica da pannelli fotovoltaici, è importante sapere che questo non può essere abbandonato a se stesso, ma richiede un costante e puntuale monitoraggio per individuare e risolvere sul nascere eventuali problemi.

L'interfaccia web creata cerca quindi di compensare a tutte queste carenze, permettendo il salvataggio delle letture su di un database e la loro visualizzazione tramite grafici e report. Ho integrato un servizio meteo e sono stati predisposti invii di allarmi o notifiche tramite mail. Ho predisposto delle schede per il giorno, il mese e l'anno di produzione. Una home dove visualizzare i dati principali oltre che la possibilità di inserire la lettura reale del contatore (che chiamerò più avanti lettura di allineamento), decisiva per rafforzare la validità dei dati.

2. Tecnologie utilizzate

Andiamo ora a vedere le varie tecnologie, i vari linguaggi che ho utilizzato per la realizzazione del sito.

a. HTML

L'HTML, acronimo per *HyperText Markup Language*, è la parte static della pagina web, ovvero descrive l'output che viene visualizzato tramite il browser. E' un linguaggio di markup come l'XML che vedremo più avanti. Standard, arrivato ora alla sua versione 5, strutturato a tag o etichette dove ciascuna di queste e i suoi attributi portano informazioni riguardanti il formato del testo la posizione, le immagini e tanto altro. Tutte queste informazioni vengono lette dal browser e tradotte a video per permettere la visualizzazione della pagina all'utente finale.

Non è oggetto della tesi dilungarsi con la descrizione di questo o altri linguaggi ma andiamo comunque a vedere la caratteristiche principali di una pagina HTML.

All'inizio del documento si trova il DTD (Document Type Definition) che va ad indicare, tramite un link, la versione HTML utilizzata. Questa informazione serve al browser per capire con quali regole interpretare il documento.

Di seguito si ha la seguente struttura:

```
<html>
  <header>
    <title></title>
    <meta>
    …….
  </header>
  <body>
    <div></div>
    <a></a>
    <img>
    ……

  </body>
```

Fig .1

Dove il tag <*html*> serve per delimitare tutto il codice. Al suo interno si trovano due grandi blocchi. L'*header*, o sezione di intestazione, dove si collocano informazioni aggiuntive quali il titolo della pagina, l'autore, link a fogli di stile CSS (che vedremo meglio successivamente), script o dati utili ai motori di ricerca.

Il tag <*body*> invece, corpo della pagina, contiene la parte visualizzabile del documento, con immagini e testo, link e tabelle, il tutto formattato nei minimi particolari. Si può dunque definire per ciascun oggetto la sua posizione, il suo colore. Il font del testo e la sua dimensione. Il bordo di una tabella o di un'immagine. E così via.

Così descritto, il codice HTML, comodissimo e rappresentativo della nascita del web, risulta però un linguaggio statico, che dev'essere riscritto ogni volta che si voglia modificare il documento. Non include al suo interno una capacità di ragionamento per riuscire ad esprimere differenti output sotto determinate e verificate condizioni.

Tale compito è stato dunque relegato ad altri linguaggi, nati come naturale evoluzione del web, come ASP o PHP.

b. ASP

Per la realizzazione di questo sito ho utilizzato ASP, ma la scelta era quasi indifferente. Sono entrambi *script* lato *server* che permettono l'esecuzione di calcoli o algoritmi per la generazione di codice *html* da inviare al *browser*.

L'ASP, acronimo per *active server pages*, viene scritto insieme all'*html*, ma viene riconosciuto dal server come distinto tramite dei delimitatori di inizio <% e di fine %>.

Eccone un esempio.

```
<%
Response.write("Hello World")
%>
```

Fig .2

Il quale produce in stampa la scritta "Hello World". Oppure

```
<%
nome_utente = Request.form("nome")
If nome_utente = "Giulio" then
Response.write("Grazie Giulio per essere tornato a trovarci")
End if
%>
```

Fig .3

Dove viene mandato in stampa un saluto personalizzato solo nel caso in cui il visitatore (e quindi variabile nome_utente) sia Giulio. Questi due banali esempi ci fanno già intuire le potenzialità di questo linguaggio.
Anche per questo linguaggio non è possibile farne un'analisi completa ma, in questa sua introduzione, posso sicuramente elencarne alcune delle principali caratteristiche che andrò poi ad utilizzare nella creazione del sito:

- Variabili
- Array
- Funzioni e procedure
- Strutture condizionali ("if…then…else", "select case")
- Cicli ("for…next", "for each…next", "do…loop")
- Gli oggetti e i metodi di ASP (come per esempio l'oggetto "response" sopra che possiede vari metodi fra i quali "clear", "redirect", "write"…etc)

In particolare è bene sottolineare l'importanza delle funzioni (come anche delle procedure). Queste, infatti, consentono di raccogliere una serie di istruzioni tra le parole Function e End Function per poterle poi richiamare ogni qual volta ci servono.

La funzione legge una serie di variabili passate come argomento e restituisce un valore (caratteristica che la contraddistingue dalla procedura che invece non restituisce alcun valore).

Vedremo più avanti come per il calcolo delle produzioni in kwh o in € in un certo intervallo di giorni, o come per ottenere l'efficienza della produzione e per molti altri calcoli, sia di fondamentale importanza l'utilizzo delle funzioni.

Queste, infatti, raccolte in un unico file che viene richiamato nelle pagine dove viene utilizzato come fosse una nostra libreria personale, ci permettono di centralizzare e razionalizzare il codice. Con il vantaggio di dover mettere le mani, per eventuali modifiche, alle sole istruzioni presenti all'interno della singola funzione.

c. CSS

CSS, acronimo per *Cascading Style Sheets* o *Fogli di stile,* è un linguaggio che permette la formattazione della pagina web. La definizione quindi dello stile grafico e di visualizzazione.

Si può quindi definire la dimensione e il colore del testo, il posizionamento e le dimensioni delle immagini, le spaziature e i bordi dei vari contenitori che si vanno a definire.

Più in generale il CSS fornisce delle regole di formattazione che vengono applicate ai selettori seguendo la seguente struttura:

Fig .4

I selettori possono essere di diversi tipi:

- **Universale**, tramite il quale, utilizzando il carattere asterisco *, si selezionano tutti gli elementi di un documento. Questo permette di definire immediatamente e con pochissime righe delle modifiche sostanziali all'intera pagina. Per esempio si può definire il font utilizzato, la dimensione del carattere, etc.
- **Selettore di tipo**, come nell'esempio in fig. 4, costituito dal nome di uno specifico elemento HTML.
- **Id e classi**, che danno la possibilità di organizzarsi lo stile con la massima flessibilità mantenendo pur sempre, come per le funzioni in ASP, la razionalizzazione del codice e una pratica manutenibilità.

Soffermandoci proprio su quest'ultimo tipo di selettore, andiamo a farne un esempio per capirne meglio il suo utilizzo:

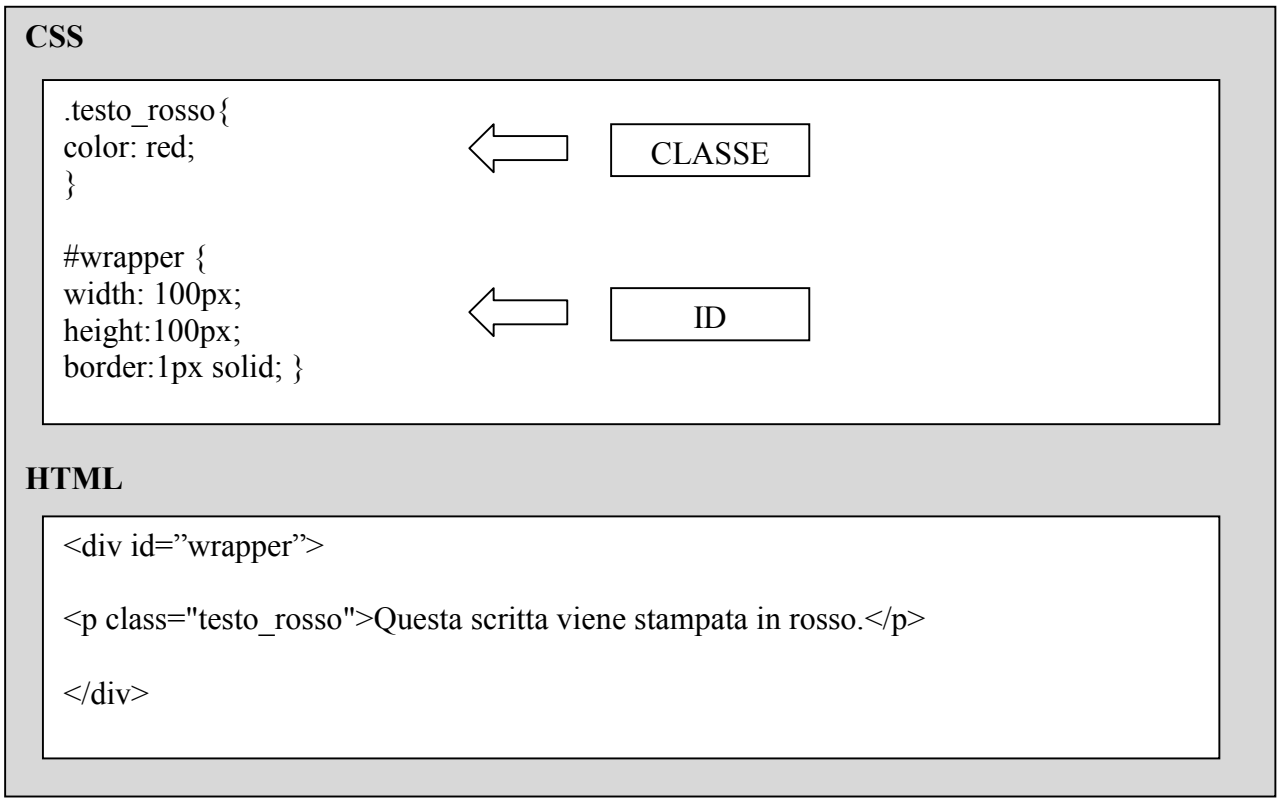


Fig .5

Dove nel CSS definisco una classe “.testo_rosso” e un id “#wrapper” che poi richiamo nei tag HTML per ereditarne le proprietà. Sia la classe che l’id rappresentano quindi un collegamento a delle righe di codice di formattazione presenti solitamente in un’altra pagina. Il CSS può essere infatti scritto anche esclusivamente nella singola pagina dove viene utilizzato o addirittura nel singolo tag. Queste ultime opzioni vengono scelte solo nei casi eccezionali di formattazioni *ad hoc.* La differenza principale tra classe e id stà nel fatto che in una pagina HTML l’id viene utilizzato per identificare univocamente un elemento. Quando si vuole applicare lo stesso stile a più elementi della stessa pagina si usa quindi la classe.

Concettualmente, questa organizzazione del codice, è simile all’utilizzo delle funzioni visto con ASP. Sono due tecniche per portano il programmatore ad organizzarsi meglio la scrittura del codice e soprattutto ad investire più tempo nella fase iniziale di ideazione del sito.

d. Database (DB)

Il *database* (o base di dati) è un insieme di archivi le cui informazioni sono strutturate secondo un modello che ne consenta una organizzazione efficiente e una consultazione tramite un linguaggio standard (come il linguaggio SQL che vedremo dopo).

Il modello oggi più utilizzato è quello relazionale, nonostante ne esistano di altri tipi (gerarchico, reticolare, semantico, ad oggetti) alcuni in disuso o superati mentre altri che rappresentano delle soluzioni innovative destinate a prendere piede col tempo.

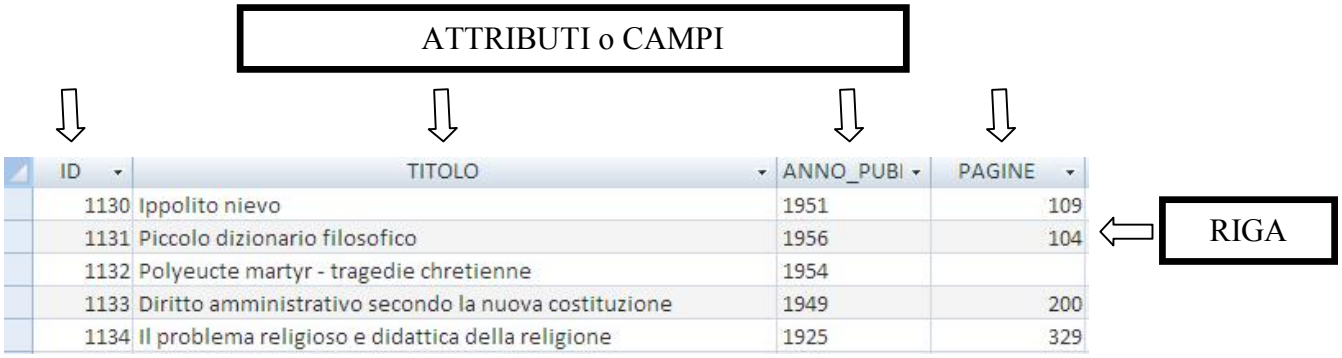
Tornando al modello relazionale (proposto negli anni '70), questo si basa sul concetto matematico di relazione (tabella) e sull'algebra relazionale. Gli operatori di quest'ultima servono infatti per manipolare i dati, rappresentati come relazioni.

E' possibile dunque definire

- tabelle
- attributi (ovvero i campi da assegnare alla tabella)

Poi l'inserimento dei valori per ciascun attributo genera la riga (o record o tupla o istanza) che rappresenta i dati veri e propri da memorizzare. Ecco dunque che la tabella è una relazione, ovvero un insieme di colonne e di righe.

Eccone un esempio



ID	TITOLO	ANNO_PUBI	PAGINE
1130	Ippolito nievo	1951	109
1131	Piccolo dizionario filosofico	1956	104
1132	Polyeucte martyr - tragedie chretienne	1954	
1133	Diritto amministrativo secondo la nuova costituzione	1949	200
1134	Il problema religioso e didattica della religione	1925	329

Fig .6

Per ciascun campo può essere definito, in particolare, il tipo di dato (numero, testo, data) e dei vincoli (intervallo numerico o di date, solo alcuni valori, etc).

Come si può vedere nella tabella d'esempio in Fig. 6, esiste un campo che a differenza degli altri sembra non avere un contenuto informativo importante: il campo "id". Nonostante sia spesso un campo di questo tipo (a contatore) svolge una funzione di importanza fondamentale. E' quel dato (o anche insieme di dati) che identifica univocamente un record, e prende il nome di chiave primaria.

A volte coincide direttamente con un campo già previsto nella tabella come il codice fiscale per una rubrica o l'ISBN per i libri di una biblioteca (come poteva essere per l'esempio in tabella).
L'importante è che mantenga le sue due caratteristiche fondamentali:

- non può essere nullo
- non può essere ripetuto

Vediamo ora un'altra caratteristica importante di questo modello: la relazione fra tabelle. Ovvero il segnalare il fatto che un record di una tabella contenga dati che interessano un record di un'altra tabella. Queste relazioni possono essere di tre tipi:

1. uno a uno (per esempio un immobile e il contratto di locazione che lo interessa)
2. uno a molti (un immobile e gli inquilini che lo abitano)
3. molti a molti (un intestatario che può avere uno o più immobili e un immobile che può avere uno o più intestatari)

Il database rappresenta una parte fondamentale per la creazione di un sito web come di una qualsiasi applicazione che ne faccia uso. E' quindi importante dedicarci il giusto tempo per la sua progettazione. A tal riguardo, nel passaggio dallo schema concettuale (ovvero la descrizione a parole di ciò che si vuole ottenere) e quello relazione del database, si utilizza lo schema ER (entità relazione). Il database che si andrà quindi a creare dovrà rappresentare entità diverse per ciascuna tabella e non dovrà avere inutili duplicazioni di dati (normalizzazione del database).

e. SQL

SQL, acronimo per *Structured Query Language,* è un linguaggio standard per la gestione di *Database* relazionali. Per gestione si intendono tutte quelle attività inerenti il *database*, dall'inserimento-modifica-eliminazione dei dati, alla creazione del database o delle tabelle.
E' un linguaggio dichiarativo, dove tramite una serie di istruzioni si definisce lo scopo da raggiungere lasciando al linguaggio stesso l'esecuzione dei vari algoritmi.
Come accennato sopra, si divide in:

- DDL (data definition language) per creare o cancellare il database e definirne la struttura
- DML (data manipulation language) per l'inserimento, la cancellazione e la modifica del dati
- DCL (data control language) per la gestione degli utenti e dei permessi
- QL (query language) per l'interrogazione del database

Per la realizzazione dell'interfaccia web oggetto di questa tesi ho utilizzato solamente l'ultimo, avendo creato autonomamente (tramite Access) il database e le tabelle.
Vediamo quindi qualche esempio relativo al *query language*:

- **SELECT cognome, nome FROM rubrica**
- **SELECT TOP 10 * FROM inventario WHERE id_marchio = '61' ORDER BY nome ASC**
- **SELECT COUNT(*) AS uscite_totali FROM contabilità WHERE tipo = 'dare'**

Fig . 7

Nel primo caso si selezionano per tutti gli utenti le sole colonne cognome e nome dalla rubrica. Nel secondo si trovano i primi 10 risultati per le sole righe della tabella inventario dove il campo id_marchio sia uguale a 61, ordinate per la colonna 'nome' (dove ASC individua l'ordinamento dalla a alla z).
Nell'ultimo esempio invece si ha come risultato un solo valore (una tabella 1x1) con un campo uscite_totali che avrà come valore il conteggio di tutte le righe della tabella contabilità in uscita (tipo = 'dare').
Questi sono comunque esempi molto banali che non rappresentano a dovere le potenzialità di questo linguaggio. Soprattutto per database complessi (o comunque per ricerche apparentemente complicate) risulta estremamente efficace grazie alla sua estrema flessibilità.

f. AJAX

Ajax, acronimo di Asynchronous JavaScript and XML, si intende un concetto che utilizza tecnologie già esistenti, Javascript e XML appunto, per mirare ad un obbiettivo specifico: lo scambio di dati in background tra client e server senza che vi sia l'aggiornamento della pagina Web. Se per pagina dinamica si intende una pagina con la quale è possibile interagire e che risulta sempre diversa a seconda della richiesta che si invia, con questa nuova tecnologia il dinamismo risulta molto più accelerato. Sembra che la pagina stessa caricata sul client si trascini dietro un potenziale di calcolo e di dati tale da consentire una moltitudine di operazioni senza richiesta al server. Ma così non è.

Quando si parla di Ajax si intende soprattutto un oggetto in particolare: l'XMLHttpRequest. Ormai incluso in tutti i Web browser, è ciò che permette l'invio di dati in background al server, e quindi una richiesta, in modo del tutto indipendente dal browser del client.

Ciò comporta una rivoluzione nella comunicazione tra client e server. La comunicazione tipica è infatti di questo tipo:

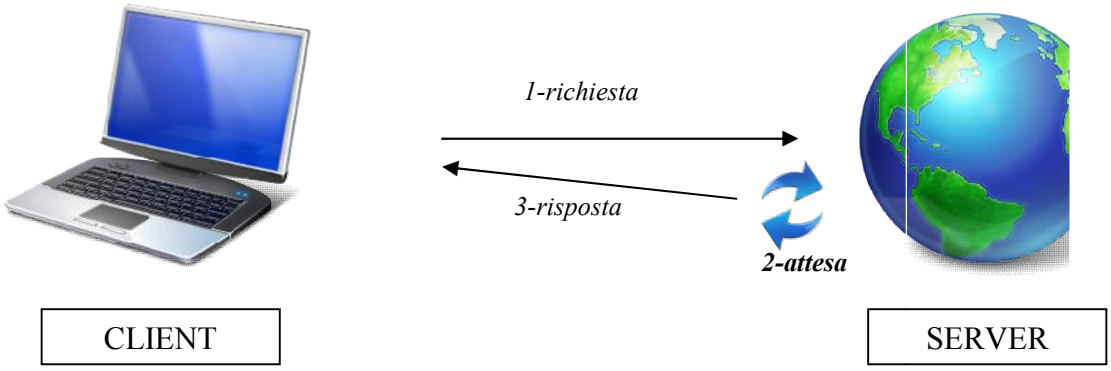


Fig . 8

E quindi si avevano 3 step forzati per la comunicazione. La richiesta, il tempo di attesa e infine la risposta con il caricamento della nuova pagina nel browser.

Con Ajax la comunicazione rispetta ancora questa gerarchia ma svincolandosi dalla pagina Web che genera la richiesta. Questa infatti non viene ricaricata ma di volta in volta viene ripopolata di nuovi contenuti a seconda dell'evento scatenato, sia da parte dell'utente sia previsto dalla pagina stessa. La configurazione è la seguente

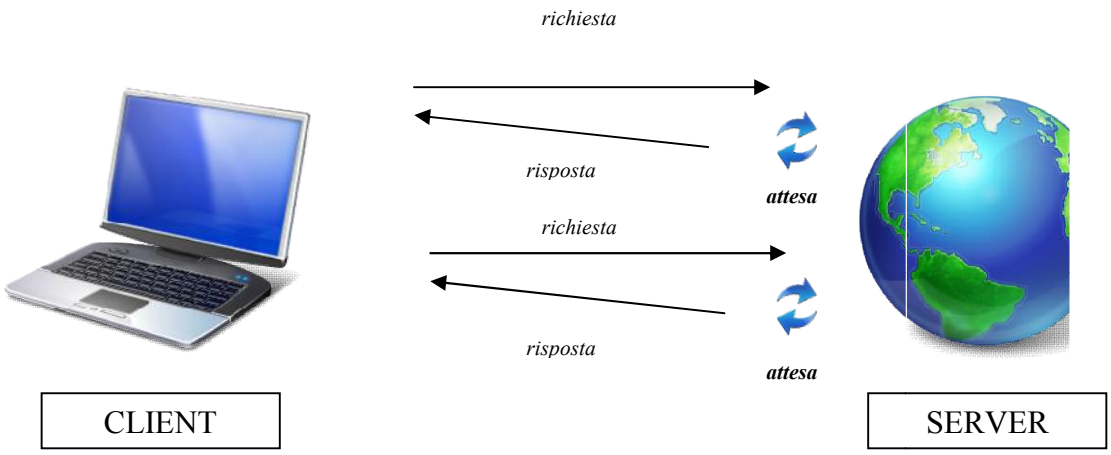


Fig . 9

Si possono quindi avere più richieste simultanee e tutte indipendenti tra loro. Ciò comporta da una parte una maggior velocità di esecuzione poiché questi tempi di attesa sono spesso impercettibili, d'altra parte l'utente, se il tempo di attesa pur piccolo si protrae troppo, può essere talvolta tentato di chiudere la pagina Web ignaro che in realtà ci sia una richiesta di informazioni in atto.

g. XML

XML, acronimo per eXstensible Markup Language, è un linguaggio di markup, che consente di descrivere dei dati tramite dei marcatori, i cosiddetti tag.
Inizialmente, quando è stato introdotto, il suo obiettivo era di migliorare lo scambio di informazioni nel Web e consentire una facile ed efficace pubblicazione di dati arbitrariamente complessi. Per le sue caratteristiche si presta a moltissime applicazioni per lo scambio di dati o informazioni tra software o applicativi eterogenei.
La formattazione di un documento XML deve prevedere:

- Un prologo, dove si dichiara la versione di xml alla quale ci si riferisce ed eventualmente le regole di conformità del documento
- L'elemento radice, ovvero il nodo principale
- Elementi interni al nodo radice

Il contenuto informativo viene passato all'interno di un tag oppure direttamente tramite attributo come si può notare nella figura qui sotto:

```
<studenti>
   <studente>
      <matricola>0020120102</ matricola >
   </ studente >
</studenti>

OPPURE

<studenti>
   <studente  matricola=' 0020120102'/>
</studenti>
```

INTERNO AL TAG

COME ATTRIBUTO

Ma vediamo ora un esempio completo di come potrebbe apparire un documento XML completo:

```
<?xml version="1.0" encoding="UTF-8"?>
<veicoli>
   <veicolo>
      <marca>Fiat</ marca >
      <modello>500 L</ modello >
      <targa>GD121314</ targa >
   </ veicolo >
   <veicolo>
      <marca>Fiat</ marca >
      <modello>Punto</ modello >
      <targa>FT908777</ targa >
   </ veicolo >
</veicoli>
```

Come si può notare, un documento XML così costituito altro non è che un archivio di dati simile alla tabella di un database. La tabella in questo caso sarebbe quella dei veicoli, con 3 colonne o campi (marca, modello, targa) e 2 record inseriti.
In sintesi, è un linguaggio che permette strutture ad-hoc per l'invio di dati con il vantaggio di essere facile da scrivere e da leggere. A causa però della sua struttura testuale e dei tag, il documento XML risulta essere più pesante dei documenti in formato binario, con la conseguenza di una trasmissione non ottimale.

h. API

API, acronimo di Application Programming Interface, è un'interfaccia che estende le funzionalità di un software o di un'applicazione Web. E' un insieme di funzioni, metodi e proprietà, che i programmi possono richiamare al fine di delegare il lavoro al sistema sottostante. Per spiegare meglio questo concetto faccio un breve esempio.
Le caselle di posta elettronica possono essere visualizzate tramite il sito nel quale risiedono ma anche e soprattutto tramite altri programmi come Outlook. L'accesso alla casella viene fatto via POP3 o IMAP, protocolli di comunicazione che consentono l'accesso sicuro alla propria email, con due enormi vantaggi.

1. L'utente può decidere di visualizzare i messaggi ricevuti tramite il software che preferisce, con la formattazione che desidera. L'accesso può essere fatto multiutente e tramite qualunque dispositivo che consenta l'utilizzo di programmi con protocollo POP3 o IMAP.
2. Si aumenta la fidelizzazione dell'utente al proprio sito o servizio

Il software o l'applicazione utilizzati diventano più flessibili e manipolabili nei modi che si preferisce.

Youtube permette di integrare i suoi video e le sue funzionalità nel proprio sito web, software o dispositivo con un'altissima personalizzazione. Faccio ora una prova per poter visualizzare un video di Youtube in un mio spazio Web.

Fig . 10

A fianco a ciascun video di Youtube troviamo infatti una schermata come questa. La parte interessante è il "Codice da incorporare" che può essere personalizzato agendo sulle impostazioni appena sotto. Si può infatti mostrare o meno il bordo, includere i video correlati per intrattenere l'utente nel nostro sito alla fine della visualizzazione del primo video e impostare alcuni ritocchi grafici.

Andando infine a copiare quel codice è immediato, integrandolo a quello della nostra pagina, ottenere l'effetto desiderato.

In figura 11 si può vedere la tipica interfaccia di Youtube che, per rendere ancora più evidente la possibilità di integrazione, ho inserito in una piccola tabella. E così, senza dover acquistare o creare un servizio per visualizzare qualsivoglia filmato nel mio spazio Web, posso utilizzare gratuitamente i servizi resi già disponibili da un leader in questo campo quale Youtube. Ovviamente però, sull'interfaccia video ci sarà il suo logo così come ci saranno le varie pubblicità associate al video.

Fig . 11

Ma vediamo ancora un altro esempio per meglio cogliere le potenzialità di questa tecnica. E' possibile infatti far visualizzare agli utenti una mappa con i propri punti vendita per mostrare in maniera semplice e intuitiva quelli disponibili nella zona e far si che possa velocemente scegliere il più idoneo.

Anche Google Maps offre questo contributo con un'elevatissima possibilità di personalizzazione. E' possibile inserire le proprie icone e descrizione da visualizzare nei vari punti della mappa.

Un sito di compravendita immobiliare potrà quindi far visualizzare la mappa del singolo immobile o di più immobili nella zona sfogliando gli indirizzi dal proprio database e inviandoli come parametri all'API di Google Maps ottenendo il risultato cercato.

3. Descrizione del problema

a. *L'impianto fotovoltaico*

L'impianto fotovoltaico è un impianto elettrico atto alla conversione di energia solare in energia elettrica tramite l'effetto fotovoltaico. Quest'ultimo è un fenomeno fisico elettronico che consiste nell'innalzamento dei livelli energetici degli elettroni di materiali semiconduttori a seguito di radiazione incidente sulla superficie di tali materiali.

Fig. 12

L'unità energetica o elemento base di tali impianti è la cella fotovoltaica: costituita da una lamina di silicio, è di fatto un generatore di intensità di corrente. In essa gli elettroni vengono canalizzati in un sistema di connessioni in serie e parallelo ad altre celle fotovoltaiche, costituendo cosi il modulo fotovoltaico. Questi moduli fotovoltaici sono generatori di corrente continua di potenza nominale prestabilita in sede progettuale. L'entità di potenza nominale di un impianto fotovoltaico e quindi la sua energia prodotta dipendono in primis dal numero di pannelli installati. Essi opportunamente posati, orientati e cablati tra di loro prima in serie e poi in parallelo portano a valle del sistema di conversione dell'energia un dipolo corrente di potenza istantanea il cui valore di dipende dalla potenza nominale dell'impianto (numero di pannelli), irraggiamento istantaneo, e rendimento dei moduli fotovoltaici. Questa corrente continua viene così convertita in corrente alternata idonea all'immissione nella rete elettrica nazionale tramite il dispositivo inverter (figura 13).

Fig. 13

A parità di potenza nominale, la quantità di energia prodotta dipende dall'irraggiamento annuale dell'impianto, e quindi dalla latitudine in primis e dalle condizioni meteorologiche della zona di ubicazione dell'impianto.

Fattori importanti che mantengono il rendimento degli impianti sono la temperatura dei pannelli (basse temperature favoriscono il passaggio di corrente elettrica) la pulizia delle superfici dei moduli fotovoltaici e la costante manutenzione degli impianti (controlli elettrici, controllo ombreggiamenti, controllo rendimenti dei dispositivi inverter, quadri elettrici, trasformatori).

Si rende quindi, al fine di mantenere altro il rendimento e di monitorare la produzione dell'impianto, l'installazione di opportuni sistemi di monitoraggio che permettano di collezionare ed elaborare i dati, di energia, potenza, irraggiamento solare, temperatura, rendimenti parziali e totali.

La necessità di dovere innalzare i livelli di energia fotovoltaica prodotta a livello nazionale al fine di favorire lo sviluppo di energie rinnovabili a sfavore di quelle derivanti dalla combustione di fonti fossili ha spinto i governi dei vari paesi avanzati a trovare svariate forme di incentivazione alla costruzione di impianti elettrici a energia solare.

In Italia si è scelto di dare ai produttori un contributo monetario per unità energetica prodotta per 20 anni continuativi in modo che essi avessero un break-even point compreso tra i 6 e i 12 anni a seconda del tipo di incentivo (invece di più di 40 anni senza incentivazione governativa).

Tale provvedimento legislativo (denominato Conto Energia) ha determinato in breve una repentina ascesa degli investimenti privati e non (con capitali nazionali e esteri) nel settore. Dal 2005 fino all'anno in corso la costruzione di impianti fotovoltaici piccoli (fino a 20 Kwp), medi (fino a 200Kwp) e grandi (oltre i 200 kw) ha subito una aumento notevole.

Attualmente in Italia si è andati oltre i 15 Gwp prodotti con più del 5% del fabbisogno energetico nazionale soddisfatto tramite energia fotovoltaica.

Il conto energia come detto garantisce una quota incentivo per 20 anni variabile a seconda del tipo di impianto e del conto energia al quale l'impianto installato è iscritto. L'impianto in oggetto di 100kwp è iscritto al Secondo Conto Energia e riceve dal Gestore Servizi Elettrici (GSE – Ministero delle Finanze) un incentivo pari a 0,346€ per ogni Kwp prodotto (quota incentivo).

L'energia prodotta viene inoltre venduta a un prezzo che varia col prezzo di mercato e che attualmente si aggira sullo 0,08 € per ogni Kwp venduto (quota vendita).

Il produttore incassa quindi mensilmente un introito monetario variabile a seconda dell'energia prodotta.

Una stima dell'energia prodotta annuale tenendo conto della latitudine fa prevedere un ritorno al pareggio dell'investimento in 6,5 anni per l'impianto in oggetto. In assenza di quota incentivo e quindi con la sola vendita dell'energia il ritorno teorico si avrebbe oltre i 40 anni, un tempo che non permetterebbe di considerare tale investimento come profittevole.

4. Costruzione del sito

a. *Schema ER e database*

Come primo passo ci occuperemo di creare il database che dovrà permettere l'archiviazione dei dati riguardanti l'impianto fotovoltaico, nonché la loro modifica ed eliminazione.
In realtà, in questo caso, la strutturazione del database è molto semplice. Facendo un riassunto del capitolo precedente possiamo sintetizzare il problema in questo modo:

> *"La produzione dell'impianto fotovoltaico è costantemente controllata da un PLC che ne salva i valori tutte le sere alle 9. Da questi si ottiene in particolare la produzione effettiva dell'impianto. Questa deve essere confrontata con la produzione prevista che viene fornita tramite un valore mensile. Per consentire un'indagine più accurata riguardo l'efficienza dell'impianto è bene avere memoria delle condizioni metereologiche del giorno. Si vogliono inoltre salvare i pagamenti mensili da parte di GSE e ENEL per avere un immediato confronto tra ciò che si è prodotto in € e ciò che ci è stato pagato.*
> *Infine, per consolidare i dati, è bene prevedere l'allineamento delle letture a quella effettiva del contatore tramite un inserimento manuale periodico di quest'ultima."*

A questo punto si può passare alla progettazione del database. Una metodologia utilizzata è quella di schematizzare il problema tramite un modello ER (entity-relationship) per farne una rappresentazione concettuale. E' una prima "traduzione" del problema per renderlo oggettivamente comprensibile, per eliminare quindi eventuali ambiguità del linguaggio.
I principali costrutti del modello E-R sono le entità, le associazioni e gli attributi.
Le entità sono classi di oggetti (fatti, cose, persone, ...) che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse. Esempi di entità:

> *"La lettura giornaliera viene salvata automaticamente dal PLC. Per confrontare la produzione effettiva con quella prevista è necessario utilizzare un valore mensile che viene fornito dalla regione o enti specifici."*

In questo caso si evidenziano le due entità: letture giornaliere e previsto mensile.
Le associazioni (dette anche *relazioni*) rappresentano un legame tra due o più entità. Nel problema qui trattato non ci sono esempi di associazioni. O comunque le eventuali associazioni che si possono trovare sono trascurabili perché vengono "assorbite" come campi aggiuntivi nelle varie

tabelle-entità. Per capire meglio l'associazione facciamo un esempio di un eventuale sviluppo successivo del programma:

> *"Solo le persone autorizzate possono inserire le letture di allineamento."*

Oppure

> *"Progettare l'interfaccia per renderla disponibile a gestire non solo un impianto."*

Qui si identifica l'entità persone che risulta in associazione con letture di allineamento. L'associazione è l'inserimento della lettura. Questa associazione può avere degli attributi aggiuntivi come la data e l'ora.
Gli attributi sono fatti ciò che descrive le caratteristiche delle istanze di entità e le caratteristiche delle istanze di associazione. Un esempio può essere il seguente:

> *"Per ciascuna lettura giornaliera si vuole tener traccia della temperatura massima giornaliera pannello e delle potenze effettive e teoriche alle ore 10:00, 12:00 e 15:00."*

Questa dicitura ci evidenzia come l'entità lettura abbia come attributi (o proprietà): temperatura massima giornaliera, potenza effettiva ore 10:00, potenza effettiva ore 12:00, potenza effettiva ore 15:00, potenza teorica ore 10:00, potenza teorica ore 12:00 e potenza teorica ore 15:00.
Vado ora a creare il modello ER per il problema sopra esposto.

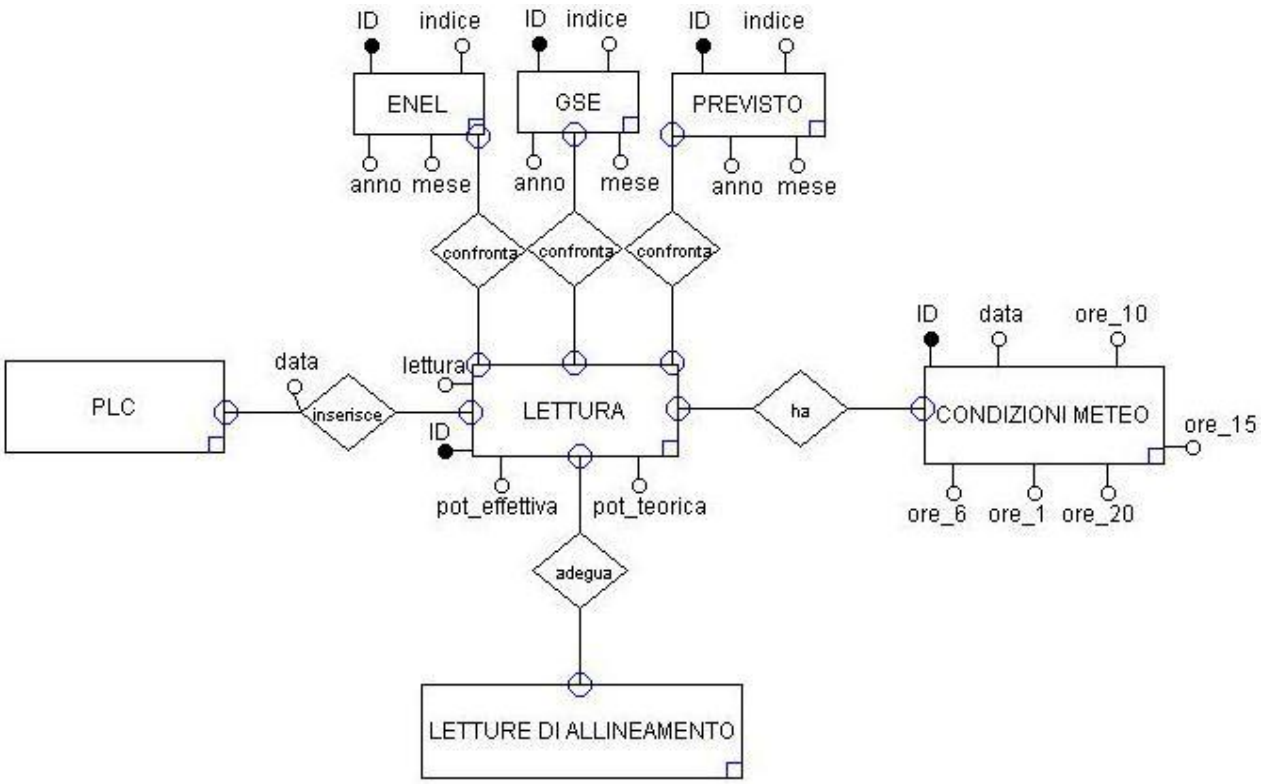


Fig . 14

Questa rappresentazione evidenzia le entità (quadrati) e le associazioni (rombi). Da questa si deve ottenere il database con le varie tabelle per l'archiviazione dei dati. In realtà nel database è importante dunque tradurre il modello nelle sole entità/associazioni utili ai fini del problema. Iniziando dall'entità PLC si nota subito come possa essere trascurata poiché c'è solo un impianto da trattare.

Nell'associazione tra PLC e LETTURA si possono integrare gli attributi (in questo caso solo *data*) direttamente nell'entità rimanente: LETTURA.

Per quanto riguarda invece le tre entità GSE, ENEL e PREVISTO si possono riunire in una tabella unica da chiamare *indici* dove si aggiunge un campo *tipo_indice* per distinguerli. I confronti non vengono salvati sul database ma generati ogni volta che se ne fa una interrogazione. Quindi le tre associazioni vengono eliminate ai fini della strutturazione del database e sostituite da una o più funzioni da scrivere in *asp*.

L'entità LETTURE DI ALLINEAMENTO deve essere conservata in una tabella a parte mentre la sua associazione rappresenta un aggiornamento che viene fatto non appena si inserisce la singola lettura di allineamento. Anche per questa parte andremo più avanti a vedere in dettaglio come gestirla.

In ultimo le CONDIZIONI METEO rappresentano un'altra entità non trascurabile. Andrà creata una tabella e l'associazione che la lega alle letture viene eliminata poiché tramite il campo data presente in LETTURE e il campo data in CONDIZIONI METEO possono "legare" direttamente le singole letture alle condizioni meteo corrispondenti.
In conclusione si ottiene un database così composto da queste tre tabelle e senza relazioni:

- LETTURE
- LETTURE ALLINEAMENTO
- METEO
- INDICI

Un'evoluzione del modello relazionale esposto fin qui (e quindi un'evoluzione dell'interfaccia gestionale) potrebbe essere questa:

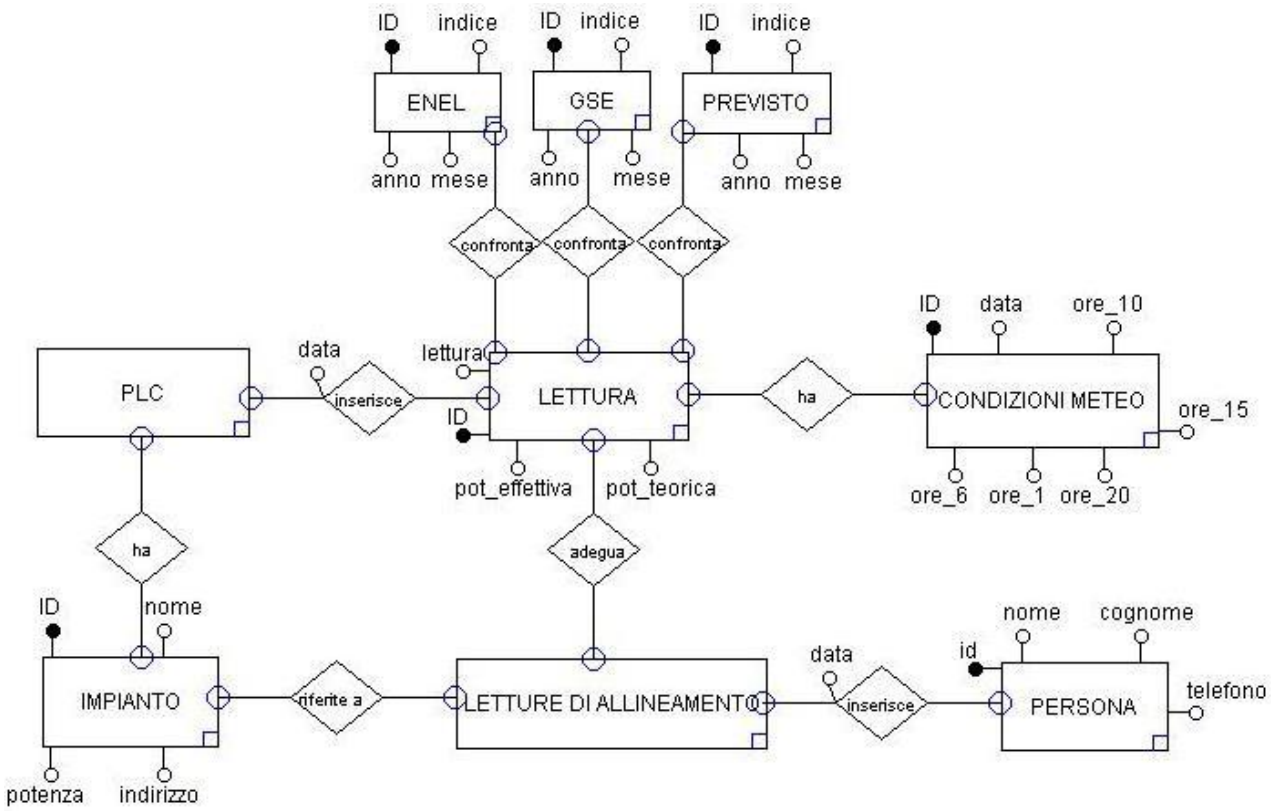


Fig . 15

In questo caso si vanno ad ottenere due entità aggiuntive

- IMPIANTI
- UTENTI (persone)

La prima che integra al suo interno anche gli eventuali dati del PLC (poiché sono in relazione 1 a 1). Mentre l'associazione uno a molti che lega il PLC con le letture diventerà una relazione. Per ciascun impianto possono essere inserite 1 o più letture. Nella tabella LETTURE andrò semplicemente ad aggiungere un campo *id_impianto*.
Stessa cosa per l'entità persone. Qui posso registrarmi i dati di chi è autorizzato ad inserire le letture di allineamento (e prevedere di conseguenza un servizio di login al sito). Nella tabella LETTURE DI ALLINEAMENTO andrò ad aggiungere il campo *id_persona*.
Tornando ora al primo modello ER possiamo definire le tabelle nel database come si può vedere nell'immagine qui sotto.

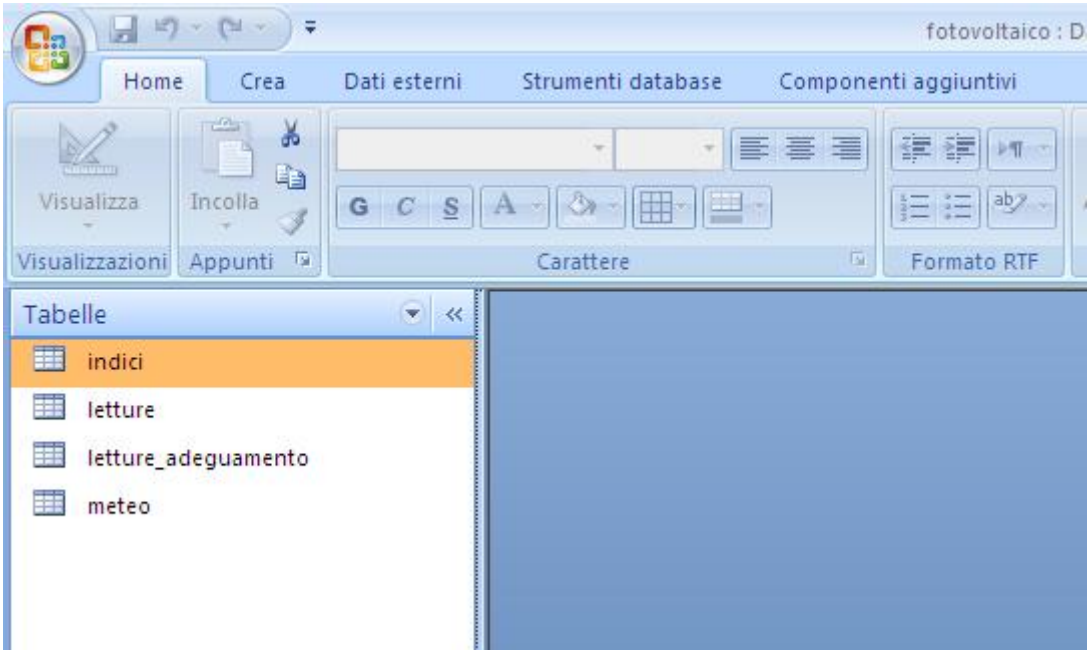


Fig . 16

Per la creazione del database ho utilizzato il programma Microsoft Access. L'elenco delle tabelle è dunque quello ottenuto come da analisi del modello ER. Per ciascuna tabella sono stati definiti i campi o attributi. Tra questi è essenziale averne uno che svolga il ruolo di chiave primaria. Questa è infatti formata da uno o più campi e permette di individuare univocamente un record o tupla all'interno della tabella. Se per esempio la tabella si riferisse ad un elenco di persone ecco che un campo *codice_fiscale* sarebbe perfetto per svolgere questo ruolo appunto per la sua caratteristica di univocità. Solitamente, e per praticità, si sceglie di aggiungere comunque un campo ID (identificatore) di tipo numerico incrementale o casuale. Per ogni nuovo record sarà compito del database stesso andare a generare un nuovo numero da associare.

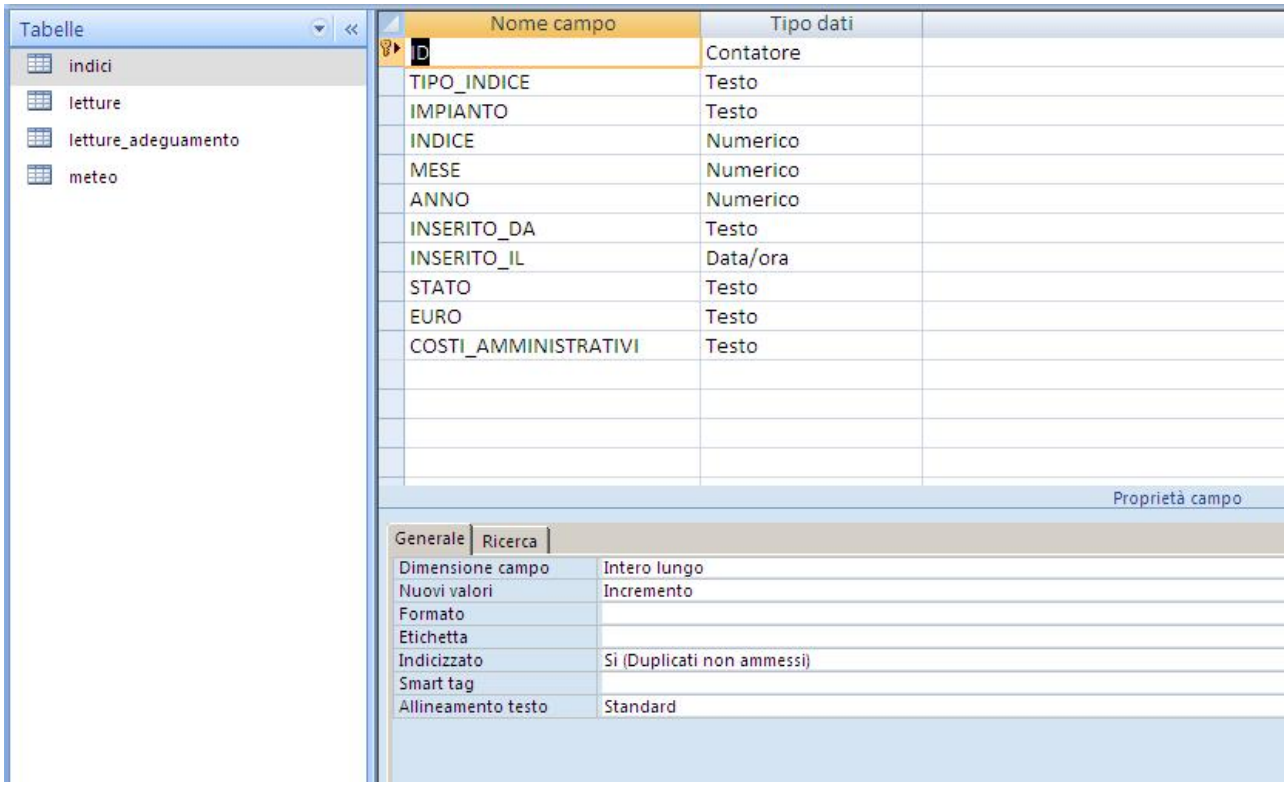


Fig . 17

In figura 17 ho preso ad esempio una delle tabelle del database (indici). Come si può notare, il primo campo in alto presenta un'icona a forma di chiave. E' la chiave primaria, ed è di tipo contatore incrementale. Tutti gli altri campi necessari devono essere impostati con una dicitura e un tipo. Quest'ultima caratteristica ci è utile sia per avere un controllo in fase di inserimento (campi numerici, data) che per poter agevolmente sfruttare le varie funzioni ad hoc native nel linguaggio SQL per effettuare le ricerche.

b. <u>Struttura della pagina e del menu</u>

La struttura della pagina è stata creata utilizzando i CSS. Come già anticipato in precedenza questo linguaggio ha l'importante caratteristica di rendere il sito facilmente modificabile per future eventuali esigenze.

Ho diviso quindi la pagina in diversi blocchi:

- Body (elemento radice della pagina html)
- Wrapper
- Pagina
- Header
- Avvisi
- Contenuto

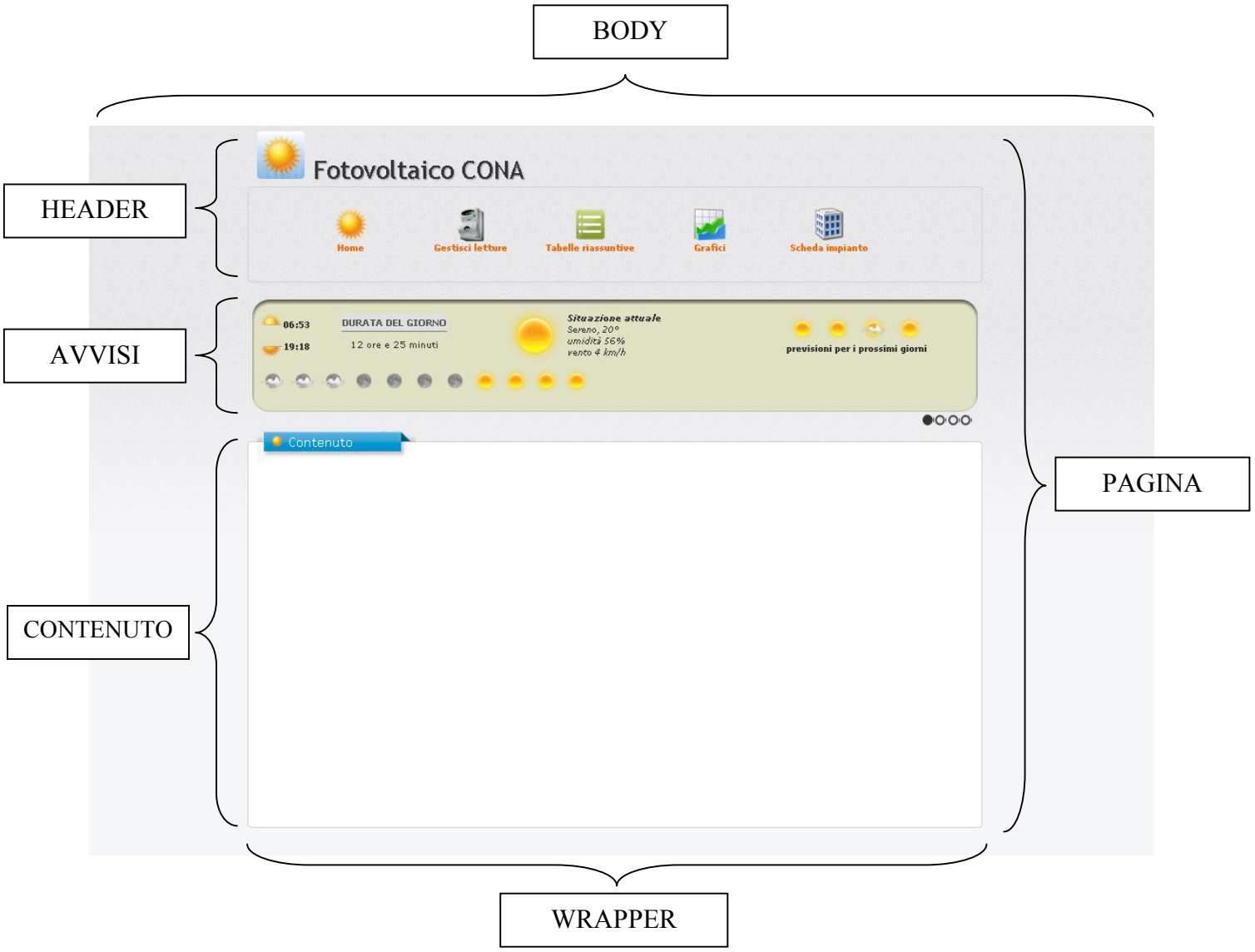


Fig . 18

Nel foglio di stile questi blocchi diventano degli identificatori per i quali è possibile definire le rispettive regole:

```
body{
width: 100%;
height: auto;
margin:0;
padding:0;
background-image: url('../images/sfondo.jpg');
background-repeat: repeat-x;
background-color: #F5F6F7;
}

#wrapper {
width: 900px;
height: auto;
text-align: center;
margin-left : auto;
margin-right : auto;
margin-bottom : 20px;
top: -20px;
min-height: 580px;
margin-bottom:20px;
clear: both;
}

#pagina {
width: 98%;
text-align: center;
padding: 5px;
margin-top: 130px;
margin-left : auto;
margin-right : auto;
height: auto;
}

#sottomenu {
text-align: center;
width: 95%;
height: 90px;
margin-left : auto;
margin-right : auto;
margin-bottom: 60px;
padding: 10px;
border: 1px solid #D3D3D3;
-moz-box-shadow: 0 0 1px 1px #E5E5E5;
-webkit-box-shadow: 0 0 1px 1px #E5E5E5;
box-shadow: 0 0 1px 1px #E5E5E5;
```

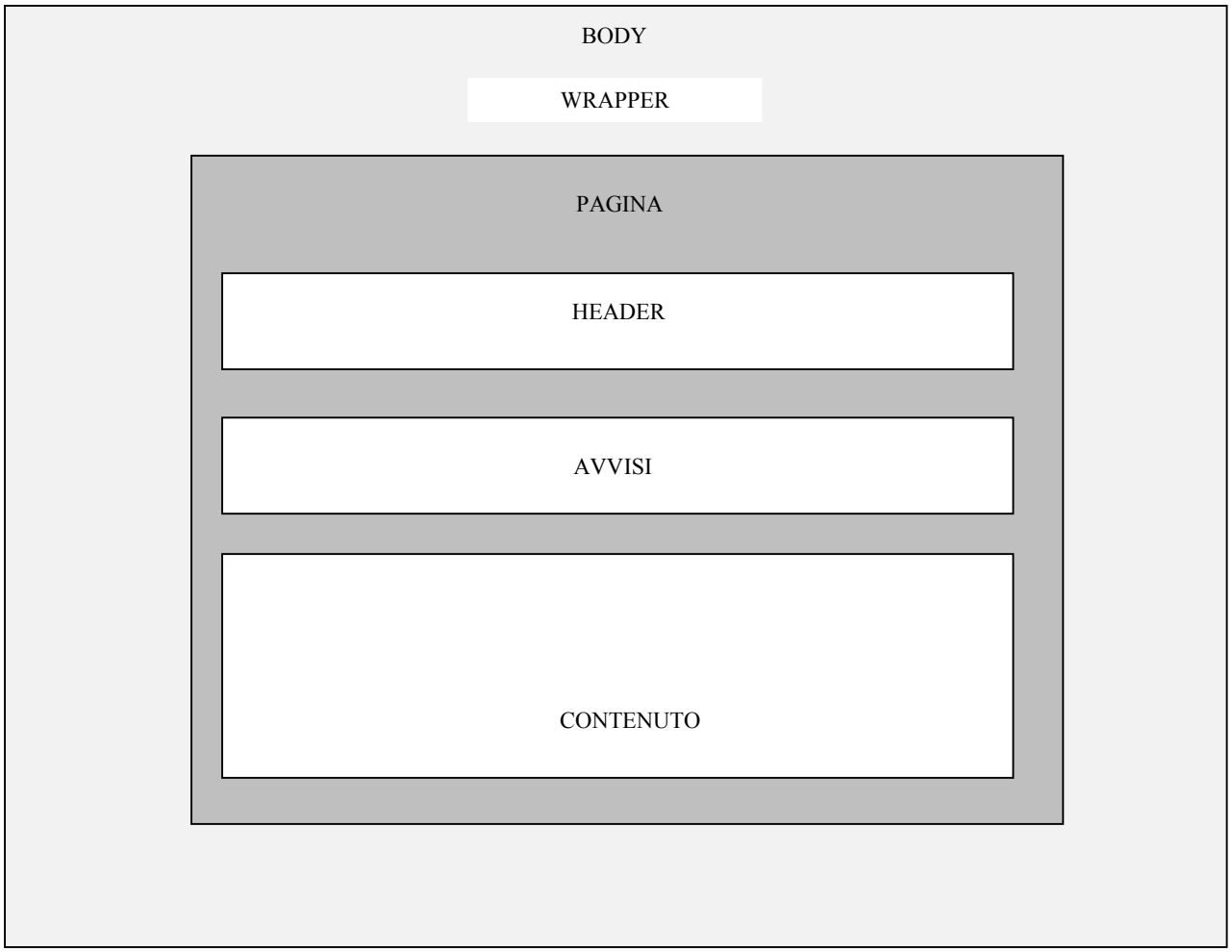
```
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
border-radius: 5px;
}

#avvisi {
position:relative;
width: 95%;
background-color: #E1E1C4;
height: auto;
margin-top: -40px;
margin-left : auto;
margin-right : auto;
margin-bottom: 35px;
padding: 5px 5px 5px 5px;
border-bottom: 1px solid #BCBCBC;
-moz-box-shadow: 0 1px 0 rgba(255,255,255,.2),
inset 0 4px 5px rgba(0,0,0,.6),
inset 0 1px 0 rgba(0,0,0,.6);
-webkit-box-shadow: 0 1px 0 rgba(255,255,255,.2),
inset 0 4px 5px rgba(0,0,0,.6),
inset 0 1px 0 rgba(0,0,0,.6);
box-shadow: 0 1px 0 rgba(255,255,255,.2),
inset 0 4px 5px rgba(0,0,0,.6),
inset 0 1px 0 rgba(0,0,0,.6);
-moz-border-radius: 20px;
-webkit-border-radius: 20px;
border-radius: 20px;
}

#contenuto {
text-align: center;
min-height: 400px;
width: 95%;
background-color: #FFFFFF;
padding-top: 10px;
padding-bottom: 40px;
padding-left: 10px;
padding-right: 10px;
margin-left : auto;
margin-right : auto;
margin-bottom: 50px;
border: 1px solid #D3D3D3;
-moz-box-shadow: 0 0 1px 1px #E5E5E5;
-webkit-box-shadow: 0 0 1px 1px #E5E5E5;
box-shadow: 0 0 1px 1px #E5E5E5;
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
border-radius: 5px;
}
```

In particolare si può notare come l'identificatore *wrapper* viene impostato a larghezza fissa di 800px e centrato alla pagina tramite le due proprietà *margin-left* e *margin-right* impostate al valore *auto*.

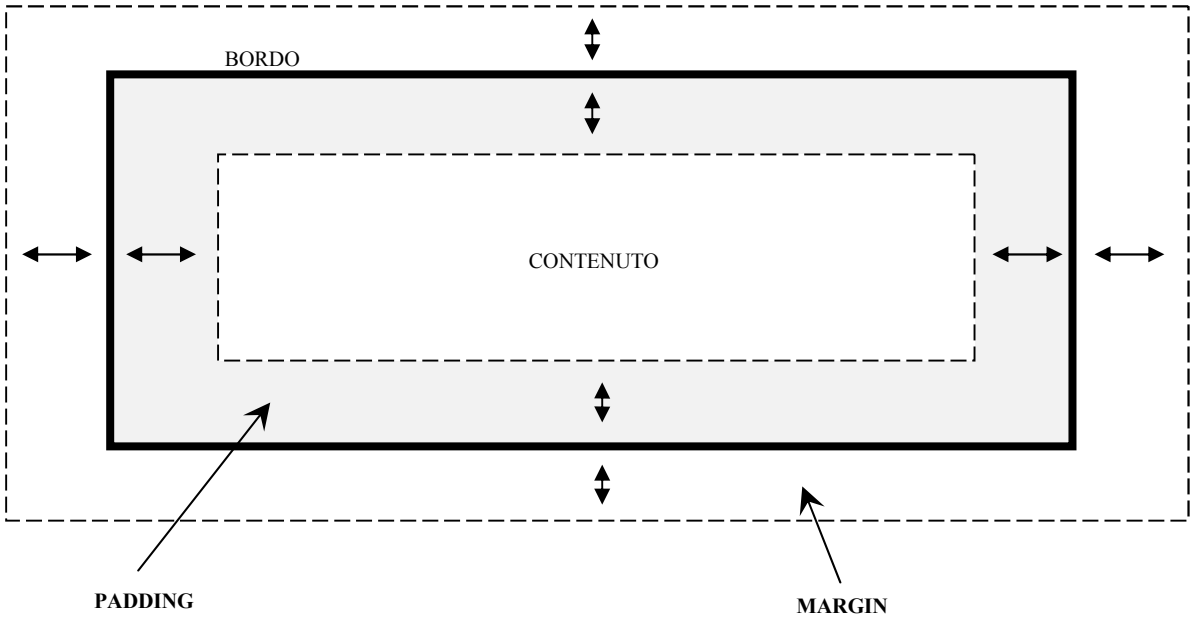
Ma prima di commentare il codice vediamo come vengono annidati i vari identificatori nell'HTML. Una proprietà di altezza o larghezza, dove il valore sia impostato in percentuale, produce infatti un effetto che dipende da dove si trovi quell'identificatore all'interno della pagina.

```
<body>
<div id="wrapper">
        <div id="pagina">
                <div id="header"></div>
                <div id="avvisi"></div>
                <div id="contenuto"></div>
        </div>
</div>
</body>
```

Questo codice dal punto di vista grafico e stilizzato produce:



La proprietà *width* dell'id *pagina* è impostato al 98%. Ciò produce una larghezza effettiva di 784px, essendo la *pagina* posizionata internamente al *wrapper* (con larghezza 800px).
Questo dimensionamento relativo consente di avere un solo valore da modificare, nel caso si voglia aumentare o diminuire la larghezza della pagina. Basterà infatti passare da 800px a 900px per ritrovarsi la *pagina* automaticamente alla larghezza di 882px (98% di 900).
Il valore della larghezza percentuale è impostato anche per l'*header*, la sezione *avvisi* e il *contenuto*. Anche questi dunque si adatteranno ad eventuali modifiche.
Guardiamo ora altre caratteristiche del css utilizzate per la struttura della pagina. Le proprietà *padding* e *margin*, quest'ultima già citata in precedenza, servono per definire le distanze tra il bordo e gli elementi interni, per il primo, ed esterni per il secondo.



Entrambe le proprietà possono essere impostate con un valore singolo

```
padding: 10px;
margin: 5px;
```

oppure con un valore differente per ciascun lato (vediamo l'esempio solo per il *padding*)

```
padding-top: 10px;
padding-right: 5px;
padding-bottom: 10px;
padding-left: 5px;
```

che scritti in maniera compatta diventano

```
padding: 10px 5px 10px 5px;
```

i cui valori identificano in senso orario i 4 lati partendo dal superiore (top – right – bottom - left). Molte altre proprietà del CSS hanno invece un ruolo esclusivamente estetico. Il colore dello sfondo, dei bordi, le sfumature, gli arrotondamenti degli angoli o le ombre. La parte estetica è però ciò che rende la pagina web più piacevole da vedere o che, e da non trascurare, è capace di mettere in risalto delle zone piuttosto che altre. E' quindi una componente che gioca un ruolo essenziale e che vedremo meglio nel prossimo sottocapitolo. Qui vediamo solamente due esempi che sono stati utilizzati per la struttura: l'arrotondamento del bordo e le ombre.

Il primo si ottiene tramite la proprietà border-radius che, vista nella sua versione più semplice, può essere scritta così

```
border-radius: 5px;
```

andando ad impostare un arrotondamento di tutti i 4 angoli. In figura 16 si può vedere l'effetto che si ottiene nel div *#contenuto*.

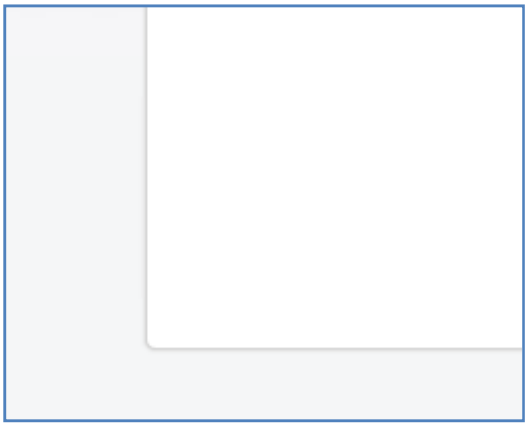


Fig. 19

Come per il *margin* e *padding* si possono impostare gli arrotondamenti in maniera differente per ciascun angolo oltre che dare due valori diversi al raggio, separati dal carattere "/", i quali vengono interpretati come raggio orizzontale e raggio verticale. L'effetto sarà, per esempio, quello di ottenere angoli ellittici.

Oltre al radius, come ho anticipato, l'altro effetto utilizzato è l'ombreggiamento. Questo serve per mettere in rilievo per esempio un box, un'immagine o una scritta. Vediamone un esempio per impostare un'ombra esterna:

```
-moz-box-shadow: 0 0 1px 1px #E5E5E5;
-webkit-box-shadow: 0 0 1px 1px #E5E5E5;
box-shadow: 0 0 1px 1px #E5E5E5;
```

Dove *box-shadow* si usa come regola generale da inserire sempre, mentre -*webkit-box-shadow* e -*moz-box-shadow* servono rispettivamente per browser come Chrome e Safari il primo, Firefox il secondo. Con l'effetto che si può vedere in figura 20 di un leggero rilievo sul bordo del div *#contenuto*.

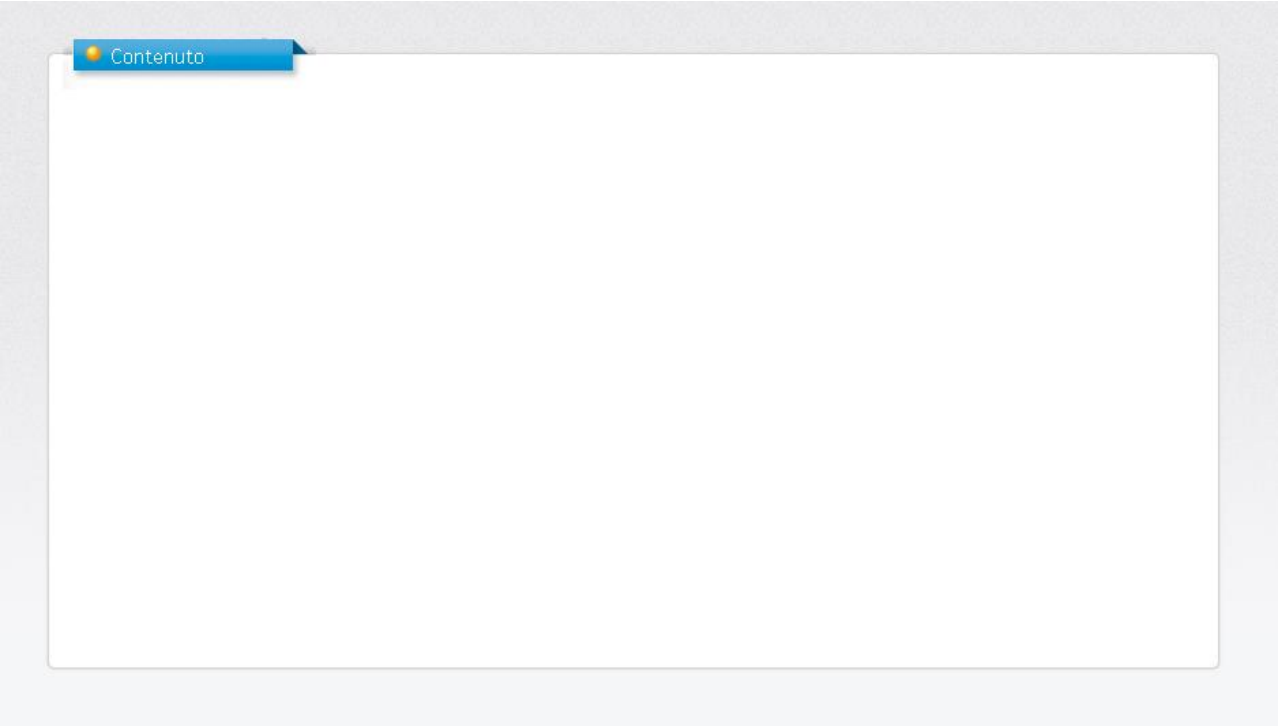


Fig. 20

Si deve chiaramente scegliere un colore per l'ombra. Avendo scelto un grigio, ed essendo grigio sfumato anche il colore di sfondo della pagina, l'effetto è molto leggero, quasi impercettibile.

E' possibile anche creare un'ombra interna che andrà a dare un effetto "incassato" all'elemento per la quale la si utilizza. La proprietà è sempre *box shadow* dove si aggiunge la parola *inset* appena prima di impostare il valore. Andando a modificare l'ombra del div *#contenuto* di prima si avrà:

```
-moz-box-shadow: inset 0 0 1px 1px #E5E5E5;
-webkit-box-shadow: inset 0 0 1px 1px #E5E5E5;
box-shadow: inset 0 0 1px 1px #E5E5E5;
```

Ho utilizzato questa proprietà nel box *#avvisi* come si può notare in figura 18.

Fig. 21

L'effetto percepito è di un ritaglio, di una finestra attraverso la quale visualizzare un elemento che sembra quindi apparire al di "sotto".

c. *Elementi grafici*

Ora mi concentrerò su alcuni altri aspetti grafici utilizzati per il sito. Con ciò, voglio sia commentare e descrivere le regole CSS utilizzate, sia far capire l'importanza di una pianificazione del sito da farsi avendo bene in mente gli elementi grafici di cui si ha bisogno. Tabelle, titoli, sottotitoli, immagini o box. Da definire e formattare *ad hoc* per poi riutilizzarli nelle varie pagine a seconda della necessità.
Elenco qui i principali elementi (che corrispondono agli identificatori o classi del CSS)

- titolo
- sottotitolo
- scheda
- tabella elenco
- tabella sottoelenco
- tabella caratteristiche
- tabella excel

Analizziamo ora per ciascuno la parte di codice CSS, HTML e il risultato.

SCHEDA
Creata per contenere i dati di un singolo record (lettura, indice) come anche per la scheda del mese o dell'anno.

```
.scheda {
width: 90%;
height: auto;
min-height: 200px;
border: 1px solid #CCC;
background-image: -webkit-gradient(linear, left top, left bottom,  from(#EAEAEA), to(#FFF),
color-stop(0.8,#FFF));
-moz-box-shadow: 0 0 1px 1px #EBEBEB;
-webkit-box-shadow: 0 0 1px 1px #EBEBEB;
box-shadow: 0 0 1px 1px #EBEBEB;
-moz-border-radius:5px;
-webkit-border-radius:5px;
border-radius:5px;
margin: 40px auto 10px auto;
padding: 5px 5px 10px 5px;
vertical-align: text-top;
}
```

Impostata come classe per essere anche utilizzata più volte all'interno di una pagina. Posizionata all'interno del div *#contenuto* centralmente e con una larghezza relativa del 90%. Per lo sfondo si è utilizzato il *-webkit-gradient* per ottenere una sfumatura il cui effetto si può vedere nell'immagine qui sotto.

Fig. 22

Nella scheda, contenendo appunto dei dati del database che possono essere eventualmente modificabili o eliminabili, ho aggiunto una "linguetta" in alto a destra dove inserire delle icone che rimandino alle principali azioni (modifica, elimina, precedente o successiva lettura, etc.).

```
.opzioni_scheda {
position: relative;
top: -45px;
right: 5px;
background-image: -webkit-gradient(linear, left top, left bottom,  from(#999), to(#EAEAEA),
color-stop(1,#EAEAEA));
float: right;
width: auto;
height: auto;
border: 1px;
padding: 3px;
-moz-box-shadow: 0 0 1px 1px #EBEBEB;
-webkit-box-shadow: 0 0 1px 1px #EBEBEB;
box-shadow: 0 0 1px 1px #EBEBEB;
border-top: 1px solid #CCC;
border-right: 1px solid #CCC;
border-left: 1px solid #CCC;
-moz-border-radius:5px 5px 0px 0px;
-webkit-border-radius:5px 5px 0px 0px;
border-radius:5px 5px 0px 0px;
}
```

La proprietà *position* del CSS permette il posizionamento di un oggetto all'interno della pagina. Può assumere 4 valori:

- static
- relative
- absolute
- fixed

Il primo è il valore di default (ovvero quello impostato se non espressamente indicato diversamente) e posiziona l'elemento per come viene letto l'HTML nel normale flusso della pagina.
Il secondo, invece, ne indica un posizionamento relativo al suo box contenitore. La posizione viene dunque decisa utilizzando le proprietà *top*, *right*, *bottom* e *left* che permettono di traslare l'elemento di pixel in pixel.

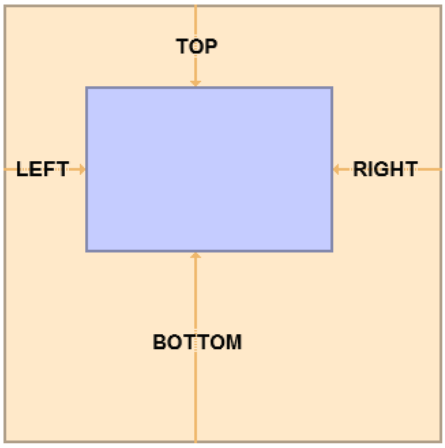


Fig. 23

L'*absolute* invece rimuove l'elemento dal flusso del documento e lo posiziona traslato, tramite le proprietà viste sopra, rispetto al primo contenitore "antenato" con posizionamento diverso da *static*.
Il *position: fixed* in ultimo, sottrae anche qui l'elemento al normale flusso del documento e lo posiziona come per absolute ma rispetto all'elemento radice della pagina, quindi rispetto a tutta la finestra di visualizzazione del browser, e non lo fa scorrere con il resto del documento ma lo mantiene appunto fisso nel punto preimpostato.
L'aver osservato queste importanti proprietà del CSS ci permetti di capire meglio il posizionamento dell'elemento che avrà la classe *.opzioni_scheda*. Interna al box *.scheda,* come da codice HTML,

```
<div class=”scheda”>
        <div class=”opzioni_scheda”></div>
</div>
```

la proprietà *positon: relative* e le distanze impostate tramite right e top la andranno a mantenere nell'angolo in alto a destra come da figura 24.

Fig. 24

TITOLO

Creato come classe dell'elemento html <p>, ovvero il paragrafo, con il seguente CSS:

```
p.titolo {
font-family: Trebuchet MS;
font-size: 25px;
font-weight:bold;
text-align: left;
vertical-align:bottom;
color: #333;
padding: 10px 5px 0px 10px;
text-decoration: none;
clear: both;
margin: 0px 0px 20px 0px;
text-shadow: 1px 1px 1px #999;
}


.titolo img{
position: relative;
background:#FFF;
border:1px solid #D1D1D1;
padding:3px;
max-height: 90px;
max-width: 90px;
left: 0px;
margin-right: 3px;
}
```

produce l'effetto che si può vedere in figura 25, dove viene utilizzato come intestazione dell'oggetto presente nella *.scheda*.

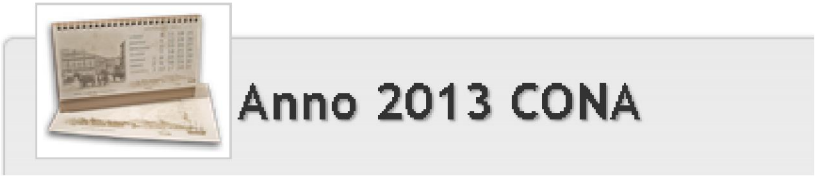


Fig. 25

SOTTOTITOLO

Simile al titolo per il codice CSS utilizzato, dove però cambiano essenzialmente le dimensioni oltre che lo sfondo dell'immagine

Fig. 26

MESSAGGIO

In questo caso ho voluto creare un elemento da poter utilizzare come box per le info organizzate in tre categorie:

- attenzione
- informativo
- conferma
- divieto

Per descriverne il codice mi limiterò al primo. Negli altri casi cambiano solamente i colori di sfondo e del testo.

```
.messaggio{
width: auto;
min-height: 25px;
text-align: center;
font-size: 10px;
font-weight: bold;
padding: 5px 10px 0px 55px;
margin: 30px 10px 30px 10px;
-moz-border-radius:3px;
-webkit-border-radius:3px;
border-radius:3px;
-moz-box-shadow: 0 0 2px 2px #EBEBEB;
-webkit-box-shadow: 0 0 2px 2px #EBEBEB;
box-shadow: 0 0 2px 2px #EBEBEB;
}
```

```
.attenzione{
color: #FFFFFF;
text-transform: uppercase;
text-shadow: 1px 1px 1px #333333;
background-image: -webkit-gradient(linear, left top, left bottom,  from(#F99500), to(#C17400),
color-stop(0.4,#C17400));
}
```

La prima classe, *.messaggio*, serve a definire le dimensioni del box compresi *padding* e *margin,* l'allineamento del testo e le sue dimensioni. Ovvero tutte le caratteristiche che ho inteso "di base" per il contenitore. Con la seconda classe, *.attenzione*, vado a personalizzare qualche caratteristica per meglio esprimere il tipo di informazione da inviare. In questo caso un messaggio di allerta, come si nota dal nome della classe. L'effetto ottenuto dai colori sarà questo:

Fig. 27

Nel caso del messaggio di conferma si sceglierà un verde, nel caso del messaggio di divieto un rosso. E così per altre categorie eventualmente da aggiungere in successive implementazioni del programma. Per importare nel singolo elemento HTML le caratteristiche delle due classi basterà semplicemente scriverle entrambi come valore della proprietà class dell'elemento stesso. In questo modo:

```
<div class="messaggio attenzione"></div>
```

TABELLA CARATTERISTICHE

Questa è la prima delle tre tipologie di tabelle che ho creato per l'esposizione dei dati. Ne mostro subito come appare a video:

Effettivo	98.748 Kwh
Previsto	100.331 Kwh
Incentivo (GSE)	92.402 Kwh (31.971,09 €)
Incentivo (ENEL)	89.993 Kwh (31.137,58 €)
Vendita energia (RID)	89.993 Kwh (7.978,95 € ⓘ)

Fig. 28

```
table.caratteristiche td.nome, table.caratteristiche td.valore{
font-size: 10px;
border-bottom: #c5c5c5 1px dotted;
padding: 5px;
vertical-align:middle;
}
.caratteristiche .nome{
font-size: 10px;
margin-right:8px;
padding:1px 0 1px 0;
text-align: right;
color: #666666;
font-weight: bold;
}
.caratteristiche .valore{
font-size: 10px;
 padding:1px;
}
```

Come si può vedere nella prima parte del CSS è possibile descrivere delle regole per più classi o identificatori. E' sufficiente dichiararli in seguenza e separati da una virgola. Ciò permette di scrivere un codice compatto, senza ridondanze. La classe *.caratteristiche* viene data alla tabella, mentre la classe *.nome* e *.valore*, rispettivamente alla prima e alla seconda colonna di ciascuna riga. L'HTML per definirla sarà:

```
<table class="caratteristiche">
    <tr>
        <td class="nome">Nome</td>
        <td class="valore">Valore</td>
   </tr>
   <tr>
        <td class="nome">Nome</td>
        <td class="valore">Valore</td>
   </tr>
</table>
```

Il codice risulta molto pulito e pratico per essere comodamente utilizzato per gli elenchi di molte voci.

TABELLA

Creata per visualizzare l'elenco dei record di una tabella.

```
table.elenco {
padding: 0;
margin: 0;
border-collapse: collapse;
}

table.elenco thead tr td {
font-family: Trebuchet MS;
font-size: 13px;
font-weight:bold;
color: #D1D1D1;
font-weight: bold;
text-transform:uppercase;
clear: both;
text-align: center;
line-height: 30px;
vertical-align: middle;
height:auto;
background-image: url('../images/sfondo_tabella.jpg');
}

table.elenco tr td{
font-size: 10px;
}

table.elenco tbody tr:hover, table.elenco tbody tr.alterna:hover {
background: #EBEBEB;
}

table.elenco tfoot tr{
height: 15px;
background-image: url('../images/sfondo_tabella.jpg');
}

table.elenco tr.alterna {
background-color: #F4F4F4;
;}
```

Dove nel definirla vengono utilizzati i tag opzionali che vengono utilizzati nelle tabelle per raggruppare e mettere ordine a tutte le righe che compongono una tabella:

- <thead>
- <tbody>
- <tfoot>

Il primo racchiude l'intestazione della tabella, ovvero solitamente quella riga che descrive i valori delle varie colonne. Il secondo invece rappresenta il contenuto. Il terzo la chiusura, la riga finale dove si possono per esempio inserire totali o altre voci.

Per il *thead* e il *tfoot* è stata utilizzata un'immagine nera di sfondo e di conseguenza un colore del testo bianco.

DATA	LETTURA	PRODUZIONE KWH	PRODUZIONE €	TIPO LETTURA	AZIONI
01/03/2013	222.939,71	338,04	150,53	AUTOMATICA	
02/03/2013	223.342,18	402,48	179,23	AUTOMATICA	
03/03/2013	223.755,53	413,35	184,07	AUTOMATICA	
04/03/2013	224.229,23	473,70	210,94	AUTOMATICA	
05/03/2013	224.486,52	257,29	114,57	AUTOMATICA	
06/03/2013	224.518,30	31,78	14,15	AUTOMATICA	
07/03/2013	224.602,18	83,88	37,35	AUTOMATICA	
08/03/2013	224.708,06	105,88	47,15	AUTOMATICA	
09/03/2013	224.814,89	106,83	47,57	AUTOMATICA	
10/03/2013	225.065,00	250,11	111,38	AUTOMATICA	

Fig. 29

La parte HTML avrà questa struttura:

```
<table width="100%" class="elenco">
   <thead>
   <tr>
      <td>Colonna 1</td>
      <td>Colonna 2</td>
   </tr>
   </thead>
   <tbody>
   <tr>
      <td>record 1</td>
      <td>record 1</td>
   </tr>
   <tr>
      <td>record 2</td>
      <td>record 2</td>
   </tr>
   </tbody>
   <tfoot><tr><td colspan="2"></td></tr></tfoot>
</table>
```

La creazione della tabella diventa molto semplice. Basta infatti aggiungere al tag table la classe *.elenco* per ottenere i vari stili già associati a *thead*, *tbody* e *tfoot*.

Come si può notare in figura 29, le righe presentano colori alterni. Questo effetto è ottenuto tramite un piccolo codice javascript che va ad applicare la classe *.alterna* al tag *<tr>* ogni due. Quindi, nonostante la tabella venga scritta con l'HTML visto poco fa, il javascript, nell'evento onload della pagina (quindi nel caricamento) andrà ad aggiungere al posto nostro la classe. Ciò diventa necessario soprattutto quando il contenuto della tabella viene espresso dinamicamente per mostrare i risultati di una query.

```
<script type="text/javascript">
function ColoraRighe(){
var tabelle=document.getElementsByTagName("table");
for(i=0;i<tabelle.length;i++){
   if(tabelle[i].className.indexOf("elenco")>=0){
      trs=tabelle[i].tBodies[0].rows;
      for(j=1;j<trs.length;j+=2)
         trs[j].className="alterna";
      }
   }
}

addEvent(window,'load',ColoraRighe);
</script>
```

SOTTOTABELLA

Alternativa alla tabella precedente, utilizzata per elencare dei dati o dei record che abbiano un risalto differente.

LETTURA	PRODUZIONE KWH	PRODUZIONE EURO	% RISPETTO AL PREVISTO	DATA	AZIONI
311.663,82	339,56	146,10	-10,8	20/09/2013	
311.324,26	351,85	302,77	-7,57	19/09/2013	
310.620,56	488,02	209,98	28,2	17/09/2013	
310.132,54	467,59	201,19	22,83	16/09/2013	
309.664,95	177,30	76,29	-53,42	15/09/2013	
309.487,65	494,21	637,92	29,83	14/09/2013	
308.005,02	297,77	128,12	-21,78	11/09/2013	

Fig. 30

Evito di riportare e commentare il codice perché del tutto simile alla classe *.elenco*.

TABELLA EXCEL

In questo caso invece ho cercato di ottenere una tabella, chiamata non a caso excel, nella quale riportare dati di consuntivo per i quali esprire anche delle unità di misura e un riepilogo nella riga finale. Il risultato assomiglia a ciò che si può infatti ottenere tramite il famoso programma di Microsoft Office.

MESE	EFFETTIVO	PREVISTO	Δ	Δ	GSE	ENEL	RID
	Kwh	Kwh	Kwh	%	Kwh	Kwh	Kwh
settembre 2013	8.637,12	7.613,33	1.023,79	13,45 %			
agosto 2013	15.175,49	14.603,00	572,49	3,92 %	15016	15016	15.016
luglio 2013	16.894,10	16.172,00	722,10	4,47 %	16934	16934	16.934
giugno 2013	15.525,63	15.158,00	367,63	2,43 %	15630	15630	15.630
maggio 2013	13.512,60	14.748,00	-1.235,40	-8,38 %	13309	13309	13.309
aprile 2013	11.523,84	11.925,00	-401,15	-3,36 %	10930	11480	11.480
	81.268,79	80.219,33	1.049,46	1,31 %	71.819,00	72.369,00	72.369,00

Fig. 31

```
table.excel tr.unita_misura{
height: 10px;
}

table.excel tfoot tr td{
font-size: 10px;
text-transform: uppercase;
text-decoration: underline;
font-weight:bold;
color: #5D5D5D;
border-top: 1px solid #999999;
border-bottom: 1px solid #999999;
height: 30px;
}
```

E' bastato definire una classe aggiuntiva *.unità_misura* e applicare un diverso stile al *tfoot.*

Come si è visto, il CSS diventa un supporto fondamentale per organizzarsi la grafica delle pagine di un sito. Resta poi al programmatore avere ben chiaro in mente quali siano gli obiettivi da raggiungere e creare tutti gli elementi di cui si ha bisogno. Anche se mi ripeto, ci tengo a sottolineare quanto questo linguaggio faciliti lavoro di programmazione e soprattutto le eventuali modifiche. Si pensi infatti ad un sito dove ciascuno degli elementi sopra descritti compaia in centinaia di pagine. Tramite poche righe CSS è possibile intervenire su ciascuno di essi.

d. Google charts

Vado ora a parlare di uno dei tanti strumenti (tramite API) forniti da Google per i programmatori. Consente di produrre dei grafici per ottenere un'esposizione più chiara dei dati, dei confronti tra questi e del loro andamento nel tempo.
Tramite poche righe di Javascript è possibile ottenere un ottimo risultato. Vediamone un esempio:

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>

<script type="text/javascript">
google.load("visualization", "1", {packages:["corechart"]});
google.setOnLoadCallback(drawChart);
function drawChart() {
var data = google.visualization.arrayToDataTable([
['Mese', 'Effettivo', 'Previsto', 'GEN-<%=Ucase(Left(MonthName(Month(Date())),3))&"
"&Year(Date())-1%>'],
['',<%=Replace(Round(produzione_anno_kwh,0),",",".")%>,<%=Replace(Round(previsto_an
no,0),",",".")%>,<%=Replace(Round(produzione_parte_anno_prima_kwh,0),",",".")%>]
]);

var options = {
  fontSize: 12,
  vAxis: {minValue:0, title: 'Kwh'},
  legend: {position:'bottom'}
};

var chart = new google.visualization.ColumnChart(document.getElementById('anno'));
  chart.draw(data, options);
}
</script>
<div id="anno" style="width:100%; height:200px;"></div>
```

La prima riga serve per incorporare le JavaScript API (jsapi) dal server di Google. Con queste si hanno quindi a disposizione tutte le librerie necessarie al funzionamento dello script. Successivamente si utilizzano i vari comandi presenti nella guida (https://developers.google.com/chart/) per ottenere il grafico desiderato. Non mi dilungherò sulla descrizione dei vari grafici ma, tramite l'esempio sopra, mostro quanto sia semplice ottenere un grafico a barre.
Tramite *google.visualization.arrayToDataTable* si passano i dati con i quali costruire il grafico

```
var data = google.visualization.arrayToDataTable([
  ['Year', 'Sales', 'Expenses'],
  ['2004',  1000,      400],
  ['2005',  1170,      460],
  ['2006',  660,       1120],
  ['2007',  1030,      540]
]);
```

espressi tramite un array dove il primo valore serve da intestazione. In quest'ultimo esempio di codice (preso dal tutorial di Google) ci sono 4 valori nell'array che producono ciascuno un gruppo a tre barre. Tornando ora all'esempio che ho creato appositamente per il mio sito, il risultato è il seguente:

Fig. 32

I dati che vengono elaborati sono dati dal seguente array

```
  ['Effettivo', 'Previsto', 'GEN-<%=Ucase(Left(MonthName(Month(Date())),3))&"
"&Year(Date())-1%>'],
['',<%=Replace(Round(produzione_anno_kwh,0),",",".")%>,<%=Replace(Round(previsto_an
no,0),",",".")%>,<%=Replace(Round(produzione_parte_anno_prima_kwh,0),",",".")%>]
]);
```

Dove viene quindi fatto un confronto tra la produzione che c'è stata nell'anno solare in corso (in blu), il previsto dello stesso periodo e la produzione che c'è stata nello stesso periodo dell'anno precedente.

5. Inserimento letture e calcolo produzioni

Andiamo ora a vedere come avviene, dal punto di vista pratico, l'inserimento delle letture e il calcolo delle produzioni. Vedremo lo stesso, nel capitolo successivo, per quanto riguarda gli indici.
Per l'inserimento della lettura giornaliera è stato previsto sia un metodo manuale che automatico. Il primo è da intendersi solo per quei periodi in cui ci sia un black-out nella connessione internet o altri gravi problemi tecnici. L'inserimento manuale può infatti essere fonte di errori per sbadatezza o pressapochismo. Per avere il conteggio della produzione giornaliera è infatti essenziale avere le due letture prese allo stesso orario. Ancora meglio se direttamente dopo il tramonto. Il PLC che effettua il monitoraggio è programmato per "fotografare" lo stato dell'impianto alla stessa ora tutti i giorni. Tale valore viene quindi inserito in un file .CSV del mese in corso tenendo memoria delle precedenti letture fino ai 6 mesi precedenti.
Questi file vengono poi resi disponibili tramite internet per essere letti da remoto. Come anticipato nell'introduzione, tali dati sono visualizzabili facilmente tramite programmi specifici. Per poterli però manipolare comodamente tramite l'interfaccia web è necessario importarli.

a. Lettura giornaliera automatica da file CSV

Per risolvere questo problema, e rendere la procedura completamente autonoma, ho scelto un web server che fornisse la possibilità di esecuzione automatica di una pagina specifica ad un orario e con una frequenza che si preferisce. Solitamente questo servizio è limitato ad un certo numero massimo di pagine.
Ma vediamo ora nello specifico come ho organizzato lo script.

```
'LEGGO IL MESE DEL DATO DA SCRIVERE DA REMOTO

on error resume next

StrURL="http://indirizzo_ip_pubblico_plc/DataLog.html?FileName="&mese&".csv"

if mid(StrURL,1,3)="www" then StrURL="http://"&StrURL
Set objXMLHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
objXMLHTTP.Open "GET", StrURL, false
objXMLHTTP.Send
```

In questa parte inizializzo il codice con l'url presso il qauale trovare il file csv. Al posto della dicitura "indirizzo_ip_pubblico" si dovrà scrivere l'ip (se statico) o il DNS. Effettuo quindi la lettura del file tramite l'oggetto ServerXMLHTTP.

Per controllare ora il csv e trasferirmi solo la lettura o le letture mancanti, trasferisco i dati in un file csv temporaneo che salvo nel server locale.

```
'TRASFERISCO IL CONTENUTO IN UN FILE CSV

Set fsoMyFile = CreateObject("Scripting.FileSystemObject")
Set tsTextStream = fsoMyFile.OpenTextFile(Server.MapPath("lettura_automatica.csv"), 8,
True)
contenuto = CStr(objXMLHTTP.ResponseText)
tsTextStream.Write (contenuto)
tsTextStream.Close
Set fsoMyFile = Nothing
Set objxml = Nothing
```

Il file creato si chiamerà *lettura_automatica.csv* che verrà quindi sovrascritto ogni volta. Ho quindi previsto una gestione degli errori in caso di mancato collegamento o timeout (errore numero 80072ee2)

```
'GESTIONE ERRORI


If Err.Number= "80072ee2”  Then
        Set mySmartMail = Server.CreateObject("aspSmartMail.SmartMail")
        mySmartMail.SenderName = " Gestione Fotovoltaico"
        mySmartMail.SenderAddress = "mail_di_invio"
        mySmartMail.Recipients.Add ("mail_destinatario")
        mySmartMail.Subject = "La lettura automatica non è andata a buon fine"
        mySmartMail.Body = "Ciao,  invio questa mail a titolo informativo poichè il mio
tentativo di lettura automatica delle produzioni dell'impianto fotovoltaico non è andato a buon
fine. Ritenterò come sempre domani alla solita ora. "
        mySmartMail.SendMail
        Set mySmartMail = nothing
        blocca_script = 1
Err.Clear
End If
```

In questo caso viene inviata una email utilizzando *mySmartMail* di *asp*. Nel caso non ci siano problemi, procedo invece con la lettura del file csv temporaneo appena creato:

```
Do While Not objApriFile.AtEndofStream
valore_da_inserire = "0"
contatore = contatore + 1
riga=objApriFile.ReadLine
if contatore>1 and contatore<33 then

        valori=Split(riga,",")
        conta_valori=0
        for each valore in valori
        conta_valori=conta_valori+1

        if conta_valori=2 then
        Data_lettura = valore

        ApriRs "SELECT * FROM letture WHERE data = #"& valore &"# AND impianto =
'"& "CONA" &"'", rs
        if rs.eof then
        valore_da_inserire = "1"
        end if
        ChiudiRs(rs)

        End if

        If valore_da_inserire = “1” then

        if conta_valori=4 then
        EnAttOg = valore
        elseif conta_valori=5 then
        EnIndOg = valore
        elseif conta_valori=6 then
        EnCapOg = valore
        elseif conta_valori=7 then
        EnApOg = valore

        Etc..

        end if

        end if

        next
end if
```

Effettuo la lettura riga per riga, andando a controllare per ciascuna il secondo campo. Qui trovo la data della lettura. Se per questa non trovo corrispondenza con le letture già presenti nel database allora procedo con il salvataggio tramite delle variabili temporanee di tutti gli altri valori che, per un maggior ordine dello script, procedo solo dopo a memorizzare nel database.

Ci sono quindi due cicli: il primo inizializzato da *Do While Not objApriFile.AtEndofStream* che serve per leggere tutte le righe, il secondo inizializzato da *for each valore in valori* (dove *valori* è un array formato da tutti i campi della riga grazie alla funzione *Split(riga,",")*).

```
        'INSERISCO I DATI NEL DATABASE

if valore_da_inserire = "1" then

ApriRs "SELECT * FROM letture", rs_aggiungi
rs_aggiungi.addnew

rs_aggiungi("data") = formatta_data(Data_lettura)
rs_aggiungi("EnAttOg") = EnAttOg
rs_aggiungi("EnIndOg") = EnIndOg
rs_aggiungi("EnCapOg") = EnCapOg
rs_aggiungi("EnApOg") = EnApOg

        …….
        Etc…
        ……

rs_aggiungi("tipo") = "AUTOMATICA"

rs_aggiungi.update

ChiudiRs(rs_aggiungi)

end if

loop
objApriFile.Close
set objApriFile=nothing
set objFileSy=nothing
```

Dove, sempre solo se *valore_da_inserire = "1",* procedo con il salvataggio nella tabella letture di tutti i valori. Ho tralasciato nei due pezzi di codice l'elenco completo dei valori perché troppo lungo e comunque non interessante ai fini dello spiegare il funzionamento del programma.

Considerando ora il caso del primo del mese e dovendo ancora recuperare l'ultima o le ultime letture del mese precedente, si può inizializzare la variabile mesi in questo modo

```
mese_corrente = Monthname(Month(Date()))
mese_scorso = Monthname(Month(DateAdd("m",-1,Date())))
mesi_controllare = mese_corrente&","&mese_scorso

for each mese in Split(mesi_controllare,",")

        PARTE SCRIPT DI CONTROLLO E LETTURA VISTO SOPRA

next
```

In questo caso si è inserito un ciclo a soli due valori, il mese in corso e quello precedente. Si può anche pensare di estendere tale precauzione a tutti i mesi precedenti ma la "pesantezza" del codice potrebbe generare problemi (mandare in timeout il PLC o il server dove viene processata la pagina).

b. *Lettura manuale di adeguamento*

Come già descritto nell'introduzione, le letture fornite dal PLC possono differire da quelle presenti nel contatore dell'ENEL. Quest'ultima lettura è quella per la quale vengono calcolate le produzioni ed emessi i pagamenti da parte del GSE. Questa discrepanza è dovuta principalmente a due fattori:

- la distanza tra il PLC e il contatore ENEL e le relative perdite o dispersioni dovute al cavo
- la spesa energetica per il mantenimento dei vari apparati installati (inverter, PLC, router, impianto di sicurezza e videosorveglianza)

Ogni 3 o 4 mesi è bene dunque prendere una lettura reale e distribuire l'eventuale differenza trovata con la corrispondente (dello stesso giorno) lettura del PLC.
Ho quindi previsto un form di inserimento dove, prima dell'invio definitivo, mostro a video su quali letture verrà effettuata la modifica. Per semplificare il problema si permette l'invio di una lettura di allineamento solamente con data successiva all'ultima già inserita.

Fig. 33

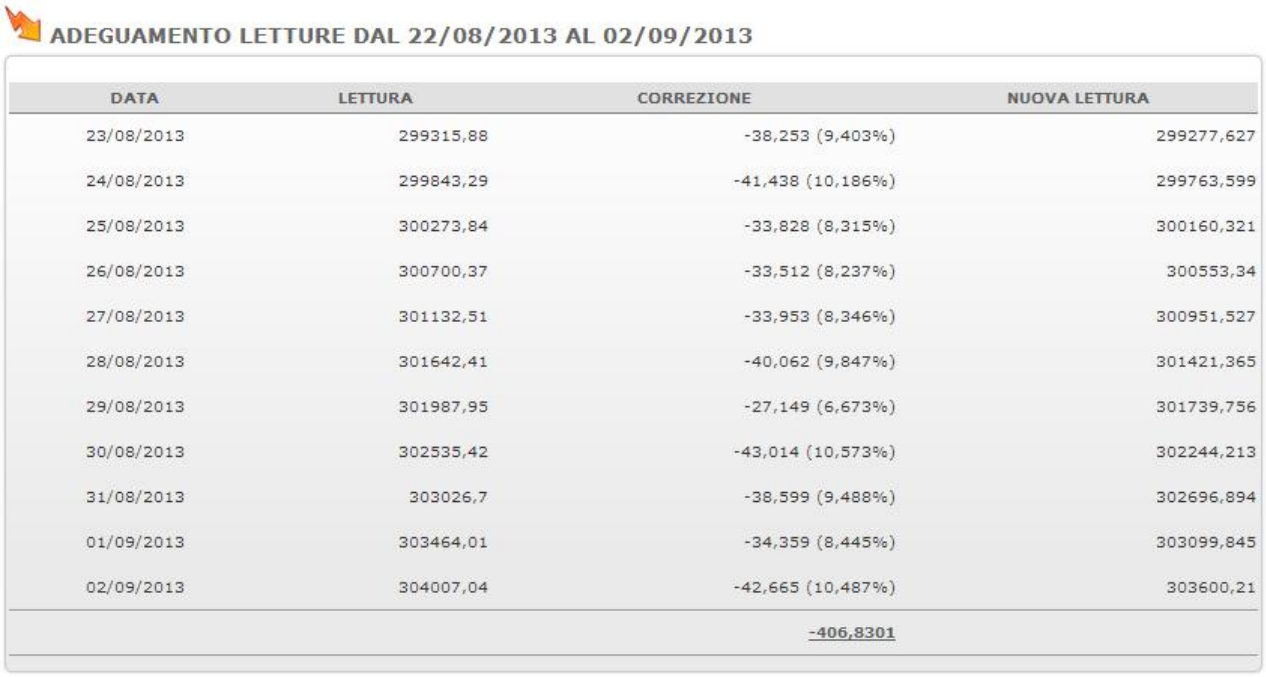
In figura 33 si può vedere il form come si presenta inizialmente. Si inserisce la lettura e si sceglie la data. Premendo poi il pulsante “verifica”, senza che sia necessario il ricaricamento dell’intera pagina, si attende sotto il riepilogo delle modifiche che ho qui diviso in tre immagini per praticità.

```
<div class="pulsante" onClick="xmlhttpPost('controlla_inserimento_lettura_adeguamento.asp',
'lettura', 'risultati','<img
src=\'../images/caricamento.gif\'>');mostra('risultati');">Verifica</div></div>
```

All’evento onclick, viene richiamata una funzione *xmlhttpPost* che utilizza lo script *FORM SUBMIT WITH AJAX* di Simone Rodriguez versione 1.2. E’ sufficiente passare come parametri la pagina da andare a processare, il form dal quale si passano i valori e l’id del box nel quale si vogliono visualizzati i risultati.

Fig. 34

Per prima cosa viene mostrata la corrispondente lettura da PLC e di conseguenza la differenza (in negativo) con la lettura di allineamento appena inserita. In rosso si evidenzia il nuovo valore cumulato. Questo serve a fare da memoria delle differenze che, negli anni, si accumulano tra contatore ENEL e PLC. Serve soprattutto da sommare alla future letture quotidiane che dovranno anche quelle adeguarsi a tale correzione.



ADEGUAMENTO LETTURE DAL 22/08/2013 AL 02/09/2013

DATA	LETTURA	CORREZIONE	NUOVA LETTURA
23/08/2013	299315,88	-38,253 (9,403%)	299277,627
24/08/2013	299843,29	-41,438 (10,186%)	299763,599
25/08/2013	300273,84	-33,828 (8,315%)	300160,321
26/08/2013	300700,37	-33,512 (8,237%)	300553,34
27/08/2013	301132,51	-33,953 (8,346%)	300951,527
28/08/2013	301642,41	-40,062 (9,847%)	301421,365
29/08/2013	301987,95	-27,149 (6,673%)	301739,756
30/08/2013	302535,42	-43,014 (10,573%)	302244,213
31/08/2013	303026,7	-38,599 (9,488%)	302696,894
01/09/2013	303464,01	-34,359 (8,445%)	303099,845
02/09/2013	304007,04	-42,665 (10,487%)	303600,21
		-406,8301	

Fig. 35

In figura 32 invece, viene dettagliata la correzione da fare sulle letture precedenti (e successive alla precedente lettura di allineamento). Per ciascuna viene calcolato un peso percentuale che permette di essere più corretti nella distribuzione della differenza di 406,830 Kwh prima osservata.
Riporto ora qui sotto la parte di codice che esegue questo calcolo:

```
correzione_cumulata = 0
nuove_precedenti = ""
do while not rs_letture.eof
peso                          =
Round(peso_lettura(rs_letture("data"),controlla_data_lettura_precedente+1,data,impianto),10)
correzione                    = (controlla_differenza * peso) / 100
correzione_cumulata           = Ccur(correzione_cumulata) + Ccur(correzione)
nuova                         = Ccur(correzione_cumulata) + rs_letture("lettura")
%>
   <tr>
      <td align="center"><%=rs_letture("data")%></td>
      <td align="right"><%=rs_letture("lettura")%></td>
      <td align="right"><%=Round(correzione,3)%> (<%=Round(peso,3)%>%)</td>
      <td align="right"><%=Round(nuova,3)%></td>
   </tr>
<%rs_letture.movenext
loop)%>
```

Dove viene utilizzata la funzione *peso_lettura* che ho creato a parte:

```
Function peso_lettura(dataf,daf,af,impiantof)

peso_lettura =
(produzione_kwh(data_lettura_precedente(dataf,impiantof)+1,dataf,impiantof)/produzione_kwh(daf,af,impiantof))*100

End Function
```

Che esegue la divisione tra la produzione relativa alla lettura considerata e la produzione dell'intero periodo.

LETTURE SUCCESSIVE AL 02/09/2013 ALLE QUALI TOGLIERE LA DIFFERENZA DI -406,830

DATA	LETTURA	CORREZIONE	NUOVA LETTURA
03/09/2013	304533,81	-406,830	304126,81
04/09/2013	305041,9	-406,830	304634,9
05/09/2013	305490,07	-406,830	305083,07
06/09/2013	305997,86	-406,830	305590,86
07/09/2013	306488,59	-406,830	306081,59
08/09/2013	306792,31	-406,830	306385,31
09/09/2013	307313,16	-406,830	306906,16
10/09/2013	307707,25	-406,830	307300,25
11/09/2013	308005,02	-406,830	307598,02
14/09/2013	309487,65	-406,830	309080,65
15/09/2013	309664,95	-406,830	309257,95
16/09/2013	310132,54	-406,830	309725,54
17/09/2013	310620,56	-406,830	310213,56
19/09/2013	311324,26	-406,830	310917,26
20/09/2013	311663,82	-406,830	311256,82
21/09/2013	312095,45	-406,830	311688,45
22/09/2013	312447,18	-406,830	312040,18
23/09/2013	312927,46	-406,830	312520,46

Invia

Fig. 36

Infine, in figura 36, viene mostrato come questo valore vada a ritoccare le eventuali letture da PLC successive alla data della lettura di allineamento inserita. Per queste andrà infatti semplicemente sottratta la cifra di 406,830 Kwh.

a. *Calcolo produzioni in Kwh e in €*

Dopo aver salvato i dati della lettura giornaliera sul nostro database possiamo ora passare alla parte di elaborazione dei dati. La funzione più importante sarà quella del calcolo della produzione. L'ho voluta rendere il più flessibile possibile e soprattutto in grado di gestire in futuro eventuali anomalie nel sistema. Bisogna sempre considerare, infatti, che il PLC potrebbe non funzionare e che alcune letture potrebbero andar definitivamente perse. Ho quindi immaginato una situazione di questo tipo:

Fig. 37

Dove a fronte di una richiesta per la produzione di un periodo prendo in considerazione 4 letture:

- L1: prima lettura disponibile precedente alla data di inizio produzione richiesta
- L2: prima lettura disponibile successiva alla data di inizio produzione richiesta
- L3: prima lettura disponibile precedente alla data di fine produzione richiesta
- L4: prima lettura disponibile successiva alla data di fine produzione richiesta

Per ricavarmi queste quattro letture posso effettuare delle query sulla tabella letture. Faccio un esempio per L1

```
ApriRs "SELECT TOP 1 lettura,data FROM letture WHERE data < #"& (daf) &"# ORDER
BY data DESC" , rs_L1
if not rs_L1.eof then
L1 = rs_L1("lettura")
DL1 = rs_L1("data")
else
L1 = "N.D."
DL1 = "N.D."
end if
ChiudiRs(rs_L1)
```

Dove vado a prendere il primo record risultante precedente a daf (data di inizio produzione richiesta) e per il quale mi salvo sia la lettura che la sua data.
Fatto questo per tutte e quattro le letture posso passare all'algoritmo per il calcolo della produzione:

```
Function produzione_kwh(daf,af,impiantof)
produzionef = 0

if L2<>"N.D." and L3<>"N.D." then
internaf = L3 - L2
end if

if L1<>"N.D." and L2<>"N.D." then
        primaf = ((L2 - L1) / (DateDiff("y",DL1,DL2))) * ((DateDiff("y",daf,DL2))+1)
elseif L1="N.D." and L2<>"N.D." then
        primaf = L2
else
        primaf = 0
end if
if L3<>"N.D." and L4<>"N.D." then
        secondaf = ((L4 - L3) / (DateDiff("y",DL3,DL4))) * (DateDiff("y",DL3,af))
elseif L3<>"N.D." and L4="N.D." then
        secondaf = 0
else
        secondaf = 0
end if
```

```
if (L2="N.D." and L3="N.D." and L4="N.D.") OR (L1="N.D." and L2="N.D." and L3="N.D."
and L4<>"N.D.") then
produzionef = 0
elseif (L1<>"N.D." and L2="N.D." and L3="N.D." and L4<>"N.D.") then
produzionef = ((L4 - L1) / (DateDiff("y",DL1,DL4))) * ((DateDiff("y",daf,af)+1))
else
produzionef = primaf + internaf + secondaf
end if

produzione_kwh = produzionef

End Function
```

Vado quindi a dividere la produzione in tre parti

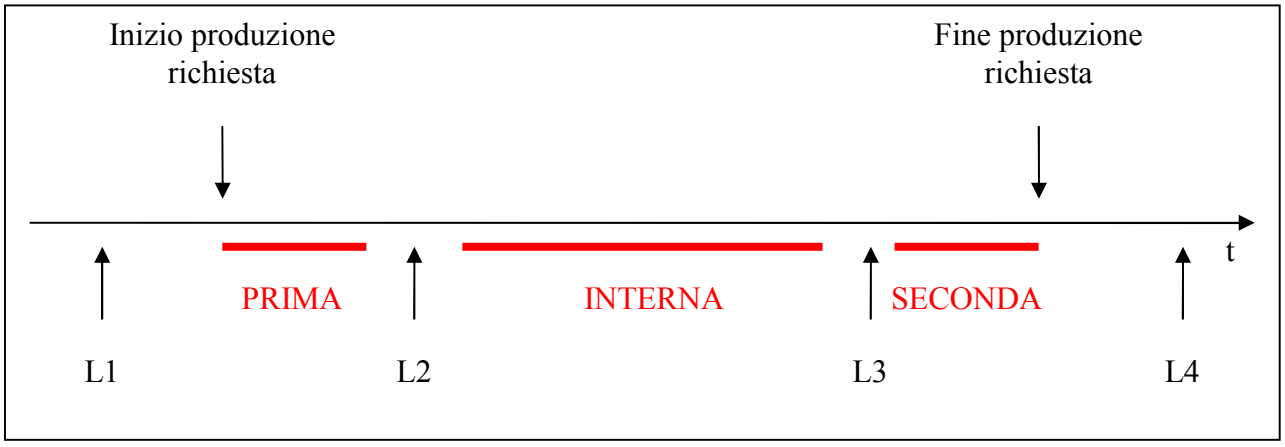


Fig. 38

Per la prima parte della produzione (come anche per la seconda) devo dividere la produzione, ovvero la differenza tra L2 e L1 per il numero di giorni tra le due. Ottenendo così la lettura del singolo giorno posso moltiplicarla per i giorni trascorsi tra inizio produzione richiesta e L2.
Bisogna ovviamente considerare anche i casi particolari. Per esempio il caso in cui L1 non esista (data inizio produzione antecedente alla prima lettura disponibile nel database) o che L2 e L3 non

siano presenti (basterà considerare le due letture esterne e proporzionarle al periodo che ci interessa).

6. Inserimento indici mensili

a. GSE, ENEL e previsto mensile

L'inserimento di questi indici avviene mensilmente. Il previsto mensile è un dato che viene comunicato dalla regione che, in base alla potenza dell'impianto e alla sua posizione, rappresenta il valore in kwh che ci si dovrebbe attendere dalla produzione dei pannelli.
Gli altri due valori sono invece i dati che il Gestore dei Servizi Energetici comunica: il primo effettivo da contatore Enel (in quanto distributore per l'impianto in oggetto) mentre il secondo a volte stimato. Il primo dei due rappresenta la produzione effettiva per la quale ci si deve aspettare il relativo pagamento.
Sono dunque importanti per individuare la corrispondenza tra ciò che si è prodotto e ciò che viene comunicato per il pagamento.

MESE	EFFETTIVO	PREVISTO	Δ	Δ	GSE	ENEL	RID
	Kwh	Kwh	Kwh	%	Kwh	Kwh	Kwh
settembre 2013	9.420,48	8.374,67	1.045,81	12,49 %			
agosto 2013	15.175,49	14.603,00	572,49	3,92 %	15016	15016	15.016
luglio 2013	16.894,10	16.172,00	722,10	4,47 %	16934	16934	16.934
giugno 2013	15.525,63	15.158,00	367,63	2,43 %	15630	15630	15.630
maggio 2013	13.512,60	14.748,00	-1.235,40	-8,38 %	13309	13309	13.309
aprile 2013	11.523,84	11.925,00	-401,15	-3,36 %	10930	11480	11.480
	82.052,15	80.980,67	1.071,48	1,32 %	71.819,00	72.369,00	72.369,00

Fig. 39

7. Servizio meteo

Per avere un quadro più completo sull'efficienza dell'impianto è opportuno sapere e soprattutto avere memoria delle condizioni meteorologiche nelle giornate di produzione. Ci sono molti siti che offrono le previsioni meteorologiche e alcuni tra questi anche delle API per poterle visualizzare sul proprio sito web. Solitamente questi servizi sono però a pagamento. Un servizio affidabile e gratuito è invece quello offerto dalla Microsoft: MSN Weather. Questo fornisce in maniera completamente gratuita le sole condizioni attuali (nel momento in cui si fa la richiesta) e dei quattro giorni successivi. E' quindi necessario provvedere ad un salvataggio delle condizioni meteorologiche, ad una frequenza prefissata, nel database dell'impianto fotovoltaico per avere una memoria storica delle condizioni meteo.

a. *XML della Microsoft per situazione corrente e previsioni*

La risposta viene fornita tramite un file xml. Provando a digitare nel browser il seguente indirizzo:

```
http://weather.msn.com/find.aspx?outputview=search&weasearchstr=bologna
```

Si ottiene la seguente risposta

```
<weatherdata>
        <weather weatherlocationcode="wc:ITXX0006" weatherlocationname="Bologna,
        ITA" zipcode="" weatherfullname="Bologna, Emilia-Romagna,
        Italy" searchlocation="Bologna, BO, Em.Rom.,
        Italy" searchdistance="0" searchscore="0.95" url="http://local.msn.com/worldweather.a
        spx?eid=4378&q=Bologna-ITA" imagerelativeurl="http://blu.stc.s-
        msn.com/as/wea3/i/en-
        us/" degreetype="F" provider="Foreca" isregion="False" region="" alert="" searchresult
        ="Bologna, Emilia-Romagna,
        Italy" lat="44.5048294067383"lon="11.3451595306396" entityid="4378">
        <current temperature="64" skycode="31" skytext="Clear"/>
        </weather>
</weatherdata>
```

In realtà qui non si hanno le condizioni meteorologiche, ma un identificatore della località, il *weatherlocationcode* , che permette, passato come parametro ad un altro link, di ottenere le informazioni desiderate. In questo caso l'identificatore è *wc:ITXX0006.*

```
http://weather.service.msn.com/data.aspx?src=vista&weadegreetype=C&culture=it-
IT&wealocations=wc:ITXX0006
```

Che produce

```
<weatherdata xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance">
<weather weatherlocationcode="wc:ITXX0006" weatherlocationname="Bologna,
Em.Rom." zipcode="40121" url="http://meteo.it.msn.com/local.aspx?wealocations=wc%3aITX
X0006&q=Bologna%2c+Em.Rom.&ctsrc=vista" imagerelativeurl="http://wst.s-
msn.com/i/it/" degreetype="C" provider="Foreca" attribution="Dati forniti da
Foreca" attribution2="©
Foreca" lat="44.5035934" long="11.3407812" timezone="2" alert="" entityid="4378" encodedl
ocationname="Bologna%2c+Em.Rom.">
<current temperature="17" skycode="31" skytext="Sereno" date="2013-09-
24" observationtime="00:50:00" observationpoint="Bologna / Borgo
Panigale" feelslike="17"humidity="72" winddisplay="7 km/h
SO" day="martedì" shortday="mar" windspeed="7"/>
<forecast low="14" high="29" skycodeday="32" skytextday="Sereno" date="2013-09-
24" day="martedì" shortday="mar" precip="2"/>
<forecast low="14" high="28" skycodeday="32" skytextday="Sereno" date="2013-09-
25" day="mercoledì" shortday="mer" precip="2"/>
<forecast low="15" high="29" skycodeday="34" skytextday="Sereno" date="2013-09-
26" day="giovedì" shortday="gio" precip="3"/>
```

```
<forecast low="15" high="28" skycodeday="32" skytextday="Sereno" date="2013-09-
27" day="venerdì" shortday="ven" precip="5"/>
<forecast low="18" high="28" skycodeday="30" skytextday="Parzialmente
nuvoloso" date="2013-09-28" day="sabato" shortday="sab" precip="15"/>
<toolbar timewindow="60" minversion="1.0.1965.0"/>
</weather>
</weatherdata>
```

Abbiamo ora finalmente tutte le informazioni necessarie. Ci resta solamente da leggerle e copiarle nel database alla frequenza desiderata.

b. Funzioni di lettura dell'XML per un utilizzo flessibile

Utilizzo l'oggetto DOM per caricar il file xml.

```
Set objXML = Server.CreateObject("msxml2.DOMDocument.3.0")

objXML.async = false
objXML.setProperty "ServerHTTPRequest", True
objXML.validateOnParse =false
objXML.preserveWhiteSpace = false
```

A questo punto mi creo una funzione per andare a leggere l'id della località dal primo link

```
Function trova_id_localita(localitaf)

MyXML =
objXML.Load("http://weather.msn.com/find.aspx?outputview=search&weasearchstr="&localit
af)

if err.number=80070005 then
trova_id_localita = "N.D."
end if
If Not MyXML Then
trova_id_localita = "N.D."
Else
Set Nodo = objXML.getElementsByTagName("weatherdata")
```

```
For Each objNodi1 In Nodo
For Each objNodo1 In objNodi1.childNodes
Select Case objNodo1.nodeName
Case "weather"
id_localita = objNodo1.getAttribute("weatherlocationcode")
End Select
Next
Next
trova_id_localita = id_localita
end if
End Function
```

Dove, arrivato al tag *weather*, prendo il valore dell'attributo *weatherlocationcode.* Ora utilizzo una funzione simile per la lettura delle condizioni meteo dal secondo link. In realtà ho creato due funzioni, una per leggere la situazione corrente e una per le previsioni. Vado ora a descrivere solamente la prima, che è quella che mi permette di ricavare i dati da salvare nel database. Non essendo una previsione meteo dovrebbe essere molto più affidabile, con dei dati vicini se non identici a quelli reali.

```
Function situazione_corrente(localitaf)

on error resume next
```

```
MyXML =
objXML.Load("http://weather.service.msn.com/data.aspx?src=vista&weadegreetype=C&culture=it-IT&wealocations="&trova_id_localita(localitaf))

if err.number=80070005 then
situazione_corrente = "N.D."
end if
If Not MyXML Then
situazione_corrente = "N.D."
Else
Set Nodo = objXML.getElementsByTagName("weather/current")

For Each objNodo1 In Nodo
temperaturaf = objNodo1.getAttribute("temperature")
condizionif = objNodo1.getAttribute("skytext")
percepitaf = objNodo1.getAttribute("feelslike")
umiditaf = objNodo1.getAttribute("humidity")
velocitaventof = objNodo1.getAttribute("windspeed")
direzioneventof = objNodo1.getAttribute("winddisplay")
Next

situazione_corrente =
temperaturaf&";"&percepitaf&";"&condizionif&";"&umiditaf&";"&velocitaventof&";"&direzioneventof
end if

End Function
```

Come si può vedere all'inizio della funzione, passo direttamente al link, il parametro id località risultante dalla prima funzione vista. Avrei quindi potuto unire le due funzioni in una unica.
Fatto ciò, e dopo aver opportunamente gestito eventuali errori così da non bloccare l'esecuzione dello script ma da permettere comunque una risposta di errore o N.D. (non disponibile), recupero tutti i dati della condizione meteo: temperatura, condizioni, precipitazioni, velocità e direzione del vento.
Unisco questi valori in un'unica stringa (dopo aver inserito un separatore) e la passo come riposta della funzione.

c. Salvataggio della situazione corrente per ogni ora

Avendo già predisposto un file che viene processato automaticamente nel server (lettura_automatica.asp visto in precedenza), inserisco li dentro le funzioni di lettura meteo da

utilizzare con la frequenza che si preferisce. Per il monitoraggio dell'impianto ho deciso di impostarle con una cadenza oraria per poter avere un andamento quasi continuo delle condizioni meteorologiche nell'arco della giornata.

```
ApriRs "SELECT * FROM meteo WHERE data = #"& Date() &"#",rs_meteo
if rs_meteo.eof then
rs_meteo.addnew
rs_meteo("data") = Date()
end if

rs_meteo(Hour(time())) = situazione_corrente("correzzola")

rs_meteo.update

ChiudiRs(rs_meteo)
```

Il codice è molto semplice. Seleziono dalla tabella meteo il record con il campo data corrispondete alla data del giorno. Se questo non fosse presente, come nel caso dell'1 di notte, ne creo uno nuovo. A questo punto inserisco nel campo richiamato dinamicamente in base all'ora *Hour(time()))* la situazione corrente utilizzando la funzione vista in precedenza.
Hour(time())) è una funzione di asp che ritorna l'ora corrente. *Rs_meteo(Hour(time()))),* alle ore 10 del mattino diventa rs_meteo(10). In questo caso, il salvataggio del dato avviene nella decima colonna della tabella. E' importante, avendo scelto la colonna per l'ordine in cui è posizionata e non per il suo nome, lasciare inalterata tale disposizione. La tabella meteo avrà quindi 24 colonne più quella per la data (oltre che il campo contatore che funga da chiave primaria).

d. Alba e tramonto e creazione widget meteo

Ho aggiunto poi il calcolo dell'alba e del tramonto. Questo è uno script gratuito, spesso in versioni amatoriali, che si trova facilmente su internet. Non andrò qui a descriverlo ma, integrato con le funzioni precedenti, mi ha permesso di creare un widget per il meteo da posizionare nella sezione avvisi (per la situazione corrente e per le previsioni) oltre che una barra di riepilogo sull'andamento del giorno nella scheda della singola lettura (figura 40).

Figura 40

Figura 41

Dove le informazioni vengono sostituite da delle icone sulle condizioni meteo e, nel secondo, la durata del giorno viene resa più semplice da osservare grazie a quel box con la sfumatura del giorno e della notte.

```
<%
notte1 = DateDiff("n", FormatDateTime ("00:00", vbShortTime), FormatDateTime (sorge, vbShortTime))
giorno = DateDiff("n", FormatDateTime (sorge, vbShortTime), FormatDateTime (tramonta, vbShortTime))
notte2 = DateDiff("n", FormatDateTime (tramonta, vbShortTime), FormatDateTime ("23:59", vbShortTime)) + 1

pxnotte1 = Round(notte1/1440*700,0)
pxgiorno = Round(giorno/1440*700,0)
pxnotte2 = Round(notte2/1440*700,0)
%>

<table class="riflesso" cellpadding="0" cellspacing="0" width="700" height="35" align="center">
	<tr>
		<td width="<%=pxnotte1%>" style="-webkit-border-bottom-left-radius: 5px; -webkit-border-top-left-radius: 5px; background-image: -webkit-gradient(linear, left top, right top, from(#003195), to(#ECBD00), color-stop(1, #ECBD00));" align="right">
			<img class="immagine" src="../images/meteo/alba.png" style="position:relative; right:-13px; z-index:100;" height="25" title="<%=sorge%>">
		</td>

<td width="<%=pxgiorno%>" bgcolor="#ECBD00" align="center">
		<div class="testo_sfondo grigio" style="position:relative; top:-16px;"><%=ore%> ore e <%=minuti%> minuti</div>
		</td>
		<td width="<%=pxnotte2%>" style="-webkit-border-bottom-right-radius: 5px; -webkit-border-top-right-radius: 5px; background-image: -webkit-gradient(linear, right top, left top, from(#003195), to(#ECBD00), color-stop(1, #ECBD00));" align="left">
		<img class="immagine" src="../images/meteo/tramonto.png" style="position:relative;z-index:100; left:-13px;"height="25" title="<%=tramonta%>">
		</td>
	</tr>
</table>
```

Dopo aver definito in minuti la durata della notte (da mezzanotte all'alba), del giorno (dall'alba al tramonto) e della sera (da tramonto a mezzanotte), creo una tabella di larghezza fissa pari a 700 px e tramite la proporzione durata_notte : 1440 = x : 700 (dove 1440 sono quindi i minuti che compongono una giornata), ottendo le profondità in pixel da dare ai tre settori.
Queste vengono quindi poi impostate come larghezza delle colonne della tabella. Tramite il *-webkit-gradient* sarà poi possibile inserire una sfumatura dal blu al giallo per la prima parte del giorno e viceversa per l'ultima.

8. Le principali pagine di esposizione dei dati

In questo capitolo andrò invece a mostrare quali sono gli effetti complessivi ottenuti descrivendo alcune delle pagine più importanti che compongono il sito.

a. *Home*

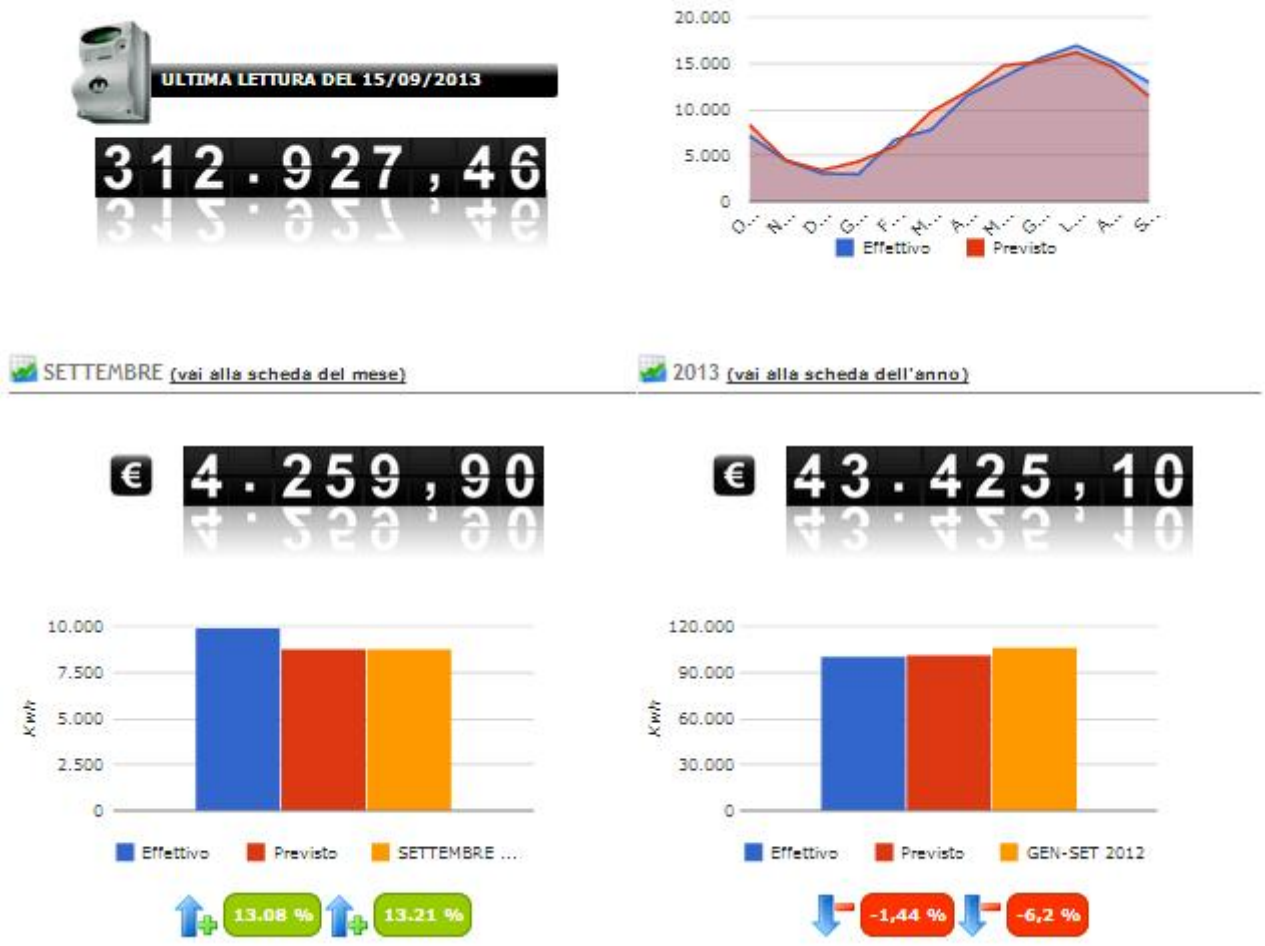


Fig. 42

Nella home page viene fatto il riepilogo della situazione dell'impianto. L'ultima lettura e un piccolo grafico con l'andamento degli ultimi 12 mesi in alto. Più sotto invece la situazione del mese e dell'anno corrente. Per ciascuno di questi ho creato un grafico che metta a confronto la produzione del periodo in corso confrontata a quella prevista e a quella ottenuta nello stesso periodo dell'anno precedente. In questo caso il mese di settembre 2013 viene confrontato con il mese di settembre 2012.

Sotto ai grafici riporto la percentuale di sovraproduzione o sottoproduzione dell'effettivo rispetto agli altri due valori.

b. *Scheda del mese e dell'anno*

Direttamente da quelle due sezioni della home page si può accedere alle schede del mese e dell'anno. Da ciascuna, tramite le frecce in alto a destra della scheda, è possibile passare ai periodi precedenti o successivi.

Fig. 43

Nella scheda metto in risalto con un grafico la produzione effettiva e quella prevista e, di conseguenza, l'indice di efficienza dell'impianto per quel mese. Utilizzando sempre lo stesso stile a contatore visto nell'home page, si possono visualizzare le quote incentivo e le quote vendita prodotte in quel periodo.

Nella seconda metà della pagina ho creato un grafico a barre con tutte le produzioni dei vari giorni del mese. Due linee (produzione media prevista e produzione media effettiva) consentono di avere un impatto immediato sull'andamento giornaliero. Al di sotto si può espandere la pagina per visualizzare l'elenco delle letture ed accedere a quella che più interessa.

LETTURA	PRODUZIONE	DATA	AZIONI
270.957,11	602,74	30/06/2013	
270.354,37	568,54	29/06/2013	
269.785,83	273,56	28/06/2013	
269.512,27	562,67	27/06/2013	
268.949,60	156,85	26/06/2013	

Fig. 44

In figura 44, nel grafico delle produzioni del mese, si può notare come per i primi 18 giorni di produzione ci siano dei valori, a gruppi, uguali tra di loro. Ciò capita quando dei dati del PLC vadano persi per qualche ragione e si sia costretti ad inserirli manualmente. In questo caso sono state inserite delle letture il giorno 6, l'11, il 13 e il 18. Dopo tale data è ripreso il normale funzionamento. Chiaramente, una frequenza di inserimento manuale bassa, genera un grafico poco interessante, poiché non si è più in grado di individuare con precisione le produzioni giornaliere e di conseguenza si hanno meno dati per risalire a guasti o problemi di sottoproduzione.

La scheda dell'anno mantiene un'impostazione simile a quella del mese appena vista. Dopo un riepilogo dei dati di produzione in kwh e €, degli indici e dell'efficienza del periodo, nella seconda parte della pagina il grafico a barre mostra le produzioni mese per mese (dove si può notare giustamente la "campana" di produzione con un picco nei mesi estivi) confrontate ciascuna con il rispettivo previsto.

Fig. 45

In figura 46 invece si ha l'elenco dei mesi in oggetto per poter comodamente accedere alla scheda.

MESE	PRODUZIONE	PREVISTO	EFFICIENZA	AZIONI
GENNAIO	5.757	4.356	32,16 %	
FEBBRAIO	7.680	6.003	27,93 %	
MARZO	13.337	9.753	36,74 %	
APRILE	10.786	11.925	-9,55 %	
MAGGIO	11.953	14.748	-18,95 %	
GIUGNO	16.193	15.158	6,83 %	
LUGLIO	16.667	16.172	3,06 %	
AGOSTO	15.501	14.603	6,15 %	
SETTEMBRE	10.523	11.420	-7,86 %	
OTTOBRE	7.152	8.345	-14,3 %	
NOVEMBRE	4.478	4.498	-0,45 %	
DICEMBRE	3.066	3.406	-9,97 %	

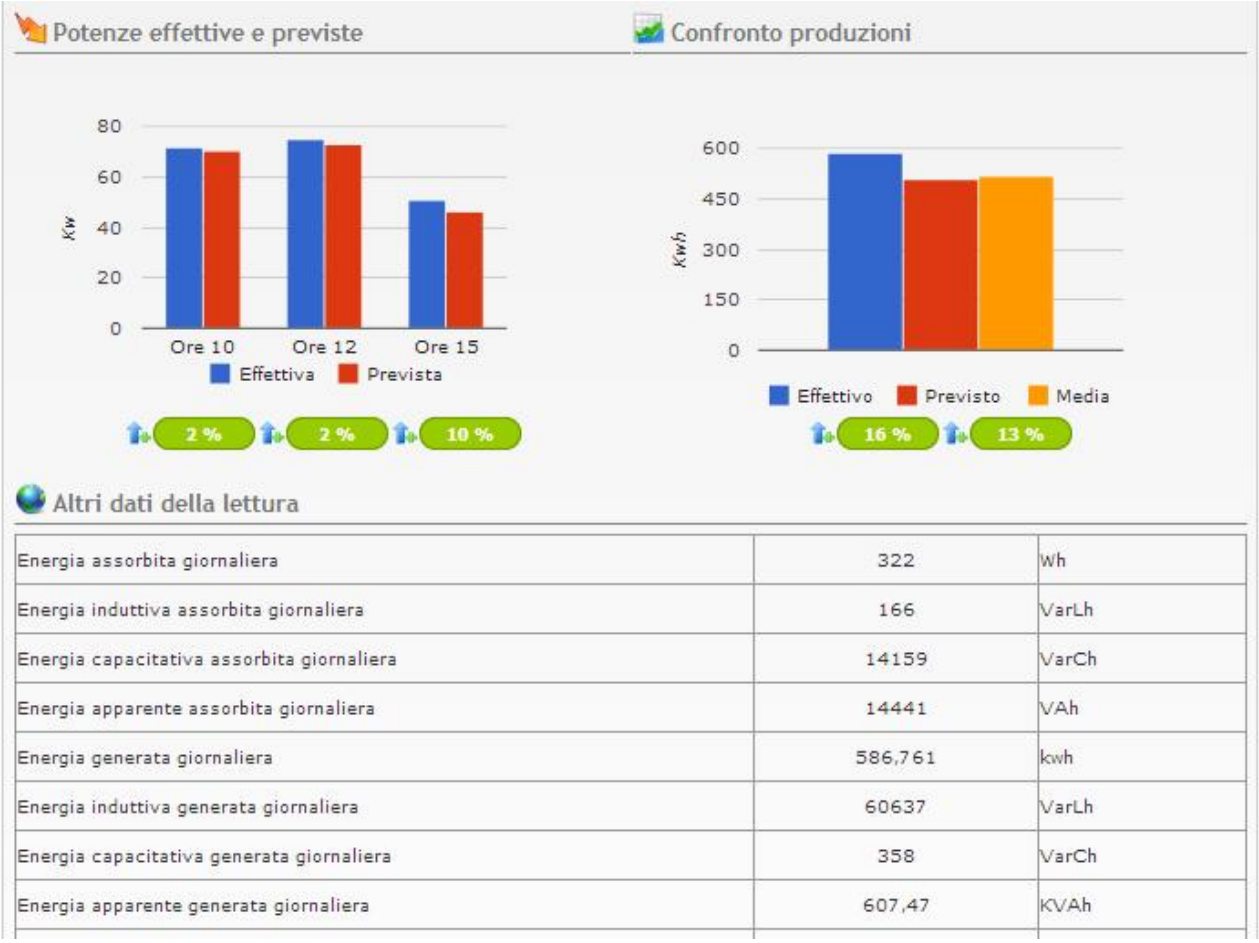
mostra/nascondi mesi

Figura 46

c. *Scheda singola lettura*

Fig. 47

La lettura giornaliera viene invece presentata con una prima parte delle informazioni di base, la produzione in € e in kwh dalla lettura precedente, l'andamento delle condizioni meteo del giorno e, tramite l'icona verde di spunta sull'immagine del PLC, una conferma del fatto che sia stata allineata alla reale lettura del contatore.



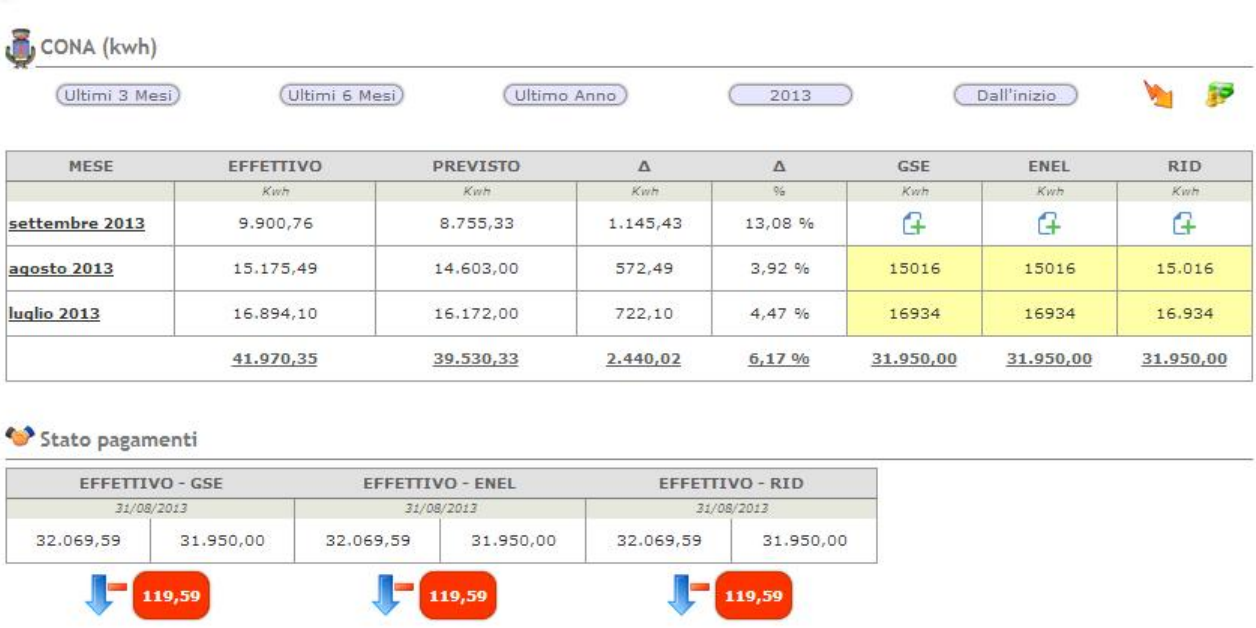
Energia assorbita giornaliera	322	Wh
Energia induttiva assorbita giornaliera	166	VarLh
Energia capacitativa assorbita giornaliera	14159	VarCh
Energia apparente assorbita giornaliera	14441	VAh
Energia generata giornaliera	586,761	kwh
Energia induttiva generata giornaliera	60637	VarLh
Energia capacitativa generata giornaliera	358	VarCh
Energia apparente generata giornaliera	607,47	KVAh

Fig. 48

Un dato molto importante che proviene dalla lettura giornaliera è quello sui valori di potenza effettiva o prevista fotografati in tre differenti orari della giornata: alle 10:00, alle 12:00 e alle 15:00. Questi devono essere allineati, lo scarto dev'essere incluso in un'oscillazione che non superi il 5-10%. Tale dato può dunque servire da campanello d'allarme per individuare prontamente eventuali anomalie dell'impianto. Un problema diffuso è relativo ai fusibili. Se uno di questi dovesse saltare, manderebbe in blackout un'intera fila di pannelli con una conseguente perdita di produzione.

Nel caso in cui non ci sia un controllo quotidiano del sito e della lettura giornaliera, si può prevedere un invio di un allarme per email o sms per segnalare subito la differenza di valori.

d. Tabella riassuntiva



MESE	EFFETTIVO	PREVISTO	Δ	Δ	GSE	ENEL	RID
	Kwh	Kwh	Kwh	%	Kwh	Kwh	Kwh
settembre 2013	9.900,76	8.755,33	1.145,43	13,08 %			
agosto 2013	15.175,49	14.603,00	572,49	3,92 %	15016	15016	15.016
luglio 2013	16.894,10	16.172,00	722,10	4,47 %	16934	16934	16.934
	41.970,35	39.530,33	2.440,02	6,17 %	31.950,00	31.950,00	31.950,00

EFFETTIVO - GSE		EFFETTIVO - ENEL		EFFETTIVO - RID	
31/08/2013		31/08/2013		31/08/2013	
32.069,59	31.950,00	32.069,59	31.950,00	32.069,59	31.950,00

Figura 49

In questa pagina si fa invece un riepilogo dei valori di un determinato periodo:

- Ultimi 3 mesi
- Ultimi 6 mesi
- Ultimi 12 mesi
- Anno in corso
- Dall'inizio

E' possibile scegliere il periodo da visualizzare cliccando sul relativo pulsante al di sopra della tabella. Verrà semplicemente richiamata la stessa pagina inviando un parametro che ne indichi il periodo. E' inoltre stata inserita la possibilità di passare dalla visualizzazione in kwh a quella in €. Anche in questo caso l'invio di un parametro veicolerà lo script all'esecuzione dei calcoli relativi.

Da questa pagina, cliccando sull'indice, si può accedere alla sua modifica, se presente, o all'inserimento.

Dopo la tabella si trova lo stato dei pagamenti che, sempre relativamente al periodo e al valore di € o kwh selezionati, mostra la differenza tra la produzione effettiva e i tre indici da confrontare, rendendo facile un conteggio di riepilogo da farsi altrimenti manualmente.

e. Scheda indice

La scheda del singolo indice presenta pochi dati. Importante quello sullo stato, pagato o in pagamento, che viene poi visualizzato, nella tabella di riepilogo vista prima, tramite un colore (verde per pagamento ricevuto, giallo se in pagamento).

Fig. 50

In alto a destra si può procedere alla modifica o all'eliminazione dell'indice. In figura 51 il form di inserimento.

Fig. 51

f. Scheda dell'impianto e gestione documenti

Fig. 52

Infine, anche prevedendo la gestione di più impianti, ho creato una pagina con i dati principali. La sezione "Info generali" visualizza dei dati fissi, non prelevati dal database. E' quindi l'unica pagina statica del sito (a parte il dato di ultima lettura). Per modificare il valore di inizio produzione dovrei intervenire infatti sul codice HTML.

In realtà, la visualizzazione dei documenti avviene tramite ASP che estrae tutti i documenti in una specifica cartella.

Ho creato, a riguardo, una funzione che svolga questo compito.

```
Set FileSystemObject = Server.createObject("Scripting.FileSystemObject")

Function visualizza_documenti_cartella(urlf)

Set cartellaf = FileSystemObject.GetFolder(Server.MapPath(urlf))
Set documentif = cartellaf.Files

conta_documentof = 0

for each documentof in documentif
estensionef = estensione_file(documentof)
nomef = nome_file(documentof)
if controlla_estensione_tipologia_file(documentof,"immagini") = 0 then
conta_documentof = conta_documentof + 1
%>
<a
href="../file_manager/vedi.asp?url=<%=radice_pagina_corrente()&urlf&"/"&documentof.name
%>&nome=<%=nomef%>&estensione=<%=estensionef%>&dimensione=<%=dimensione_fil
e(documentof)%>" target="_blank">
<div class="documento"><img src="<%=immagine_file(estensionef,"piccola")%>"
title="<%=Ucase(estensionef)%>"><%=primalettera(nomef)%></div>
</a>
<%
end if
next
if conta_documentof = 0 then
        if controlla_esistenza_cartella(urlf&"/nascosti") then
                if cartella_vuota(urlf&"/nascosti") then
                response.write("<div id=testo_corsivo> non ci sono documenti </div>")
                end if
        else
        response.write("<div id=testo_corsivo> non ci sono documenti </div>")
        end if
end if
Set objFso_doc = Nothing
Set cartellaf = Nothing
Set documentif = Nothing

End Function
```

Utilizzando l'oggetto *Scripting.FileSystemObject,* inizializzato ad inizio pagina, e in particolare la sua proprietà *GetFolder,* si ottiene un array di elementi (files) che è quindi possibile esaminare tramite un ciclo *For each….next.* Ciascun documento viene visualizzato con un'immagine che ne evidenzia il formato (pdf, doc, etc) e può essere aperto tramite un visualizzatore come da figura 53.

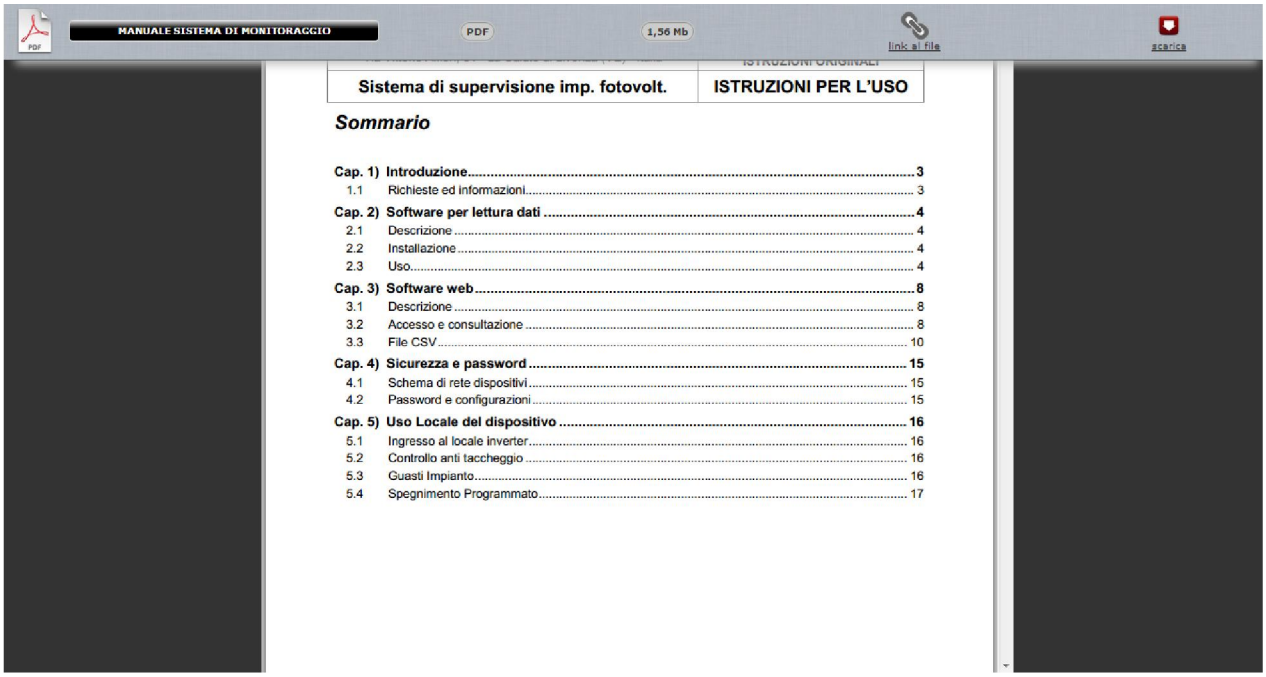


Fig. 53

Per l'upload ho preparato una maschera dove poter scegliere il file dal proprio pc e inviarlo al server.

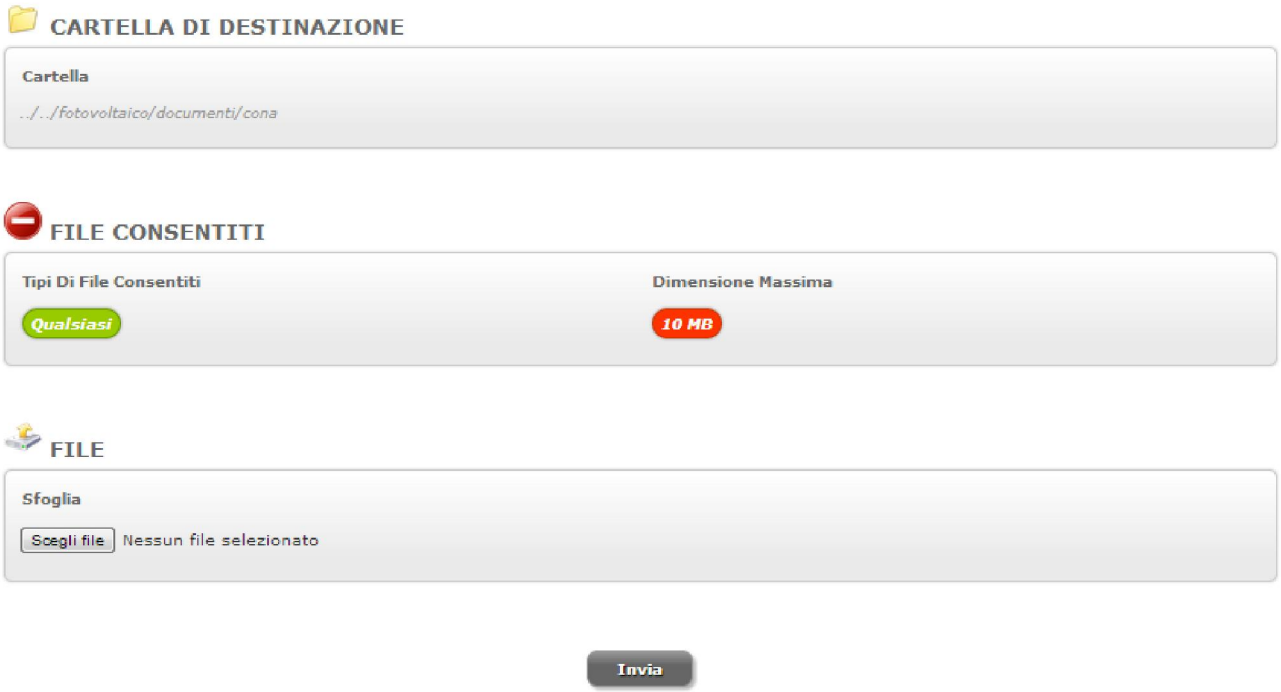


Fig. 54

Lo script utilizzato per l'upload è quello, ormai famoso per ASP, di Baol "Upload con ADO Stream". Anche questo facilmente reperibile su internet.

9. Conclusione

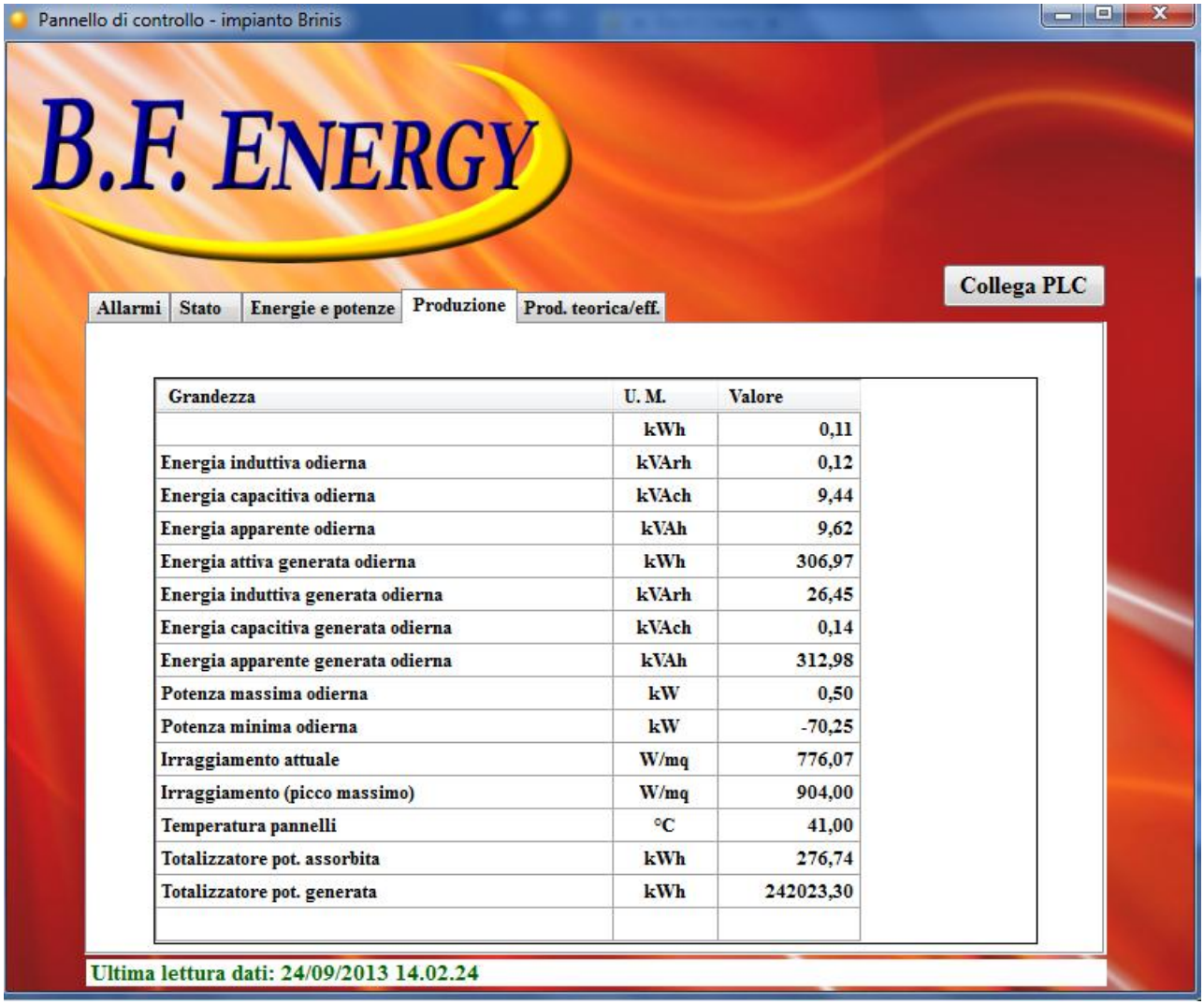
Con questa tesi ho voluto evidenziare le potenzialità delle tecnologie web e dimostrare quanto possa essere semplice creare delle interfacce comode e facilmente modificabili per la gestione di un progetto mediamente complesso come quello trattato.

Per mettere meglio in risalto questo aspetto, fra le diverse pagine e le moltissime righe di codice che compongono il sito, ho voluto descrivere e commentare solo quelle che ho ritenuto più importanti.

Questo progetto ha comunque trattato un impianto fotovoltaico reale e ha reso, con una spesa contenuta, la quotidianità delle operazioni più snelle e semplici.

Da qui, chiaramente, una possibile evoluzione dovrebbe essere quella di adattare l'applicativo alla gestione di più impianti e quindi di più utenti. Ciascuno in grado di visualizzare, da qualsiasi postazione internet, tutte le informazioni di cui ha bisogno.

Rispetto ai software che vengono distribuiti per la visualizzazione degli impianti dei pannelli (come questo in figura 55)



Grandezza	U. M.	Valore
	kWh	0,11
Energia induttiva odierna	kVArh	0,12
Energia capacitiva odierna	kVAch	9,44
Energia apparente odierna	kVAh	9,62
Energia attiva generata odierna	kWh	306,97
Energia induttiva generata odierna	kVArh	26,45
Energia capacitiva generata odierna	kVAch	0,14
Energia apparente generata odierna	kVAh	312,98
Potenza massima odierna	kW	0,50
Potenza minima odierna	kW	-70,25
Irraggiamento attuale	W/mq	776,07
Irraggiamento (picco massimo)	W/mq	904,00
Temperatura pannelli	°C	41,00
Totalizzatore pot. assorbita	kWh	276,74
Totalizzatore pot. generata	kWh	242023,30

Fig. 55

un eventuale aggiornamento viene fatto semplicemente modificando i file del sito che risiedono sul server evitando nuove *release* con le complicate sincronizzazioni direttamente nei pc dei clienti.
Inoltre la visualizzazione di un sito web non richiede alcune installazioni ma semplicemente l'utilizzo di un browser solitamente già preinstallato in tutti i sistemi operativi.
Grazie a questo, indipendentemente dal terminale utilizzato (personal computer, smartphone, tablet), il servizio diventa sempre fruibile.
Un limite del progetto realizzato è rappresentato dalla possibile "pesantezza" del codice con conseguente rallentamento nel processa mento della pagina. Questo, più dei tempi di attesa dell'utente che interroga la pagina, può generare dei black out nel funzionamento del server qualora le richieste contemporanee dovessero essere molte. Noto che aumentare le capacità di calcolo del server comporta dei costi, tale problema è risolvibile limitando il più possibile i conteggi da effettuare.
Come visto nel corso della tesi, il calcolo della produzione avviene interrogando la tabella letture e facendo il relativo conteggio. Il problema nasce però dal fatto che ogni volta che interrogo una lettura, viene rielaborato questo calcolo quando in realtà è un valore ormai fisso (salvo modifiche future della lettura).
Bisognerebbe quindi creare delle procedure che scattino in fase di inserimento della lettura (e di modifica) e che provvedano a salvare il dato della produzione della lettura, del mese, e dell'anno (le più frequentemente utilizzate) in una tabella a parte.
Ecco che la successiva vista della lettura non genererà alcun calcolo ma solamente la visualizzazione di un dato già precedentemente salvato. Tale procedura può essere relegata direttamente al database (*trigger*).
Non mi sono comunque concentrato su questo problema visto che l'utilizzo di questo servizio era destinato ad una sola utenza e che per di più si tratta solamente di un prototipo.

10. Bibliografia

- HTML e CSS / Rachel Andrew, Dan Shafer. - Segrate : Mondadori Informatica, 2007. - XVI, 494 p. : ill. ; 24 cm. (ISBN 9788861140639)
- HTML e CSS / Luca Cattaneo. - Milano [etc.] : McGraw-Hill, [2003]. - VII, 224 p. ; 19 cm. (ISBN 8838643164)
- CSS : guida completa / Gianluca Troiani ; [prefazione di Sofia Postai]. - 2. ed. - Milano : Apogeo, [2008]. - XXIV, 455 p. ; 24 cm. (ISBN 9788850327713)
- ASP : La guida definitiva / Carlo Pelliccia. - [Milano] : Master, 2002. - v. ; 24 cm. (ISBN 8883010442)
- ASP : guida di riferimento / A. Keyton Weissinger. - Milano : Apogeo, 2001!. - XVIII, 483 p. ; 23 cm. (ISBN 8873037704)
- Ajax : guida per lo sviluppatore / Joe Fawcett, Jeremy McPeak, Nicholas C. Zakas. - Milano : Hoepli, [2006]. - XV, 398 p. ; 24 cm. (ISBN 8820337541)
- Ajax : pagine web interattive grazie a Asyncronus JavaScript and XML / Enrico Amedeo. - Milano : Apogeo, [2009]. - XI, 204 p. : ill. ; 20 cm. (ISBN 9788850328970)
- Creare XML web service / Scott Short. - Segrate! : Mondadori informatica, 2002. - XIX, 411 p. ; 24 cm + 1 CD-ROM. (ISBN 8883313828)
- Access 2002 : gestione completa dei database / Nora Hantsch. - Milano : Apogeo, 2002. - 191 p. ; 24 cm. (ISBN 8850320361)
- La guida completa SQL / James R. Groff, Paul N. Weinberg. - 2. ed. - Milano [etc.] : McGraw-Hill, [2003]. - XXIV, 1036 p. ; 24 cm + 1 CD-ROM. (ISBN 883864313X)
- Il fotovoltaico : aspetti operativi e vantaggi economici ed ambientali / Pasquale Salerno, Andrea Sillani. - Roma : Buffetti, [2007]. - VIII, 188 p. ; 22 cm. (ISBN 9788819114286)