Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO
ECONOMICO

Sviluppo di strumenti per l'archiviazione, l'elaborazione e l'analisi di dati inerenti motori elettrici per ambienti industriali

G. Campobello, A. Segreto, S. Serrano,
M-A Segreto

Report RdS/PAR2017/003

SVILUPPO DI STRUMENTI PER L'ARCHIVIAZIONE, L'ELABORAZIONE E L'ANALISI DI DATI INERENTI MOTORI ELETTRICI PER AMBIENTI INDUSTRIALI

G. Campobello[1], A. Segreto[1], S. Serrano[1], Maria-Anna Segreto[2]
[1]Dipartimento di Ingegneria, Università degli Studi di Messina
[2]Centro Ricerche ENEA di Bologna

Settembre 2018

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2017
Area: "Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici"
Progetto: D.3 "Processi e Macchinari Industriali"
Obiettivo: a.1 "Facility per la sperimentazione e verifica di motori elettrici ad alta efficienza"
Responsabile del Progetto: Mauro Bertini

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "*Sviluppo di strumenti per l'archiviazione, l'elaborazione e l'analisi di dati inerenti motori elettrici per ambienti industriali"*
Responsabile scientifico ENEA: Ing. Maria-Anna Segreto
Responsabile scientifico Dipartimento di Ingegneria: Prof. Ing. Giuseppe Campobello

Indice

Sommario

Il presente report tecnico descrive le attività svolte nell'ambito dell'accordo di collaborazione tra ENEA e il Dipartimento di Ingegneria dell'Università di Messina per il PAR 2017 e che hanno avuto come obiettivo la definizione e lo sviluppo di nuovi strumenti di supporto all'archiviazione, elaborazione ed analisi di dati inerenti motori elettrici per ambienti industriali.
In particolare le attività hanno permesso di estendere con nuove funzionalità il portale web dell'ENEA, disponibile al sito http://motorielettrici.enea.it, originariamente realizzato dalla stessa ENEA in collaborazione con il Gruppo di Macchine Elettriche Rotanti di ANIE Energia, con l'obiettivo di raccogliere sul sito prodotti conformi al Regolamento della Commissione (CE) 640/2009 su efficienza energetica e classi di efficienza, oltre di segnalare le future evoluzioni normative sul tema.
Più precisamente, le attività oggetto del presente report hanno riguardato:

- la realizzazione di un database MySQL per l'archiviazione di misure sperimentali e/o verifiche di funzionamento su motori elettrici effettuate dall'ENEA oltre che dei relativi documenti tecnici ("Check Report") ;
- l'estensione del portale web con la definizione e realizzazione di nuovi form, atti a permettere agli operatori autorizzati la visualizzazione, l'upload e il download dei Check Report. In particolare, le form realizzate sono corredate da un sistema di icone a semaforo che consente l'immediata visualizzazione dell'esito delle verifiche di funzionamento;
- la realizzazione di filtri di ricerca, in grado di selezionare un sotto-insieme di motori accomunati da specifiche caratteristiche tecniche al fine ultimo di un loro agevole confronto;
- l'implementazione e integrazione di funzionalità grafiche, in grado di fornire, mediante grafici a barre e a torta, informazioni statistiche sui motori selezionati.

Le funzionalità suddette sono descritte nelle successive Sezioni del presente report.
In particolare:

- nella Sezione 1 verrà descritto il portale web ENEA evidenziando come appariva prima delle attività e come appare adesso, dopo l'integrazione delle nuove funzionalità sviluppate da UniME;
- nella Sezione 2 sarà descritto il database sviluppato per l'archiviazione e la gestione dei Check Report;
- nella Sezione 3 saranno descritte le librerie utilizzate per sviluppare le funzionalità grafiche;
- nella Sezione 4 sarà descritta l'architettura dei codice sorgente;
- infine in Appendice è riportata la procedura di installazione.

1 Descrizione del portale

Il portale web ENEA, disponibile al sito http://motorielettrici.enea.it, è stato realizzato con l'obiettivo di raccogliere sul sito dati inerenti motori elettrici e prodotti conformi al Regolamento della Commissione (CE) 640/2009 su efficienza energetica e classi di efficienza, oltre di segnalare le future evoluzioni normative sul tema.

Figura 1: Portale web ENEA (prima delle attività).

Al momento dell'inizio delle attività il portale appariva come rappresentato in Figura 1 e forniva informazioni testuali sui dati di targa dei motori elettrici, sulle relative classi di efficienza energetica oltre che la possibilità di visualizzare dei documenti, detti Test Report, forniti dalle stesse case costruttrici. Scopo principale dell'attività è stato quello di estendere le funzionalità del portale e il database ad esso connesso al fine di permette l'archiviazione e la gestione anche di informazioni e di documenti tecnici inerenti misure sperimentali e/o verifiche di funzionamento effettuate dall'ENEA (in seguito denominati "Check Report") e di fornire tali informazioni in forma grafica oltre che testuale.

Figura 2: Portale web ENEA (a seguito delle attività).

La Figura 1 mostra il portale web come appare adesso a seguito delle attività. In particolare in Figura 2 sono evidenziate in rosso le funzionalità aggiuntive sviluppate da UniME nell'ambito dell'attività e oggetto del presente report.

Dal confronto fra la Figura 1 e la Figura 2 risultano evidenti le estensioni di seguito elencate.

1. nuovi filtri di ricerca: tali filtri permettono di selezionare motori elettrici aventi caratteristiche elettriche o meccaniche simili al fine ultimo di un loro confronto. In particolare, a seguito dell'estensione e delle modifiche realizzate, è possibile filtrare e selezionare i motori in funzione della tensione di alimentazione, della frequenza nominale, del numero di giri, del $\cos\phi$ e del peso oltre che in termini di produttore, potenza, rendimento e corrente.
2. Nuovi form per la visualizzazione e gestione dei Check Report: in particolare selezionando un motore è possibile ottenere, oltre che informazioni sui dati di targa del motore, anche informazioni sui test di misura (Checks Rating) e i valori delle grandezze fisiche misurate, elettriche (tensione, corrente, potenza, frequenza), meccaniche (numero di giri) e termiche (temperatura carcassa, temperatura aria di raffreddamento, ecc.). Inoltre, nella colonna "Check Report" sono presenti delle icone a semaforo che permettono di ottenere informazioni sull'esito e sullo stato delle verifiche (VERDE = Verifica positiva, ROSSO = Verifica negativa, GIALLO = Verifica in corso, GRIGIO = Verifica non ancora effettuata). Infine, nell'ultima colonna sono visibili gli strumenti per l'inserimento dei dati inerenti i Check Report.

3. Nuove funzionalità per la generazione di grafici statistici: selezionando l'"icona grafici" riportata in alto in ogni colonna è possibile ottenere dei grafici a torta e a barre che forniscono informazioni statistiche sui motori.

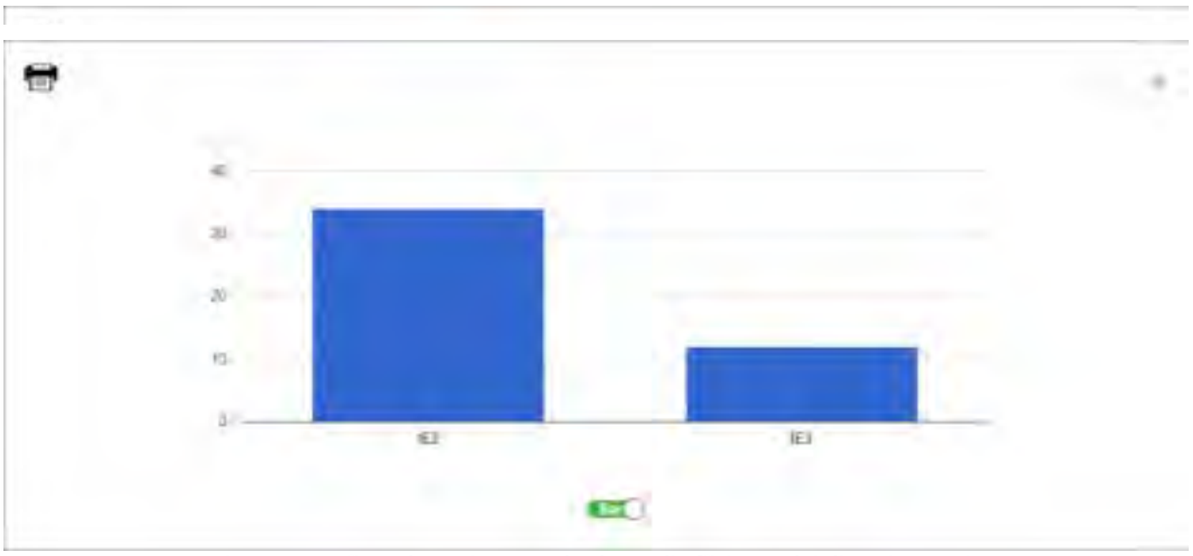


Figura 4: Grafico a barre inerente la classe di efficienza dei diversi motori.

A titolo di esempio in Figura 3 e in Figura 4 sono riportati i grafici ottenuti selezionando l'icona grafici presente in altro nella colonna "Rendimento a Carico". In particolare, i grafici in questione permettono di sapere il numero e la percentuale dei motori che rientra nelle diversi classi di efficienza (IEX).
Analoghe informazioni possono essere estratte per tutte le altre caratteristiche in cui appare l'icona grafici (ad esempio Potenza e Tensione). Infine, selezionando l'icona grafici sulla colonna Check Report è anche possibile determinare la percentuale di motori che hanno o non hanno superato le verifiche (si veda ad esempio la Figura 5).

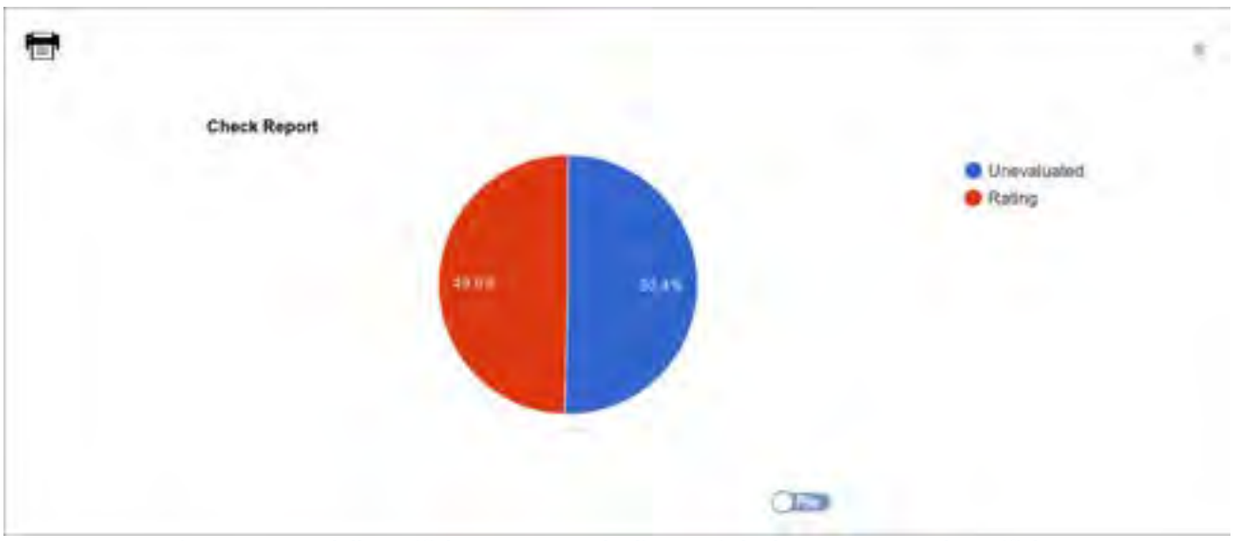


Figura 5: Stato delle verifiche e percentuale di motori che hanno superato le verifiche.

2 Architettura del DataBase

In questa sezione sarà analizzata l'architettura del database alla base del portale ENEA appena descritto nella Sezione 1.

Il database realizzato si appoggia al database precedentemente sviluppato dalla stessa ENEA in collaborazione con il Gruppo Macchine Rotanti di ANIE Energia e reso disponibile da ENEA all'inizio del progetto (si veda la Figura 6).

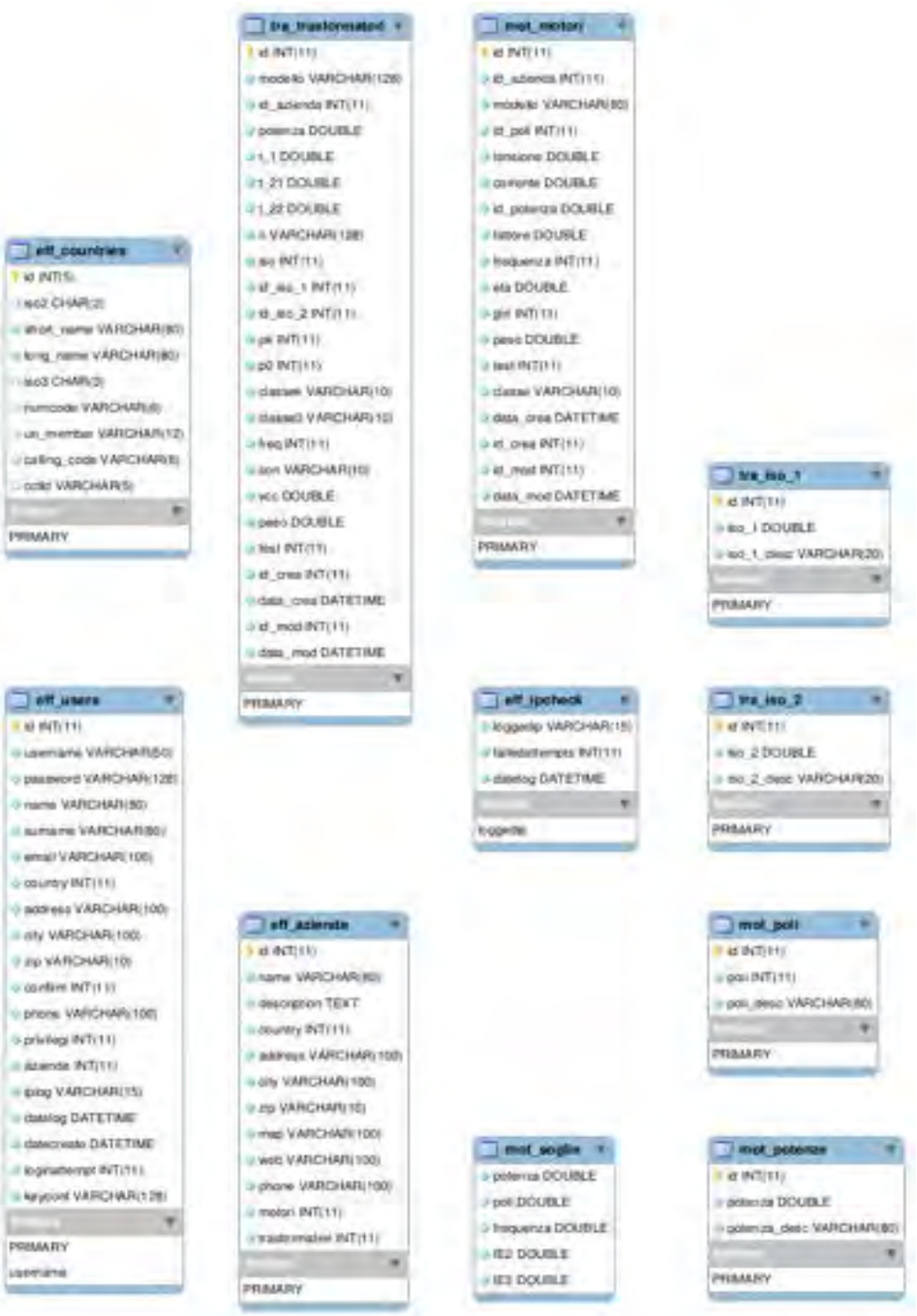


Figura 6: Database ex-ante (fornito da ENEA all'inizio delle attività).

Di seguito viene dapprima descritta la struttura del database originale (ex-ante) e successivamente evidenziate le modifiche apportate e sviluppate all'interno del progetto oggetto del presente report.

2.1 Descrizione del DataBase ex-ante

Il database originale, così come il codice sorgente del portale, è stato fornito da ENEA privo di documentazione. Si ritiene pertanto utile per eventuali ulteriori sviluppi futuri fornirne una breve descrizione in tale sezione.
Le tabelle del database sviluppato da ENEA possono essere logicamente suddivise in tre categorie:

1. tabelle contenenti informazioni sugli utenti e le aziende (`eff_user`, `eff_aziende`, `eff_countries`, `eff_ipcheck`);
2. tabelle contenenti informazioni sui motori elettrici (`mot_motori`, `mot_poli`, `mot_potenze`, `mot_soglie`);
3. tabelle contenenti informazioni sui trasformatori (`tra_trasformatori`, `tra_iso_1`, `tra_iso_2`).

La suddivisione logica si rispecchia nella scelta dei nomi delle tabelle che utilizzano tre diversi prefissi (`eff_`, `mot_`, `tra_`).
Di seguito sarà fornita una descrizione delle diverse tabelle, ad eccezione delle tabelle relative ai trasformatori che non sono state in alcun modo oggetto delle attività svolte presso UniME.

2.1.1 Tabelle comuni (`eff_`)

La tabella `eff_users` contiene le informazioni relative agli utenti registrati nel portale; oltre ad un campo univoco di identificazione (`id`), contiene i dati di autenticazione (`username`, `password`, `iplog`, `datalog`, ecc.), quelli anagrafici (`name`, `surname`, `email`, ecc.) tra cui anche il campo `country` che lega logicamente questa tabella con la tabella `eff_countries`, un campo `azienda` che crea una relazione logica con la tabella `eff_aziende`, il campo `privilegi` che identifica le operazioni concesse all'utente (all'utente comune solo la visualizzazione, all'amministratore di azienda anche la possibilità di inserire e di modificare i dati relativi ad un articolo prodotto dalla sua azienda, all'amministratore globale la possibilità di agire su tutti i dati, sia relativi agli utenti, sia ai trasformatori che ai motori, compresi i dati di check). Il campo `datecreate` indica la data di creazione del record relativo al nuovo utente.
La tabella `eff_aziende` contiene i dati relativi ad una particolare azienda. Ciascuna azienda è identificata da un campo `id`. Oltre ai dati anagrafici (`name`, `description`, `web`, `phone`, ecc.), contiene due contatori, `motori` e `trasformatori`, che tengono traccia, rispettivamente, di quanti motori e i quanti trasformatori sono stati inseriti nel database per ciascuna azienda.
La tabella `eff_countries` contiene le informazioni dei diversi Stati del Mondo e, infine, la tabella `eff_ipcheck` tiene dei diversi ip da cui si loggano gli utenti.

2.1.2 Tabelle relative ai motori (`mot_`)

La tabella `mot_motori` contiene i dati di targa per ciascun motore. È legata logicamente alla tabella `eff_aziende` attraverso il campo `id_azienda`. Il numero di poli, così come la potenza di targa, sono identificate da un id, rispettivamente `id_poli` e `id_potenza`, che lega ciascun record alle corrispondenti righe delle tabelle `eff_poli` e `eff_potenze` che contengono tutti i valori ammissibili per i relativi campi. Opportuni campi (`id_crea` e `data_crea`) identificano l'utente che ha inserito il record e la data di inserimento. In tabella sono anche memorizzati i dati relativi all'ultima modifica (`id_mod` e `data_mod`).
La tabella `mot_soglie`, infine, contiene le soglie per la determinazione delle diverse classi di rendimento a carico.

2.2 Descrizione del DataBase ex-post

Il database suddetto è stato esteso nell'ambito dell'attività al fine di permette l'archiviazione di risultati sperimentali e di documenti inerenti prove di laboratorio e/o verifiche di funzionamento effettuate da ENEA (Check Report).
Si ritiene opportuno precisare che è stata una scelta progettuale quella di limitare al minimo indispensabile il numero e la tipologia di modifiche da apportare al database originale al fine ultimo di minimizzare i rischi legati alla fase di upgrade e di garantire in ogni caso la consistenza dei dati precedentemente inseriti da ENEA
e la piena retro-compatibilità con eventuali altre applicazioni già esistenti che utilizzato lo stesso database.

Lo sviluppo e le scelte progettuali inerenti al database sono state inoltre limitate dal fatto che molte delle tabelle del database originale non risultavano essere relazionate; infatti, nonostante le diverse tabelle siano legate logicamente attraverso dei campi `id`, la struttura fisica realizzata da ENEA non presenta relazioni attraverso chiavi esterne.
In conseguenza di ciò le possibilità di estensione del database da parte di UniME è stata marginale.

Le principali modifiche apportare al database nell'ambito del progetto hanno avuto come obiettivo la possibilità di inserire dati provenienti da misurazioni di laboratorio finalizzare alla validazione dei dati di targa dei motori presenti nel database. A tal fine, opportuni campi, permettono di individuare l'esito del controllo o lo stato di avanzamento delle fasi di test.
Il database è stato esteso anche tenendo conto della necessità, derivante dalle specifiche di progetto, di associare ad ogni stato relativo al processo di verifica una rappresentazione a semaforo che permettesse all'utente che utilizza il portale un immediato riscontro attraverso una rappresentazione iconografica.

La struttura complessiva del database ex-post è mostrata in Figura 7.
È possibile osservare come le nuove tabelle (raffigurate nella prima colonna in Figura 7) sono relazionate logicamente fra di esse attraverso campi identificativi e palesati dalla presenza di legami relazionali realizzati attraverso l'uso di chiavi esterne. Ciò ad eccezione della relazione logica tra la nuova tabella `mot_checkdata` e quella preesistente `mot_motori` che non è stata realizzata attraverso una chiave esterna al fine di mantenere l'architettura preesistente totalmente indipendente. Per le tabelle suddette la relazione è infatti effettuata attraverso il campo `id_motore` della tabella `mot_checkdata` e il campo `id` della tabella `mot_motori`.

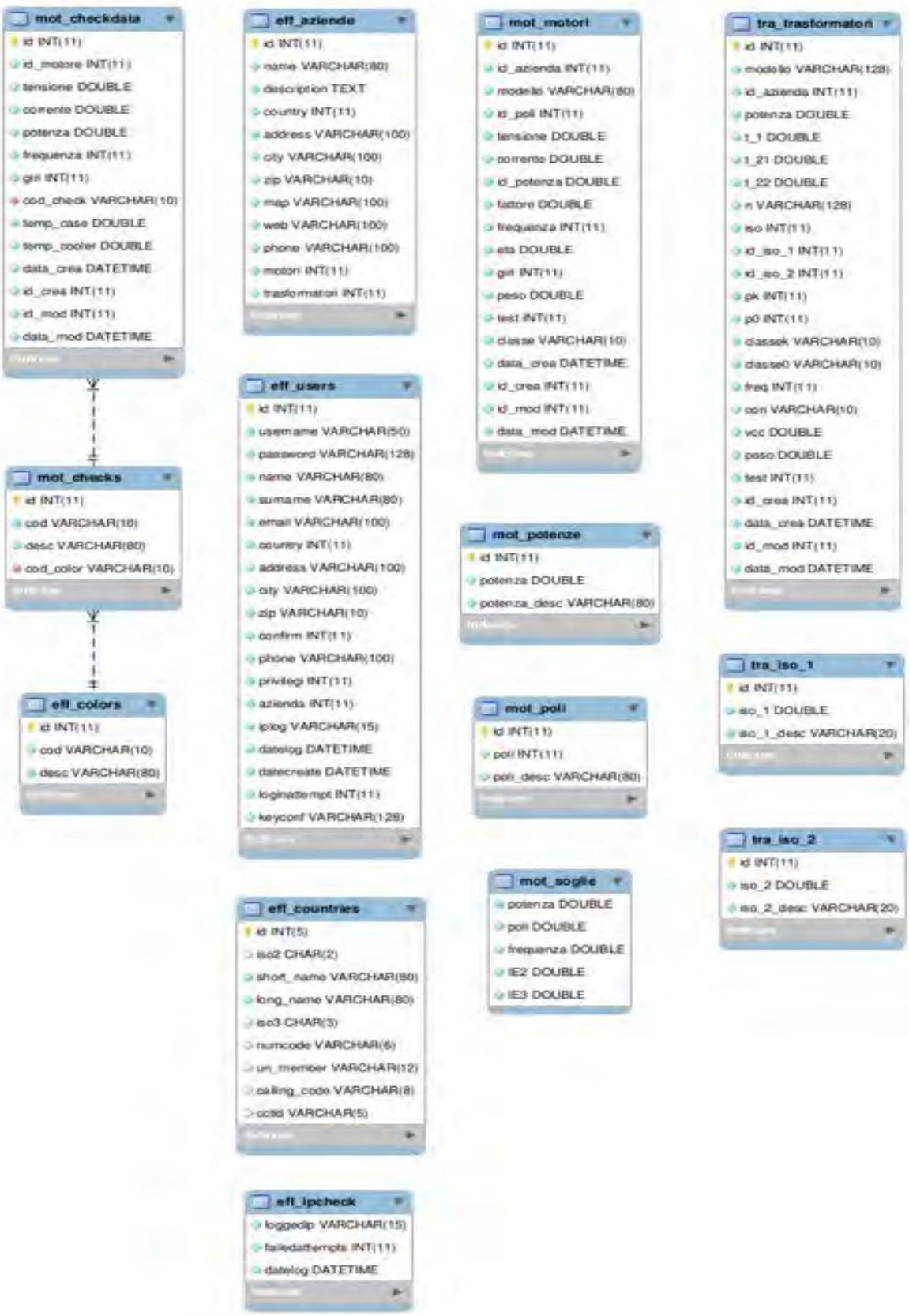


Figura 7: Struttura del DataBase ex-post.

Segue una descrizione della tabelle introdotte.

2.2.1 La tabella mot_checkdata

La tabella `mot_checkdata` contiene, oltre un campo identificativo per ogni record (`id`) e il campo `id_motore` sopra descritto, i risultati delle misurazioni di laboratorio, in particolare:

- tensione alternata a frequenza industriale (`tensione`) ;
- corrente alternata a frequenza industriale (`corrente`) ;
- potenza elettrica attiva (`potenza`);
- frequenza della tensione di alimentazione (`frequenza`);
- velocità di rotazione (`giri`);
- temperatura delle carcasse dei motori (`temp_case`);
- temperatura dell'aria di raffreddamento (`temp_cooler`).

Sono presenti inoltre i campi per indicare utente e data di inserimento e di ultima modifica (`id_crea`, `data_crea`, `id_mod` e `data_mod`).
Un ultimo campo, `cod_check`, è relazionato col campo `cod` della tabella `mot_checks`.

2.2.2 La tabella mot_checks

La tabella `mot_checks` contiene i possibili stati legati alle procedure di controllo. Ciascun record contiene un identificativo auto-incrementale (`id`), un codice univoco (`cod`) e il campo `cod_color` relazionato col campo `cod` della tabella `mot_colors`.

2.2.3 La tabella mot_colors

La tabella `mot_colors` permette di associare un colore ad ogni stato di check. Essa contiene un identificativo auto-incrementale (`id`), un codice univoco (`cod`) e una descrizione alfanumerica (`desc`).

2.2.4 Gestione della documentazione relativa ai motori

Le procedure di gestione dei documenti relativi ai Check Report sono state realizzate in modo da essere conformi alle analoghe procedure utilizzate per la gestione dei Test Report.
In particolare, i file relativi alle operazioni di validazione andranno memorizzati all'interno della cartella `files` del portale e, più precisamente, nella sottocartella `motori`. Per i nomi dei file relativi alla fase di validazione occorre inoltre utilizzare la seguente convenzione:

- check_Report → `[id_motore].check.Check_Report.pdf`;
- altri documenti di check → `[id_motore].checkdoc.[nome_file].pdf`.

Nel momento in cui l'utente visualizza sul portale i dettagli relativi ad un particolare motore, uno script php ricerca tutti i corrispondenti file visualizzando con un'icona verde il Check_Report e con icone azzurre gli altri documenti, come riportato nell'esempio di Figura 8.

Figura 8: Dati Check Report e documenti relativi.

3 Funzionalità grafiche

Dopo un'approfondita analisi sulle diverse tecnologie software e librerie funzionali alla visualizzazione di grafici all'interno di pagine web, si è optato per l'utilizzo delle librerie "Google Charts"[1]. Segue una breve descrizione delle principali caratteristiche funzionali e implementative di tali librerie oltre che alcuni esempi di impiego utili a comprendere il codice sviluppato nell'ambito dell'attività, illustrato nella sezione successiva.

3.1 Introduzione alle librerie "Google Charts"

Le "Google Charts" permettono la visualizzazione di diversi tipi di grafici: da semplici grafici a linee alle più complesse mappe gerarchiche ad albero.
Il modo più comune di utilizzarle è quello di includerle semplicemente come JavaScript all'interno di una pagina web.
Più precisamente, una volta caricata la libreria, per il loro utilizzo occorre:

- elencare i dati da graficare;
- selezionare le opzioni per personalizzare il grafico;
- creare un oggetto grafico con un identificativo a propria scelta;
- creare all'interno della pagina web un " " attraverso il quale visualizzare il grafico.

I grafici saranno così esposti come classi JavaScript e potranno essere ulteriormente personalizzati per venire incontro allo stile del sito web da realizzare.
I grafici ottenuti sono interattivi e, attraverso l'esposizione di diversi eventi, possono essere interconnessi per realizzare pannelli di strumenti o altre forme di integrazione con la pagina web da realizzare.
Per poterle utilizzare non occorre che gli utenti scarichino plugins o qualsiasi altro tipo di software; gli utenti possono infatti visualizzare i grafici a partire da un semplice browser.
Le "Google Charts" utilizzano infatti per la loro esecuzione una tecnologia integrata "HTML5/SVG" e forniscono una compatibilità multi-browser (includendo anche VML per le versioni più vecchie di Internet Explorer) oltre che una portabilità multi-piattaforma anche per dispositivi mobili come iPhones e iPads basati sul sistema operativo Android.
Per tutte le tipologie di grafici si utilizza la classe "DataTable" per inserire i dati da graficare. Questo permette agevolmente di poter cambiare la visualizzazione da un tipo di grafico ad un altro. La classe "DataTable" fornisce anche metodi per ordinare, modificare, e filtrare i dati. Inoltre i dati possono essere inseriti direttamente dalla pagina web (in maniera statica), utilizzando un database o qualunque altra sorgente che supporti il protocollo Chart Tools Datasource.
Il protocollo include un linguaggio molto simile alle query SQL ed è implementato anche in Google Spreadsheets, Google Fusion Tables e da fornitori terzi come SalesForce.
In linea di principio si può implementare il protocollo nel proprio sito e divenire fornitori di dati per altri servizi.

3.2 Integrazione ed esempi di utilizzo delle librerie "Google Charts"

Per integrare il codice all'interno di una pagina web, la prima operazione da fare è quella di inserire il "caricatore" delle stesse librerie.
Questo può essere fatto inserendo il "tag" *src="https://www.gstatic.com/charts/loader.js"* all'interno dell'intestazione o del corpo del documento oppure può essere inserito dinamicamente, mentre il documento viene caricato o dopo che il caricamento è completato.
Nell'esempio seguente il "tag" è stato inserito all'interno di una sezione di intestazione

```
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
```

Una volta che il "caricatore" è stato inserito è necessario utilizzarlo per caricare le librerie utilizzando l'apposito comando. Anche questa operazione può essere effettuata indifferentemente nell'intestazione o nel corpo della pagina web oppure dinamicamente mentre il documento viene caricato o dopo che il caricamento è completato.

[1] https://developers.google.com/chart/

```
<script type="text/javascript">
    ...
  google.charts.load('current', {'packages':['corechart']});
    ...
</script>
```

Alla funzione devono essere passati due parametri: il primo indica la versione, il secondo è un oggetto che specifica la tipologia di grafici che si vorranno utilizzare, la lingua da utilizzare ed eventualmente una funzione di callback per indicare il completamento del caricamento delle librerie. Quest'ultima è necessaria perché il completamento del caricamento della libreria potrebbe avvenire dopo che la pagina è stata caricata e quindi, in questo caso, occorre attendere prima di richiedere la visualizzazione dei grafici. Tale funzione di callback può comunque essere specificata successivamente attraverso l'istruzione "setOnLoadCallback", come si evince dall'esempio seguente:

```
<script type="text/javascript">
    ...
  google.charts.load('current', {'packages':['corechart']});
  google.charts.setOnLoadCallback(drawChart);

    ...
</script>
```

Nell'esempio suddetto viene caricata la versione "corrente" delle librerie, il package con i grafici di base e impostata la funzione "drawChart" come funzione di callback al termine del caricamento della libreria.
All'interno della funzione di callback può essere inizializzato l'oggetto di tipo "DataTable" che conterrà i dati da graficare. Nello specifico l'oggetto sarà una tabella bidimensionale con righe e colonne. Ogni colonna avrà uno specifico tipo, un identificativo opzionale e un'etichetta anch'essa opzionale. I dati di ogni entry dovranno essere inseriti nelle diverse righe.
Segue un esempio di creazione e preparazione della tabella dei dati all'interno della funzione di callback:

```
function drawChart() {

  var data = new google.visualization.DataTable();
  data.addColumn('string', '');
  data.addColumn('number', '');
  data.addRows([
    ['Casa', 14000],
    ['Trasporti', 5760],
    ['Abbigliamento', 3840],
    ['Alimentari', 9600],
    ['Altro', 4800]
  ]);
```

Nello specifico le due chiamate al metodo "addColumn" permettono di inserire due tipologie di dati rispettivamente di tipo stringa e di tipo numerico. Utilizzando la chiamata al metodo "addRows" si inseriscono i valori su ciascuna riga per entrambe le colonne.
Sempre all'interno della funzione di callback sarà possibile personalizzare il grafico specificando ad esempio la posizione della legenda, il titolo, i colori e le dimensioni del grafico.

```
  ....

  var options = {'title':'Distribuzione delle Spese',
                 'width':600,
                 'height':600};

  .....
```

Sempre all'interno della stessa funzione di callback sarà quindi necessario creare un'istanza della classe di grafico che si vuole utilizzare e quindi richiamare il relativo metodo "draw" che permette di visualizzare il grafico nella corrispondente pagina web.

La funzione “costruttore” dell’oggetto “grafico” ha in ingresso un unico parametro che specifica l’elemento “DOM” dove il grafico verrà visualizzato.
Il codice HTML della pagina web dovrà contenere un oggetto “ ” che conterrà la visualizzazione del grafico all’interno della sezione “<body>”:

```
<html>
...
  <body>
    <div id="chart_div"></div>
  </body>
...
</html>
```

l’identificativo di tale oggetto (nel caso specifico “chart_div”) sarà poi utilizzato come parametro da passare al costruttore dell’oggetto grafico all’interno del codice JavaScript:

```
function drawChart() {

  ....

  var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
  chart.draw(data, options);

  ....

}
```

Il metodo “draw” dell’oggetto potrà quindi essere utilizzato per visualizzare il grafico utilizzando i dati inseriti nell’oggetto passato come primo parametro di tipo “DataTable” (in questo caso “data”) e l’oggetto che specifica le opzioni del grafico.

Nella seguente figura è possibile osservare il grafico di tipo “a torta” generato a partire dal codice appena descritto:

3.3 Modalità di utilizzo delle “Google Charts” nell'ambito del progetto

Nell’implementazione specifica inerente il progetto oggetto del presente report, i dati da graficare sono letti dal database e successivamente copiati all’interno di un vettore PHP passato al JavaScript utilizzando la notazione JSON. E’ stato necessario quindi adattare il contenuto di tali dati in maniera tale da renderli compatibili con l’oggetto “DataTable” per poi procedere alla visualizzazione del grafico. Utilizzando questa metodologia è stato possibile realizzare la visualizzazione dei grafici relativi a diversi campi del database senza la necessità di scrivere del codice specifico per ogni singolo campo e anche tenere facilmente in considerazione gli eventuali filtri applicati dall’utente.
Nello specifico la prima modifica è stata inserita all’interno del file sorgente “PHP” `motori.php` che essenzialmente interagisce con il DB mysql per creare gli oggetti contenenti le variabili da visualizzare. A

seconda dell'azione POST richiesta dalla pagina web si occupa di effettuare diverse operazioni che sono selezionabili tramite la variabile "act2". Questa può assumere i valori, le cui funzionalità sono di seguito brevemente riassunte::

- "report", "doc": vengono preparati le variabili per permettere la visualizzazione di report o di documenti associati al motore leggendoli dal filesystem ed eseguiti i comandi per la visualizzazione stessa;
- "report_tmp", "doc_tmp": vengono preparate le variabili per la visualizzazione di report o di documenti temporanei associati al motore leggendoli dal filesystem ed eseguiti i comandi per la visualizzazione stessa;
- "report_del", "doc_del": vengono preparate le variabili per permettere la cancellazione di report o di documenti associati al motore ed eseguita la cancellazione dal filesystem;
- "report_up", "doc_up": vengono preparate le variabili per permettere il caricamento di report o di documenti associati al motore sul server ed effettuato il caricamento nelle apposite sezioni del filesystem;
- "class": vengono preparate le variabili per poter associare una classe al motore e quindi aggiornare i relativi campi del DB;
- "public_report": vengono preparate le variabili per rendere pubblico un report associato al motore e quindi aggiornati i relativi campi del DB;
- "delete": vengono preparate le variabili per poter cancellare uno specifico motore dal database ed effettuata la cancellazione dal DB;
- "insert": vengono preparate le variabili per inserire i campi di un nuovo motore all'interno del DB ed effettuare le relative operazioni sul DB stesso;
- "checkmod": questa sezione è stata inserita a seguito della nuova funzionalità legata all'operazione di "check" del motore stesso; vengono preparate le variabili per inserire o modificare i diversi campi di "check" ed aggiornato il DB;
- "list": è la sezione che permette la visualizzazione dell'elenco dei motori la quale è oggetto delle modifiche necessarie per la visualizzazione dei grafici. All'interno di questa sezione sono preparate le variabili e lette le informazioni dal DB per poter visualizzare correttamente la lista dei motori accessibili all'utente che ha effettuato il login; questa sezione è stata inoltre pesantemente modificata anche per poter visualizzare le nuove informazioni legate al check del motore;
- "stats": in questa sezione vengono preparate le variabili per contenere informazioni statistiche su alcuni parametri dei motori inseriti nel DB quali i diversi valori del campo potenza, del campo poli, del campo tensione e del campo eta;
- "new": vengono preparate le variabili per inserire un nuovo motore all'interno del DB ed effettuato l'inserimento stesso nella relativa tabella;
- "mod": vengono preparate le variabili per modificare il contenuto dei campi di uno specifico motore ed effettuate le modifiche sul DB;
- "ver": questa sezione è stata inserita per permettere la nuova funzionalità legata all'operazione di "check" dei motori da parte degli amministratori. Essenzialmente vengono preparate le variabili per inserire i nuovi campi legati alla tabella "checkdata" che conterrà i campi associati al check. Dato che i motori già inseriti nel DB potrebbero non avere il relativo entry associato nella tabella "checkdata", se il record non viene trovato sarà opportunamente creato.

Per la funzionalità specifica alla creazione dei grafici si è comunque modificata esclusivamente la sezione list. All'interno di tale sezione è stato necessario creare una nuova variabile da passare al JavaScript che permettesse la visualizzazione dei grafici a partire dai dati contenuti nel DB e filtrati dall'utente. E' stata inserita una nuova variabile di intestazione (header) denominata isgraph che contiene il valore "1" quando il rispettivo campo può essere graficato. Inoltre sono state create le variabili stat_potenza, stat_poli, stat_tensione, e stat_eta che conterranno i valori da visualizzare nei rispettivi grafici. Le variabili saranno degli oggetti nella forma "campo, valore" contenenti il numero di entry (valore) che appartengono alle diverse possibili classi (campo) a partire dai record filtrati dall'utente nell'attuale visualizzazione.

Per la visualizzazione vera e propria dei grafici si è utilizzato il codice JavaScript memorizzato all'interno del file script.js nella cartella public/js.

Per permettere la visualizzazione dell'icona che rende possibile all'utente attraverso l'azione di "click" di visualizzare il relativo grafico è stata modificata la funzione "`createList`". In pratica leggendo il valore della variabile `header` che abbiamo denominato `isgraph`, se il valore è "vero", viene inserito un item contenente anche l'icona per la visualizzazione del grafico. Segue uno stralcio del codice inserito:

```
if(resp.header[i].isgraph)
{
        var item=$('<div data-key="'+resp.header[i].key+'" data-dir="'+resp.head
er[i].ord+'"><div><a id ="id_'+(i+1)+'" class="icon white stats" ></a><br/><div>
'+resp.header[i].label+'</div></div></div>').appendTo(header).addClass('col'+(i+
1));
        $("#id_"+(i+1)).click(function(e){e.stopPropagation();document.getElemen
tById('chart_Modal').style.display="block";document.getElementsByClassName("moda
l-close")[0].onclick = function() { document.getElementById('chart_Modal').style
.display = "none"; }; drawChart(this, JSON.stringify(resp));});
}
else
...
```

si può notare che al "click" sull'icona sarà visualizzata la finestra di tipo modal "`chart_Modal`" e richiamata la funzione "`drawChart`".
La prima parte della funzione "`drawChart`" crea una variabile di tipo `array`, denominata `arrayData` che conterrà i dati da visualizzare a seconda dell'icona sui cui è stato fatto click (alla specifica colonna dati da graficare):

```
function drawChart(id_stat, resp_str) {
        if(id_stat)
        {
                index_col = parseInt(replaceAll(id_stat.id, "id_", ""))-1;
                var resp = JSON.parse(resp_str);
                //alert(JSON.stringify(resp));
                if(resp.header[index_col].key=="potenza"){
                        arrayData = JSON.stringify(resp.stat_potenza);
                        arrayData = replaceAll(arrayData, "\"potenza\":", "");
                }
                else if(resp.header[index_col].key=="poli"){
                        arrayData = JSON.stringify(resp.stat_poli);
                        arrayData = replaceAll(arrayData, "\"poli\":", "");
                }
                else if(resp.header[index_col].key=="tensione"){
                        arrayData = JSON.stringify(resp.stat_tensione);
                        arrayData = replaceAll(arrayData, "\"tensione\":", "");
                }
                else if(resp.header[index_col].key=="eta"){
                        arrayData = JSON.stringify(resp.stat_eta);
                        arrayData = replaceAll(arrayData, "\"classe\":", "");
                }
                else if(resp.header[index_col].key=="cod_check"){
                        arrayData = JSON.stringify(resp.stat_check);
                        arrayData = replaceAll(arrayData, "\"stato\":", "");
                        arrayData = replaceAll(arrayData, "{null", "{\"Unevaluat
ed\"");
                }
...
...
```

Successivamente i dati devono essere opportunamente formattati per rispettare le specifiche richieste da GoogleCharts e, quindi tramite la funzione arrayToDataTable convertiti nel tipo `DataTable`:

```
function drawChart(id_stat, resp_str) {
        if(id_stat)
        {
                ...
                ...
                arrayData = replaceAll(arrayData, "{", "[");
                arrayData = replaceAll(arrayData, "}", "]");
                arrayData = replaceAll(arrayData, "\"cc\":\"", "");
                arrayData = replaceAll(arrayData, "\"]", "]").replace("[[", "[[\
""+resp.header[index_col].label.replace("<br />", " ")+"\",\"Num\"],[");
                var dataArray = JSON.parse(arrayData);
                var data = google.visualization.arrayToDataTable(dataArray);
                ...
                ...
```

Saranno quindi impostati i valori delle opzioni del grafico da visualizzare impostando la variabile `options` e richiamate le funzioni per la visualizzazione del grafico:

```
function drawChart(id_stat, resp_str) {
        if(id_stat)
        {
                ...
                ...

                var options = {
                        'title': resp.header[index_col].label.replace("<br />",
" "),
                        'height': 400
                };
                chart = new google.visualization.PieChart(document.getElementByI
d('chart_div'));
                var chart_link = document.getElementById('chart_print');
                google.visualization.events.addListener(chart, 'ready', function
 () {chart_link.innerHTML = '<a id="chart_png" href="' + chart.getImageURI() + '
" target="_blank"></a>';});
                chart.draw(data, options);
        }
}
```

L'inserimento dell'evento `addListener` permetterà di scambiare la modalità di visualizzazione del grafico da "a torta" a "istogramma" e viceversa.
Tale funzionalità è introdotta attraverso l'inserimento della funzione `redrawChart()` della quale elenchiamo di seguito il codice

```
function redrawChart() {
        var dataArray = JSON.parse(arrayData);
        var data = google.visualization.arrayToDataTable(dataArray);
        var options = {
                'title': title_chart,
                'height': 400,
                'legend': document.getElementById('Conoffswitch').checked?'none'
:''
        };
        if (document.getElementById('Conoffswitch').checked)
                chart = new google.visualization.ColumnChart(document.getElement
ById('chart_div'));
        else
                chart = new google.visualization.PieChart(document.getElementByI
d('chart_div'));
        var chart_link = document.getElementById('chart_print');
        google.visualization.events.addListener(chart, 'ready', function () {
                chart_link.innerHTML = '<a id="chart_png" href="' + chart.getIma
geURI() + '" target="_blank"></a>';
        });
        chart.draw(data, options);
}
```

Si osservi che a seconda dello stato della variabile `Conoffswitch` sarà visualizzato il grafico “a torta” o a “istogramma”.

Infine gli elementi DOM per contenere i grafici sono stati inseriti all’interno del codice PHP residente nel file `motori_template.php`, inserendo il codice riportato di seguito nella specifica sezione:

```
<?php
...
echo '<div id="chart_print"></div>';
echo '<a class="modal-close">&times;</a>';
echo '<div id="chart_div" class="modal-body"></div>';
...
?>
```

4 Architettura dei codici sorgente

In questa sezione è descritta l'architettura del portale a livello di codici sorgente.
I codici sorgente realizzati sono articolati in cinque cartelle, illustrate in Figura 9 e di seguito descritte. Le cartelle rispecchiano una architettura modulare atta a separare la parte grafica da quella funzionale.

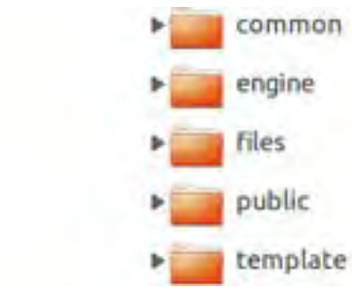


Figura 9: Cartelle principali del progetto.

Di seguito sarà descritto il contenuto delle singole cartelle soffermandosi sulle estensioni di interesse per l'attività oggetto del presente report.

4.1 *La cartella `common`*

La cartella `common` contiene i cinque script mostrati in Figura 10.

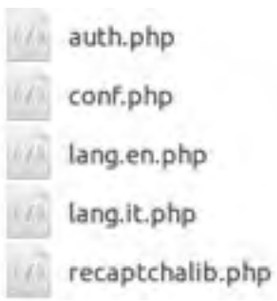


Figura 10: La cartella `common`.

In particolare:

- `auth.php` → modulo per l'autenticazione dei vari utenti;
- `conf.php` → modulo contenente i dati di configurazione del database e delle lingue supportate. Sono anche presenti funzioni utilizzate da altri moduli;
- `lang.en.php` → modulo contenente le specifiche per la lingua inglese;
- `lang.it.php` → modulo contenente le specifiche per la lingua italiana;
- `recaptchalib.php` → libreria per la gestione dei recapcha.

4.2 *La cartella `engine`*

La cartella `engine` contiene i cinque moduli mostrati in Figura 11 e che costituiscono il core del progetto, in quanto gestiscono le diverse entità del progetto.

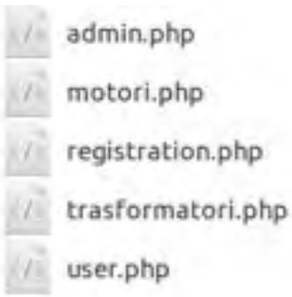


Figura 11: La cartella `engine`.

In particolare:

- `admin.php` → modulo dedicato alla gestione del portale da parte degli amministratori;

- `motori.php` → modulo dedicato alla gestione dei motori.
  Nell'ambito delle attività tale modulo è stato esteso con tutte le funzionalità necessarie per la gestione dei Check Report (inserimento, visualizzazione, modifica e cancellazione) e dei dati relativi. Il modulo si occupa anche della ricerca e della gestione dei file relativi ai diversi motori, sia quelli della fase di test che quelli della fase di check;
- `registration.php` → modulo per la registrazione di nuovi utenti;
- `trasformatori.php` → svolge funzioni analoghe al modulo `motori.php` ma riferte ai trasformatori;
- `user.php` → modulo dedicato alla gestione degli utenti (inserimento, visualizzazione, modifica e cancellazione).

4.3 La cartella `files`

La cartella `files` contiene documenti di test e di check distribuiti in tre sottocartelle, mostrate in Figura 12.

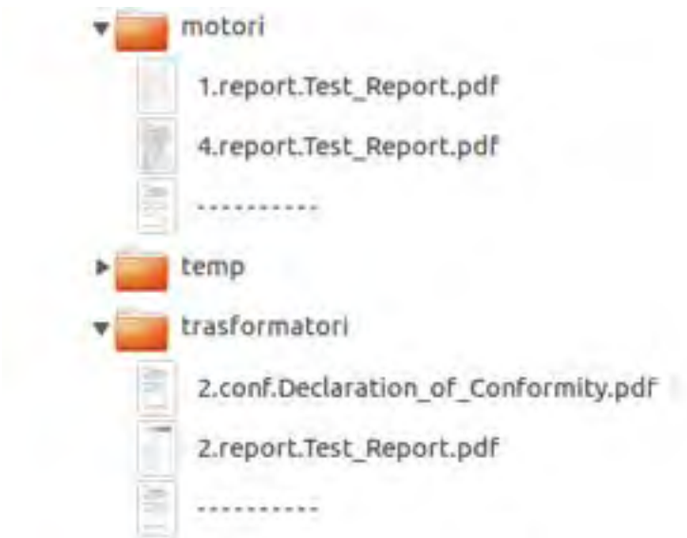


Figura 12: La cartella `files`.

In particolare:

- `motori` → la cartella contiene i documenti relativi alla fase di test e di check dei motori elettrici;
- `trasformatoti` → la cartella contiene i documenti relativi alla fase di test dei trasformatori;
- `temp` → la cartella contiene file temporanei creati durante il download dei file.

4.4 La cartella `public`

La cartella `public` e le relative sottocartelle contengono tutte le parti del portale direttamente accessibili dall'esterno.

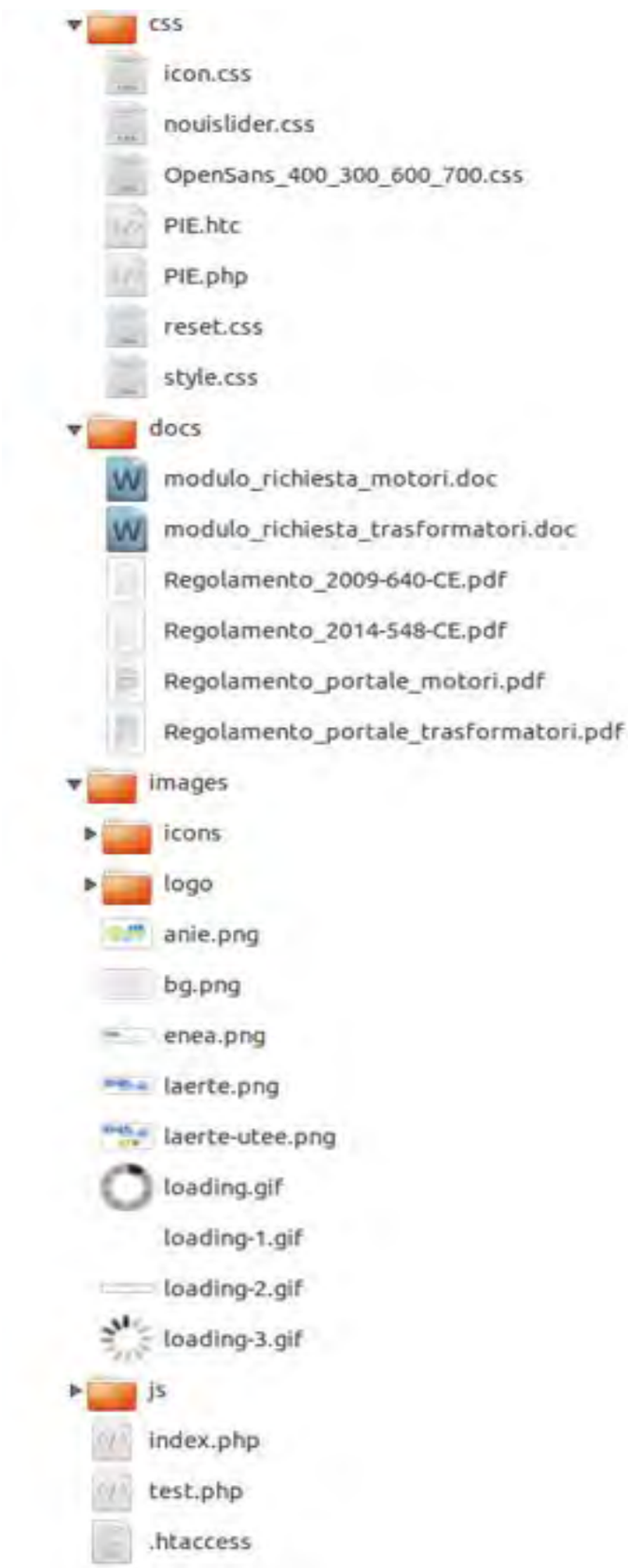


Figura 13: La cartella `public`.

In paticolare:

- `css` → contiene i fogli di stile utilizzati per fornire al portale l'aspetto grafico desiderato;

- `docs` → contiene i documenti liberamente scaricabili dalla home page del portale;
- `images` → la cartella e le relative sottocartelle contengono tutte le immagini utilizzate dai fogli di stile o direttamente dal codice html generato;
- `js` → la cartella contiene le diverse librerie JavaScript e il file `script.js` che si occupa dell'aggiornamento dinamico delle pagine. Il modulo è stato esteso per la gestione delle nuove funzionalità del portale. In particolare, contiene le funzioni per il tracciamento dei grafici e il codice per la generazione runtime delle tabelle.

4.5 *La cartella `template`*

La cartella `template` contiene tutti i template utilizzati dalle diverse pagine del portale (Figura 14).

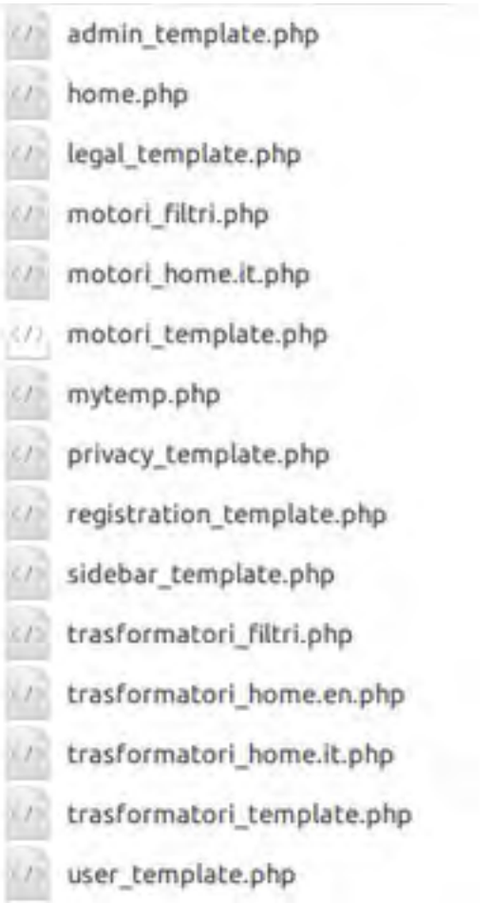


Figura 14: La cartella `template`.

In particolare, il template `motori_filtri.php` contiene i codici per la gestione dei filtri sviluppati nell'ambito dell'attività e il template `motori_template.php` contiene il codice che permette la visualizzazione dei dati di check e la form per l'inserimento e la modifica degli stessi.
In `mytemp.php`, inoltre, è richiamata la libreria Google Chart per la gestione dei grafici.

Appendice

Procedura di installazione

Preventivamente alla installazione dei codici occorre verificare che mysql e php siano installati sulla macchina che ospiterà il portale e che le relative versioni siano almeno pari a quelle minime di seguito indicate:

- mysql >= 5.4
- php >= 5

A tal fine è sufficiente digitare i comandi

```
mysql –version
php --version
```

Successivamente occorre copiare l'intero progetto nella cartella desiderata (es. `/var/www/enea/macchine_elettriche/`) avendo cura di specificare il nome della cartella nei file di configurazione di Apache. I file di configurazione di Apache hanno estensione .conf e nel caso di macchine Unix/Linux si trovano tipicamente nella cartella `/etc/apache2/` o, nel caso di Virtual Host, nella cartella `/etc/apache2/sites-available/`.
Si riporta di seguito un possibile file di configurazione fornito con i codici sorgente (`www.motorielettrici.enea.it.conf`):

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName motorielettrici.enea.it
        ServerAlias www.motorielettrici.unime.it
        ServerAlias macchineelettriche.unime.it
        ServerAlias macchineelettriche.enea.it
        ServerAlias trasformatori.unime.it
        ServerAlias trasformatori.unime.it

        DocumentRoot /var/www/enea/macchine_elettriche/public

        <Directory /var/www/enea/macchine_elettriche>
                Options FollowSymLinks
                AllowOverride All
                Require all granted
        </Directory>

        LogLevel warn
        #ErrorLog ${APACHE_LOG_DIR}/error.log
        #CustomLog ${APACHE_LOG_DIR}/access.log combined
        ErrorLog /var/www/enea/error.log
        CustomLog /var/www/enea/access.log combined
</VirtualHost>
```

Caricato il file di configurazione nella cartella `/etc/apache2/sites-available/` è necessario caricare il modulo rewrite di apache:

```
sudo a2enmod rewrite
```

e riavviare il server Apache:

```
sudo service apache2 restart
```

Infine per l'aggiornamento del database `motori`, occorre eseguire il seguente comando

```
mysql -u[USER] -p[PASSWORD] -Dmotori < motori_extension.sql
```

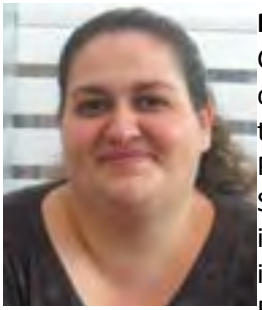
dove `[USER]` e `[PASSWORD]` vanno sostituiti con le credenziali di accesso al database e `motori_extension.sql` è un file fornito con i codici sorgente contenente le seguenti query:

```
CREATE TABLE IF NOT EXISTS `mot_checkdata` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `id_motore` INT(11) NOT NULL UNIQUE,
  `tensione` DOUBLE NOT NULL,
  `corrente` DOUBLE NOT NULL,
  `potenza` DOUBLE NOT NULL;
  `frequenza` INT(11) NOT NULL,
  `giri` INT(11) NOT NULL,
  `cod_check` VARCHAR(10) NOT NULL,
  `temp_case` DOUBLE NOT NULL,
  `temp_cooler` DOUBLE NOT NULL,
  `data_crea` DATETIME NOT NULL,
  `id_crea` INT(11) NOT NULL,
  `id_mod` INT(11) NOT NULL,
  `data_mod` DATETIME NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`cod_check`) REFERENCES `mot_checks`(cod))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `eff_colors` (`cod`, `desc`) VALUES ('GRE', 'Green');
INSERT INTO `eff_colors` (`cod`, `desc`) VALUES ('RED', 'Red');
INSERT INTO `eff_colors` (`cod`, `desc`) VALUES ('YEL', 'Yellow');
INSERT INTO `eff_colors` (`cod`, `desc`) VALUES ('GRA', 'Gray');

INSERT INTO `mot_checks` (`cod`, `desc`, `cod_color`) VALUES ('UN',
'Unevaluated', 'GRA');
INSERT INTO `mot_checks` (`cod`, `desc`, `cod_color`) VALUES ('OK',
'Approved', 'GRE');
INSERT INTO `mot_checks` (`cod`, `desc`, `cod_color`) VALUES ('ERR',
'Rejected', 'RED');
INSERT INTO `mot_checks` (`cod`, `desc`, `cod_color`) VALUES ('RAT',
'Rating', 'YEL');
INSERT INTO `mot_checks` (`cod`, `desc`, `cod_color`) VALUES ('INT',
'Integrations', 'YEL');
```

Giuseppe Campobello nasce a Messina, Italia, nel 1975.
Riceve la Laurea in Ingegneria Elettronica (summa cum laude) e il Dottorato di Ricerca (Ph.D.) in "Tecnologie Avanzate per l'Ingegneria dell'Informazione" presso l'Università degli Studi di Messina, rispettivamente nel 2000 e nel 2004.
Dal 2004 al 2006 è stato titolare di contratti finalizzati ad attività di ricerca e professore a contratto della Facoltà di Ingegneria e della Facoltà di Scienze MM.FF.NN. dell'Università di Messina.
Nel 2006 è stato altresì assegnista di ricerca presso il Dipartimento di Matematica e Informatica dell'Università di Catania.
Nel dicembre del 2006 vince il concorso per Ricercatore Universitario del settore ING-INF/03 (Telecomunicazioni) presso l'Università degli Studi di Messina.
Attualmente afferisce al Dipartimento di Ingegneria dell'Università di Messina dove è Ricercatore Confermato a tempo pieno oltre che Professore Aggregato del settore Telecomunicazioni e responsabile del laboratorio di Comunicazioni Wireless.
L'attività di ricerca, svolta sia in ambito universitario che in collaborazione con aziende ed enti di ricerca, si inquadra principalmente nell'ambito delle reti di telecomunicazioni, dell'elaborazione numerica dei segnali e dell'elettronica digitale applicata alle telecomunicazioni. In particolare l'attività di ricerca più recente è focalizzata sulle reti di sensori wireless e sulle tecniche di compressione.
È autore di oltre trenta articoli scientifici apparsi su riviste internazionali o presentati a conferenze internazionali e revisore di diverse riviste internazionali della IEEE e della Elsevier.
È inoltre membro del Consiglio Scientifico del Gruppo Telecomunicazioni e Tecnologie dell'Informazione (GTTI) e del Microwave Engineering Center for Space Applications (MECSA).

Antonino Segreto nasce a Messina, Italia nel 1979.
Consegue la Laurea quinquennale in Ingegneria Elettronica nel 2011 e il Dottorato di Ricerca (Ph.D) in "Tecnologie avanzate per l'Optoelettronica, la Fotonica e Modellizzazione Elettromagnetica" nel 2015, entrambe presso l'Università degli Studi di Messina.
Dal 2015 ad oggi è collaboratore di ricerca presso il Dipartimento di Ingegneria dell'Università degli Studi di Messina.
L'attività di ricerca riguarda principalmente le reti di telecomunicazioni, l'elaborazione dei segnali e le reti di sensori wireless.

Salvatore Serrano è nato a Catania (Italia) nel 1972. Ha conseguito rispettivamente la laurea quinquennale in Ingegneria Informatica nel 1999 e il dottorato di ricerca in Ingegneria Informatica e delle Telecomunicazioni nel 2003 presso l'Università degli Studi di Catania. Dal 2005 al 2007 ha lavorato come assegnista di ricerca presso il Dipartimento di Ingegneria Informatica e delle Telecomunicazioni dell'Università degli Studi di Catania occupandosi di riconoscimento dello stato emotivo degli individui attraverso l'analisi del segnale vocale inserito in un'attività di ricerca supportata da Telecom Italia Mobile (TIM).
Attualmente è ricercatore a tempo indeterminato e professore aggregato del settore telecomunicazioni presso il Dipartimento di Ingegneria dell'Università degli Studi di Messina dove è anche responsabile del laboratorio "Telecomunicazioni". L'attività di ricerca riguarda l'area dell'elaborazione del segnale (codifica e riconoscimento della voce, elaborazione di segnali biomedicali, elaborazione del segnale per le telecomunicazioni) e le reti di telecomunicazioni (wireless mesh network, voice over IP e wireless sensor network). È autore di oltre quaranta articoli scientifici apparsi su riviste internazionali o presentati a conferenze internazionali, revisore di diverse riviste internazionali e fa parte dell'Editorial Board della rivista International Journal of Distributed Sensor Networks (IJDSN).



Maria-Anna Segreto nasce a Messina nel 1975.
Consegue la laurea quinquennale in Ingegneria Civile-Edile nel 2005 con una tesi dal titolo "Sviluppo delle tecnologie per componenti edilizi innovativi opachi e trasparenti. Progettazione energetica di una scuola d'arte nell'area del Monte di Pietà a Messina." Dal 2005 ricercatrice presso ENEA, attualmente Responsabile Scientifico del laboratorio di Ricerca Industriale LAERTE, si occupa principalmente in studi in ambito di efficientamento energetico in ambito civile e nei processi industriali. Membro della Commissione Ambiente dei Dottori Commercialisti di Bologna, membro del Comitato Tecnico Scientifico per la redazione del Piano Energetico Regionale in Emilia-Romagna, membro del Nucleo di Valutazione per i bandi sull'efficienza energetica per la Regione Emilia-Romagna. Ha svolto, inoltre, attività di valutazione delle proposte per l'accesso al meccanismo dei Certificati Bianchi ed ha all'attivo numerose partecipazioni a Progetti Europei in ambito energetico.