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

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Implementazione upgrade Portale PELL IP

Giulio Schisani, Marco Proietti

customer management it

Report RdS/PTR2020/017

Implementazione upgrade Portale PELL IP
G. Schisani, M. Proietti

Aprile 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: 1.*28 - Consolidamento dell'uso della Piattaforma PELL per dati statici ed implementazione sezione per la raccolta dati dinamici*
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno del Contratto "Sviluppo software per upgrade PELL IP"
Responsabile Unico del Procedimento ENEA: Stefano Pizzuti
Responsabile del Contratto per il Contraente : Antonio Sposito

Indice

Sommario

Il documento che segue riporta i dettagli di sviluppo dei task 1, 2, 3, 4 del contratto di servizio **Sviluppo software per upgrade PELL IP**.

In particolare si sono svolte attività atte:

- all'adattamento dell'interfaccia agli standard di accessibilità per le pagine web (1);
- alla realizzazione del modulo Dashboard (2), ovvero una interfaccia contenente una scacchiera di widget (mappe, tabelle, gauge, grafici) customizzabili per gruppo di utenti e per dati da visualizzare.

  L'interfaccia Dashboard è stata corredata di un pannello di controllo lato amministrativo che permette una più facile gestione delle dashboard e l'assemblaggio dei widget nelle stesse;
- alla realizzazione di un sistema di API (3) che permetta il caricamento asincrono delle schede censimento sul sistema e la registrazione delle schede tramite un apposito job di lettura, scrittura e notifica;
- all'estensione del tool Oxytech (4) al fine di far recuperare i dati delle zone omogenee direttamente dal database esistente. Tali dati vengono utilizzati per il completamento della form.

Introduzione

Il lavoro svolto all'interno dell'Accordo di Programma ha avuto lo scopo di migliorare la fruibilità dell'applicativo sia dal punto di vista formale che dei contenuti sia per le utenze base che per le utenze di natura *enterprise*.

Dal punto di vista formale è stata effettuata un'attività di "bonifica" e ristrutturazione delle pagine HTML al fine di renderle quanto più compliant con gli standard AGID per l'accessibilità; questa attività ha tuttavia tenuto conto di preservare il look and feel del sito senza stravolgerlo in maniera macroscopica.

Per quanto concerne le attività più di natura applicativa è stato introdotto il nuovo modulo Dashboard; il modulo è accessibile agli utenti registrati e permette tramite la profilazione del gruppo di appartenenza di visualizzare una serie di informazioni aggregate tramite un pattern di widget disposti a scacchiera (la dashboard) interamente configurabile dall'amministratore.

L'estrema plasticità del modulo relativamente alla sua fase costruttiva permetterà la creazione di interfacce accattivanti per la rappresentazione di qualsiasi set di dati (es. KPI) preventivamente inserito nel sistema.

La dashboard è in grado di renderizzare dati secondo a seconda della dimensione di output: dati monodimensionali sono raffigurabili nei grafici di tipo gauge o pie; dataset bidimensionali in grafici di tipo linechart, dataset tridimensionali tramite grafici di tipo scatterplot e dati multidimensionali e geografici in mappe geografiche.

Una terza attività di notevole interesse, in particolare perché pone le basi per ulteriori implementazioni, è stata la creazione di un sistema di API REST con autenticazione tramite token JWT e l'esposizione di un primo quivier di endpoint per innescare varie tipologie di servizio. Il sistema creato migliora notevolmente le potenzialità di fruizione delle funzionalità del PELL e rende più matura l'intera piattaforma. Il servizio rende infatti possibile a terze parti (ovvero a utenti registrati con piattaforme di tipo *enterpise*) di attivare il caricamento delle Schede Censimento in maniera meccanizzata ed asincrona liberando in questa maniera la fase di caricamento schede da operazioni di natura manuale.

Infine in questa fase si è intervenuti ad una modifica del modulo Oxytech (modulo già esistente nel sistema) e impiegato per la creazione di un report di simulazione illuminotecnica.

Il popolamento del modulo in questione oltre alla classica modalità manuale può ora interfacciarsi con i dati presenti a sistema relativi alle schede caricate dal singolo utente; in questa maniera si semplifica e si migliora l'operazione di selezione dati utilizzando una fonte certa per gli stessi (i dati sono infatti stati preliminarmente validati in fase di creazione scheda).

1 Descrizione delle attività svolte e risultati

1.1 Attività 1 – Layout Frontend Portale

Dal punto di vista della accessibilità il sistema è stato reso compliant con gli standard e le linee guida espresse dall'AGID per i servizi digitali delle PA (https://www.agid.gov.it/it/argomenti/linee-guida-design-pa), saranno pertanto rispettati gli standard WCAG 2.0 AA.

In particolare:

Contrasto tra primo piano e sfondo: il testo e il relativo sfondo (compreso il testo contenuto nelle immagini), rispetta ora il rapporto di contrasto, basato su un algoritmo per siti WCAG 2.0 AA.

Forma e colore: le informazioni che veicola una pagina non sono dipendenti unicamente da aspetto, forma, colore, dimensione, ubicazione visiva, orientamento o suono.

Link e controlli: il codice del contenuto web viene usato secondo le specifiche W3C.

Alternative a oggetti non testuali: tutti gli elementi non testuali, come immagini, grafici, video e audio, hanno una alternativa testuale equivalente quando veicolano un significato, un'informazione o una funzione.

Link e controlli: tutti i componenti dell'interfaccia sono utilizzabili tramite comandi da tastiera analogamente a quanto si riesce a fare col mouse.

Valore, ruolo e stato: si usano componenti standard dell'HTML per la gestione del focus degli elementi. Sono integrati tag per tecnologie assistive del tipo WAI ARIA.

Per quanto concerne la modalità operativa si è fatto uso di tool di analisi dell'interfaccia in particolare:

- **AI Inspector**
  - Link: https://addons.mozilla.org/en-US/firefox/addon/ainspector-wcag/
  - Funzionamento: il tool una volta installato come plugin sul browser permette l'evidenziazione attiva degli errori secondo varie categorie: errori di landmarks, headings, style, ec.. Il tool permette una analisi approfondita del sistema pagina per pagina e
- **WebAIM**
  - Link: https://webaim.org/
  - Funzionamento: anche questo tool installabile come plugin sul browser permette una evidenziazione attiva dei contenuti con errori. Questo tool si presenta particolarmente adatto per l'individuazione di problematiche relative alla compatibilità di natura cromatica dei vari elementi ipertestuali

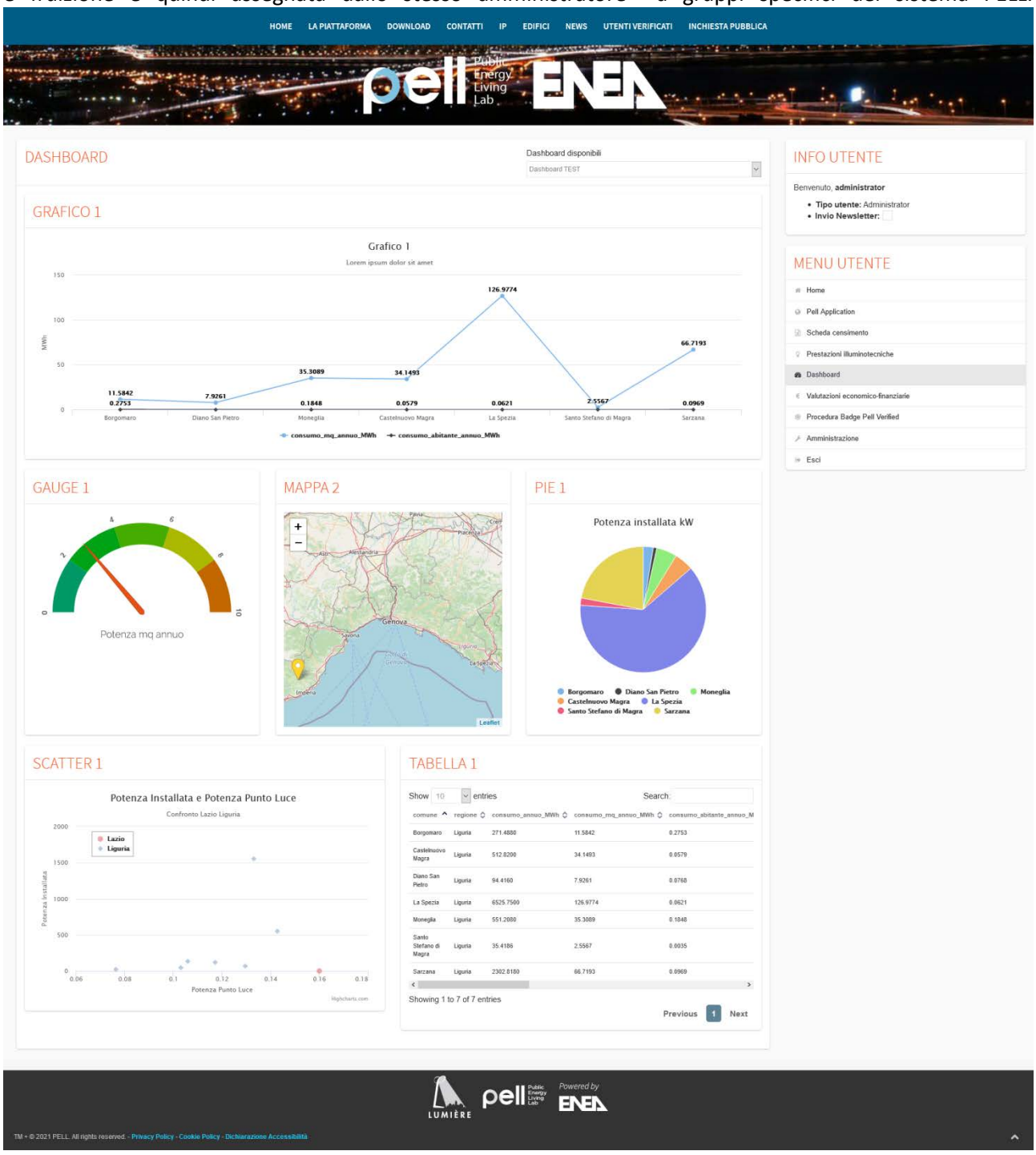
La maggior parte degli interventi è stata rivolta all'inserimento dei tag ARIA, alla ristrutturazione di alcune pagine secondo un corretto schema di incapsulamenti di tag; alla correzione della sequenza di tag title (h1, h2, h3, ecc..); all'inserimento di riferimenti *alternative* per le immagini; alla ricolorazione di alcuni elementi testual, come pulsanti e link, in maniera da rendere il contrasto cromatico coerente con l'indice minimo richiesto il tutto nell'ottica di mantenere una uniformità di stile grafico con la struttura nativa del sito.

1.2 Attività 2 - Implementazione Dashboard

Lo sviluppo dell'attività relativa alla produzione del modulo Dashboard è stato diviso in due parti: frontend e gestione.

1.2.1 FRONTEND

Il modulo prevede la possibilità di creare da parte dell'amministratore una serie di dashboard la cui visibilità e fruizione è quindi assegnata dallo stesso amministratore a gruppi specifici del sistema PELL.



L'interfaccia è accessibile dal menu utente di destra alla voce Dashboard.

Una volta raggiunta la pagina, l'utente in funzione della categoria cui appartiene, avrà in visualizzazione diretta la dashboard di default; un menù a tendina posto nell'angolo superiore destro dell'interfaccia permette invece lo spostamento su un'altra dashboard di cui l'utente ha visibilità.

1.2.2 Struttura modulare (Dashboard e Widgets)

L'oggetto **dashboard** è da intendersi costituito da un insieme di componenti modulari minori detti **widget**. Un widget è un sub modulo che ha una proprietà di natura tipologica e una posizionale. Graficamente il widget è rappresentabile come uno spazio della dashboard in cui viene espressa la raffigurazione grafica o tabellare di un dataset. Ogni Dashboard può contenere al suo interno fino ad un massimo di 12 widget (questo numero può essere tuttavia aumentato essendo parametrico).

I widget si dispongono sul piano della Dashboard secondo uno schema a scacchiera dove è possibile determinare le dimensioni in larghezza delle singole aree per via parametrica. Lo spazio occupato in larghezza da un singolo widget può essere 1/3, 1/2 o la totalità dell'area orizzontale della dashboard.

Ciò significa che si possono avere su una stessa linea 1, 2 o 3 widget. Per quanto riguarda l'altezza dello stesso si può intervenire parametricamente solo sul tipo mappa quando questo si sviluppa su tutta la lunghezza disponibile della dashboard, questo al fine di rendere più agevole una visualizzazione di entità geografiche su uno schermo.

I widget sono completamente responsive design il che significa che in una vista da cellulare si ridispongono autonomamente in colonna (perdendo dunque la disposizione a scacchiera) in maniera da ottimizzare lo spazio disponibile.

I widget disponibili sono di 6 tipologie: linechart, mappa, scatter, tabella, gauge, piechart. Alcune tipologie sono pensate più per una visualizzazione in widget di modulo orizzontale 1/3 come ad esempio il tipo piechart e il tipo gauge, proprio per il fatto che sono già comprensibili in uno spazio piccolo. Per le altre tipologie si consiglia invece un settaggio in moduli da 1/2 o 1.

1.2.3 Origine dati e parametri di renderizzazione

Ogni widget, oltre ad un attributo che ne indica la tipologia e la posizione sul piano, per poter essere disegnato necessita di due classi di informazioni che vengono sempre gestite dall'amministratore nella pagina di gestione: informazioni circa **l'origine dei dati e parametri per la renderizzazione** del singolo widget. Con **informazioni circa l'origine dei dati** si intende la modalità con cui il sistema deve estrarre i dati per quel widget, si tratta solitamente di una tabella o di una vista presente nel sistema a cui si può applicare una particolare e customizzabile query di estrazione.

I **parametri di renderizzazione** sono invece le informazioni dettaglio che servono al plugin javascript di costruzione del widget per poter "graficare" l'oggetto: ad esempio i colori dei POI di una mappa in funzione di particolari soglie o il nome del sottotitolo di un widget, ecc. Il dettaglio della costruzione di questi elementi viene esposto nel paragrafo relativo al backend.

1.2.4 BACKEND - Architettura MVC del modulo dashboard

Il modulo Dashboard si sviluppa secondo i principi del paradigma MVC (Model View Controller) pertanto sia per quanto riguarda l'interfaccia di visualizzazione che quella di gestione viene seguito questo schema.

Controllers

- **Dashboard**
  - **Path:** *application\controllers\Dashboard.php*
  - **Funzionalità:** in questa classe vengono definiti i metodi per la renderizzazione della pagina di frontend e tutte le dipendenza javascript e css della stessa
- **Dashboard API**

- **Path:** *application\controllers\api\enea\Dashboard.php*
- **Funzionalità:** in questa classe vengono definiti i metodi per l'estrazione delle dashboard disponibili, nonché la preparazione dei dati utili alla creazione dei singoli widget

Models

- **Visibilità Dashboard per classe di utente**
  - **Path:** *application\models\common\Dashboard_model.php*
  - **Funzionalità:** recupera le informazioni circa la visibilità delle dashboard
- **Dashboard CRUD**
  - **Path:** *application\models\enea\Dashboard_model.php*
  - **Funzionalità:** gestisce il CRUD relativamente alla creazione della dashboard
- **Widget CRUD**
  - **Path:** *application\models\enea\Widget_model.php*
  - **Funzionalità:** gestisce il CRUD relativamente alla creazione dei widget

View

- **BE**
  - **Path:** *application\views\administration\dashboards\content.php*
  - **Funzionalità:** si occupa della renderizzazione della pagina di backend per l'amministrazione della dashboard, si compone di n sotto viste una per tipo di widget. Ogni subview corrisponde alla form di inserimento dati (parametri e di query) per ogni tipologia di widget
- **BE Javascript**
  - **Path:** *assets\js\administration\dashboards.js*
  - **Funzionalità:** si occupa della "graficizzazione" dell'interfaccia di gestione sia per quanto riguarda le creazione dei widget che della dashboard
- **FE**
  - **Path:** *application\views\enea\dashboard\content.php*
  - **Funzionalità:** si occupa della renderizzazione della pagina di frontend; la costruzione del telaio di ogni dashboard e dei relativi widget è invece demandato a funzioni javascript che si occupano di modellare *on the fly* gli elementi.
- **FE Javascript**
  - **Path:** *assets\js\enea\dashboard\dashboard.js*
  - **Funzionalità:** si occupa della graficizzazione dell'intera dashboard lato frontend, fa uso di librerie specifiche per i singoli widget e di sub funzioni distribuite in javascript specifici nel sub folder *assets\js\enea\dashboard\widgets*

- **PLUGIN JS**
  Sono stati impiegati i seguenti plugin di terze parti per la realizzazione dei widget:
  - **Grafici linechart, piechart e scatter:** HighCharts; *assets/plugins/highcharts/*
  - **Mappe:** Leaflet; *assets/plugins/leaflet*
  - **Gauge:** svgJs; *assets/plugins/svg*
  - **Struttura Dashboard:** GridStack; *assets/plugins/gridstack/gridstack*

1.2.5 Database e meccanismo di funzionamento tecnico del modulo dashboard

Dal punto di vista del database il modulo necessita di due solo tabelle: *dashboard* e *widget*.

Schema Tabella Dashboard

```
CREATE TABLE IF NOT EXISTS `dashboard` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `label` varchar(256) DEFAULT NULL,
  `template` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL CHECK
(json_valid(`template`)),
  `groups` varchar(256) DEFAULT NULL,
  `is_default` tinyint(1) DEFAULT 0,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=26 DEFAULT CHARSET=latin1;
```

Schema Tabella Widget

```
CREATE TABLE IF NOT EXISTS `widget` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(256) DEFAULT NULL,
  `label` varchar(256) DEFAULT NULL,
  `type` varchar(32) DEFAULT NULL,
  `origin` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL CHECK
(json_valid(`origin`)),
  `params` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL CHECK
(json_valid(`params`)),
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=15 DEFAULT CHARSET=latin1;
```

La tabella Dashboard contiene informazioni relative alle singole Dashboard e in particolare il campo struttura, ovvero la registrazione della struttura a scacchiera della dashboard con la posizione relativa dei singoli widget. Questa informazione è contenuta in un json come questo di esempio che segue.

Esempio di struttura Dashboard

```
[
	[{
		"unity": 12,
		"name": "chart_1"
	}],
	[{
		"unity": 4,
		"name": "gauge_1"
	}, {
		"unity": 4,
		"name": "map_2"
	}, {
		"unity": 4,
		"name": "pie_1"
	}],
	[{
		"unity": 6,
		"name": "scatter 1"
	}, {
		"unity": 6,
		"name": "table_1"
	}]
]
```

La tabella Widget contiene informazioni relative ai singoli Widget; in particolare i campi *origin* e *params* ovvero rispettivamente le informazioni circa l'origine dei dati e le specifiche per la renderizzazione del grafico. A titolo di esempio si riporta un widget di tipo mappa con i dati relativi

Esempio di origine dati per widget di tipo Mappa

```
{
	"table": "kpi_test",
	"conditions": [{
		"mode": "where_and",
		"field": "regione",
		"value": "Liguria"
	}],
	"join": [{
		"table": "comune",
		"field_join": "comune",
		"field_origin": "comune"
	}]
}
```

Esempio di parametri per widget di tipo Mappa

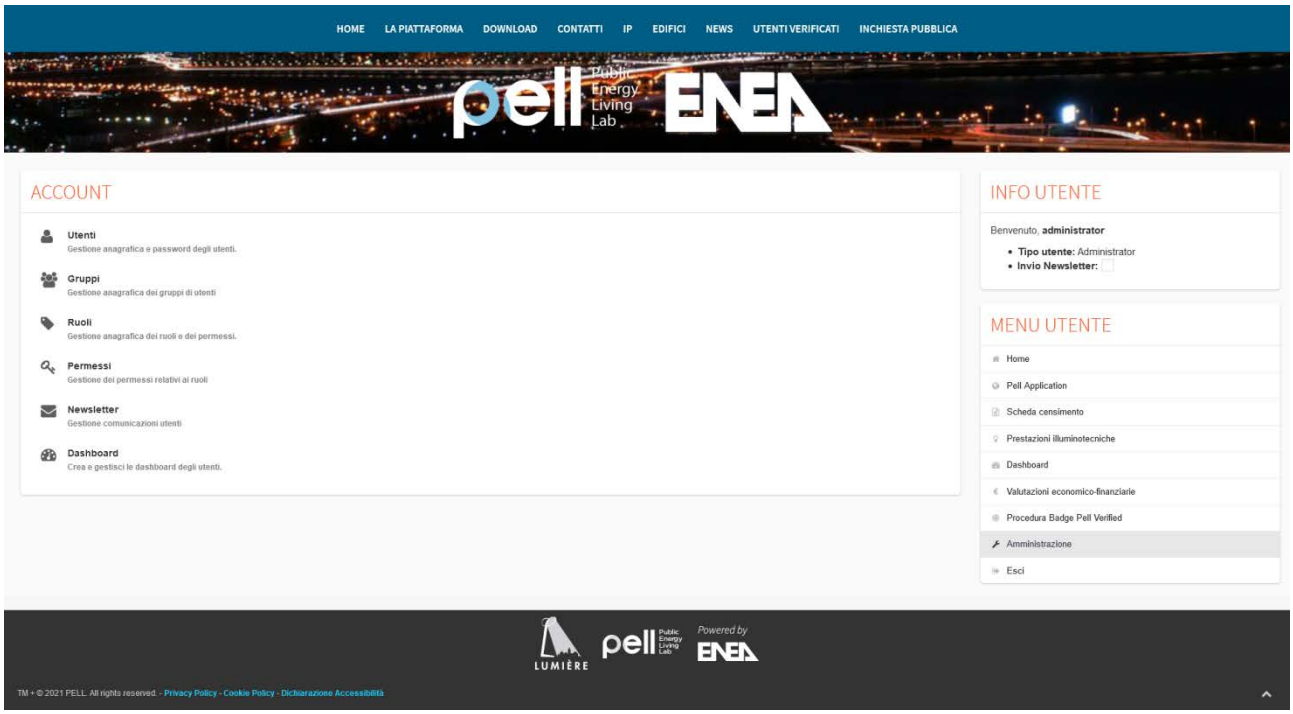
```
{
	"subTitle": "Liguria",
	"mapCenterLat": 44.4222,
	"mapCenterLng": 8.9052,
	"mapZoom": 8,
	"poiColorLogic": {
		"logic": "threshold",
		"keyParam": "potenza_abitante_annuo_kW",
		"colors": {
			"green": [0, 100],
			"yellow": [100, 200],
			"red": [200, 300]
		}
	},
	"popupItems": [
		"comune",
		"potenza_abitante_annuo_kW",
		"potenza_installata_kW",
		"potenza_mq_annuo_kW",
		"potenza_punto_luce_kW"
	]
}
```

Come si può vedere dall'esempio di cui sopra la mappa per essere disegnata necessita di varie tipologie di informazioni che sono derivate da ulteriori viste o tabelle presenti nel DB (ad esempio una tabella pre popolata di KPI) e di specifici campi della stessa da usare come categorie e serie per i calcoli e la visualizzazione.

Le informazioni possono dunque essere forgiate direttamente a livello di database da parte dell'amministratore oppure lo stesso si può avvalere per la costruzione degli stessi di una interfaccia di backend che permette una creazione guidata di dashboard e widgets

1.2.6 Interfaccia di amministrazione Dashboard

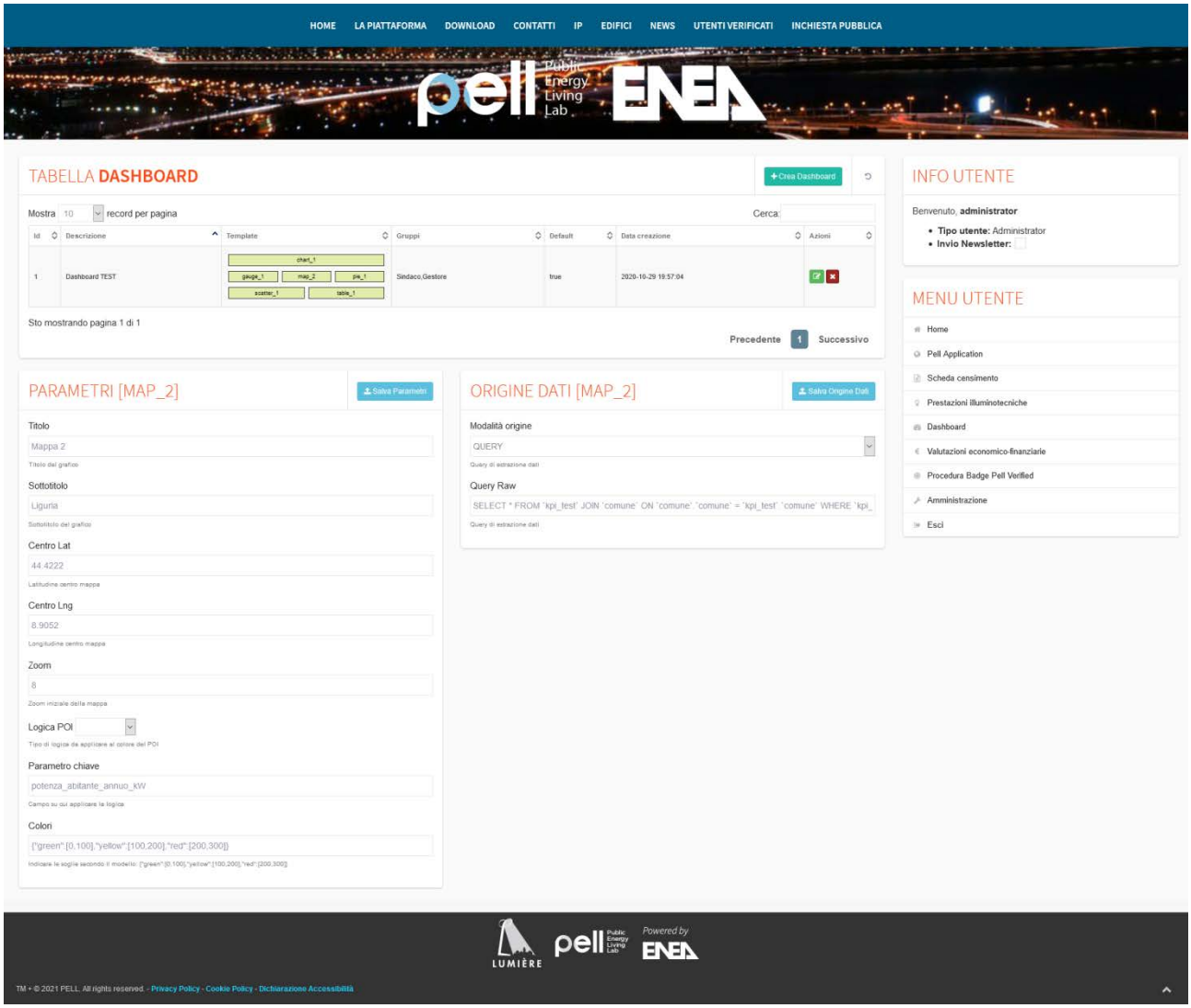
L'amministratore troverà all'interno del menu amministrativo dedicato una nuova voce: Dashboard



Una volta acceduti si avrà di fronte il pannello di controllo della dashboard che si compone di tre elementi:

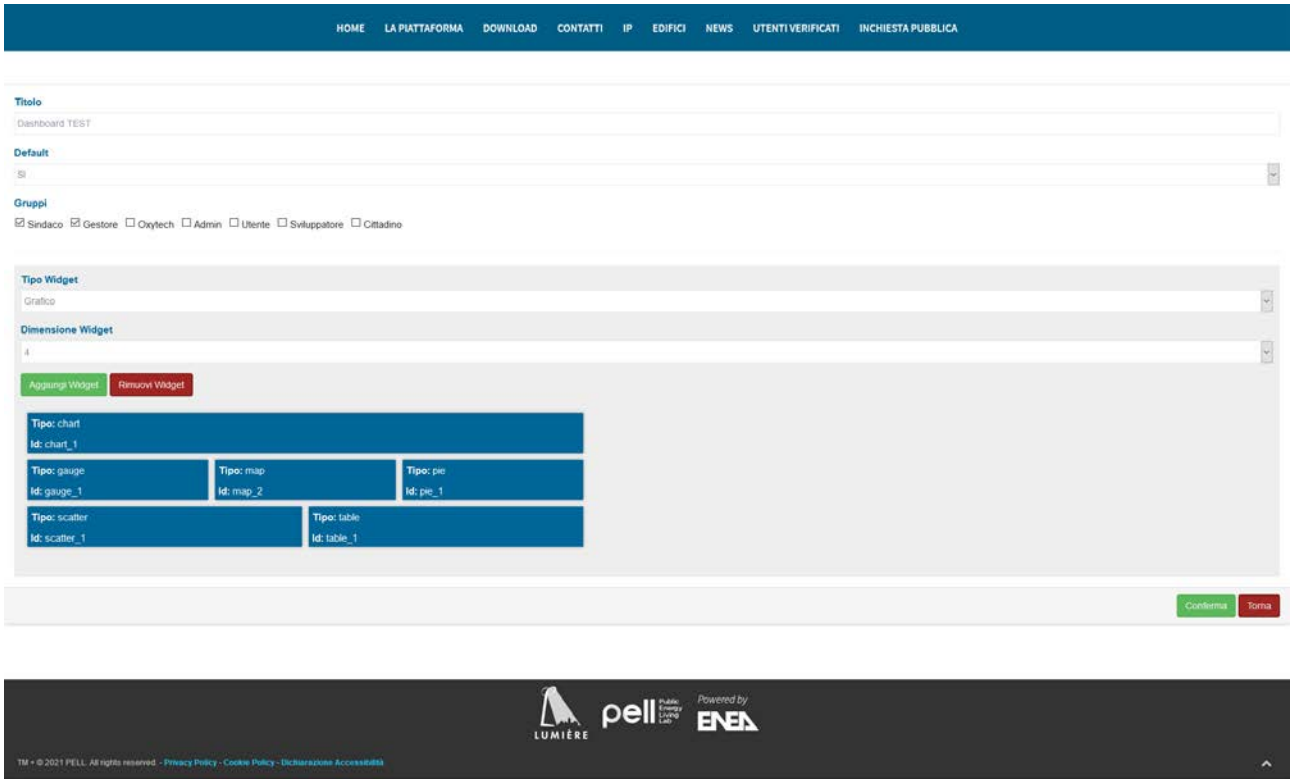
1) una tabella riassuntiva delle dashboard create; tale tabella oltre alla *label* della Dashboard mostra un indice ed un *name* univoco generato dal sistema, la tabella riporta inoltre le visibilità di gruppo per quella dashboard e uno schema minimo della struttura per come apparirà a frontend
2) Una coppia di form che permettono di editare i campi relativi al singolo widget
3) Una pagina a comparsa che permette di editare la struttura della dashboard

Vista sinottica



Al click sul singolo mini widget della tabella si attiva in maniera dinamica la rispettiva duplice form di inserimento (parametri e origine dati). La form di sinistra, quella parametrica, segue un modello specifico per ogni tipologia di widget (pertanto un widget di tipo mappa avrà campi diversi da un widget di tipo *linechart*)

Vista creazione ed editing dashboard



Questa vista permette di aggiungere cancellare i vari widget e di attribuire il nome e le visibilità della dashboard. Il posizionamento del widget nella dashboard è gestibile tramite un semplice *drag and drop* autoadattativo del blocco widget.

1.3 Attività 3 – Upload automatico schede censimento tramite PELL Gateway

Il caricamento asincrono delle schede nasce dall'esigenza di permettere ad uno o più utenti registrati di poter caricare sul sistema le proprie schede censimento senza utilizzare l'interfaccia di frontend e senza dover aspettare il caricamento sequenziale della singola scheda con tempi talvolta lunghi a causa della mole di dati delle stesse.

La funzionalità del caricamento asincrono consiste nel mettere a disposizione degli utenti, che hanno facoltà di caricare schede, una apposita API e relativi endpoint per gestire il processo di upload

1.3.1 Principio di funzionamento e fasi

Il principio del caricamento asincrono si può riassumere nella seguente descrizione di passi

- **Fase 1: autenticazione**

  Un utente abilitato si autentica su una specifica rotta dell'API utilizzando il suo username e la password.

  Il sistema restituisce un token JWT valido per un tempo configurabile (un giorno, un'ora ecc..). Tramite questo token l'utente sarà quindi in grado di interrogare i vari endpoint a disposizione per effettuare operazioni

- **Fase 2: upload scheda su db di lavorazione**

  Utilizzando il token di cui sopra l'utente carica una scheda in formato .xml o .zip nel filesystem PELL alla stregua di un caricamento di tipo manuale e l'operazione viene registrata su una tabella che raccoglie lo stato di lavorazione del pacchetto caricato. La scheda a differenza di un caricamento da frontend di tipo manuale non viene immediatamente processata, ma rimane in uno stato di *pending* fino all'attivazione di un job gestito dal sistema.

  In sostanza il sistema tramite API disaccoppia l'upload della scheda nel sistema dal suo effettiva trasformazione in un record del database.

- **Fase 3: inserimento del record scheda nel DB**

  L'operazione onerosa in termini di tempo è appunto il caricamento nel DB Pell e questa fase viene appunto gestita asincronicamente dal job preposto. Il job utilizza una chiamata con autenticazione ad unno specifico metodo del sistema che innesca il processamento delle schede. Le schede vengono dunque recuperate una ad una e il processo si preoccupa di caricare ovvero di transcodificare l' XML passato all'interno delle varie tabelle del database e attuare la registrazione della scheda vera e propria nel sistema.

- **Fase 4: notifica**

  Una volta processata e dunque inserita correttamente nel DB PELL IP viene notificata all'utente l'avvenuta esecuzione del processo tramite mail. Un sistema di log registra inoltre le varie fasi della procedura

1.3.2 Elenco endpoint disponibili (schema compatto)

L'attività relativa a questa implementazione ha prodotto i seguenti endpoint

Per lo schema di dettaglio consultare l'appendice di questo documento

- **Login**
  - **Path:** *pell-ip-portal/api/login*
  - **Verb:** GET
  - **Input:** username, password
  - **Output:** token JWT
  - **Funzionalità:** Permette all'utente di autenticarsi sul sistema e ricevere il token JWT che dovrà impiegare per le chiamate successive
- **Sheets List**
  - **Path:** *pell-ip-portal/api/sheets*
  - **Verb:** GET
  - **Input:** token JWT
  - **Output:** elenco schede censimento attualmente caricate nel sistema PELL
  - **Funzionalità:** Permette all'utente di controllare lo stato delle schede effettivamente caricate a sistema, corrisponde alla vista schede che avrebbe utilizzando l'interfaccia di frontend
- **User Info**
  - **Path:** *pell-ip-portal/api/user*
  - **Verb:** GET
  - **Input:** token JWT
  - **Output:** informazioni sulla propria utenza
  - **Funzionalità:** Permette all'utente di controllare le informazioni che il sistema PELL possiede circa la sua utenza (es. gruppo di appartenenza, mail di invio notifiche, comuni di riferimento, ecc..)
- **Sheet Download**
  - **Path:** *pell-ip-portal/api/sheet/{id }*
  - **Verb:** GET
  - **Input:** token JWT
  - **Output:** scheda in formato .xml
  - **Funzionalità:** Permette all'utente di effettuare il download di una scheda censimento presente nel sistema tramite ID scheda in formato xml. Equivale alla funzione di esportazione scheda presente nell'interfaccia di frontend
- **Sheet Upload**
  - **Path:** *pell-ip-portal/api/sheet/*
  - **Verb:** POST
  - **Input:** token JWT, sheet_label, sheet_type, sheet_current
  - **Output:** conferma caricamento
  - **Funzionalità:** Permette all'utente di caricare una scheda censimento nel sistema. La scheda una volta caricata viene messa in fase di pending in attesa che il job la lavori definitivamente ovvero la registri come record nel sistema trasferendo le informazioni dell'XML in record del database
- **Uploaded sheets list**
  - **Path***: pell-ip-portal/api/uploaded_sheets*
  - **Verb:** GET
  - **Input:** token JWT
  - **Output:** elenco delle schede caricate a sistema tramite API
  - **Funzionalità:** Permette all'utente di visualizzare una lista delle schede caricate via API, vengono visualizzate sia le schede lavorate dal job sia quelle in stato pending
- **Delete uploaded sheet**

- **Path:** *pell-ip-portal/api/uploaded_sheet/{sheet_code}*
- **Verb:** DELETE
- **Input:** token JWT
- **Output:** risultato operazione
- **Funzionalità:** Permette all'utente di eliminare una scheda caricata via API che non sia stata ancora lavorata

- **Load sheet on DB**
  - **Path:** *pell-ip-portal/api/load_on_db/{sheet_code}*
  - **Verb:** GET
  - **Input:** token JWT
  - **Output:** risultato operazione
  - **Funzionalità:** Permette ad un utente amministratore di effettuare manualmente la registrazione nel sistema PELL di una scheda caricata. Sostanzialmente mima il processo che viene eseguito dal job

1.3.3 Esecuzione JOB

L'esecuzione del JOB ovvero la sua schedulazione è una attività a carico del gestore del sistema PELL-IP

L'invocazione del job per il caricamento schede è eseguibile tramite la seguente chiamata dalla root dell'applicazione

```
php index.php job loadsheet worksheet [secret]
```

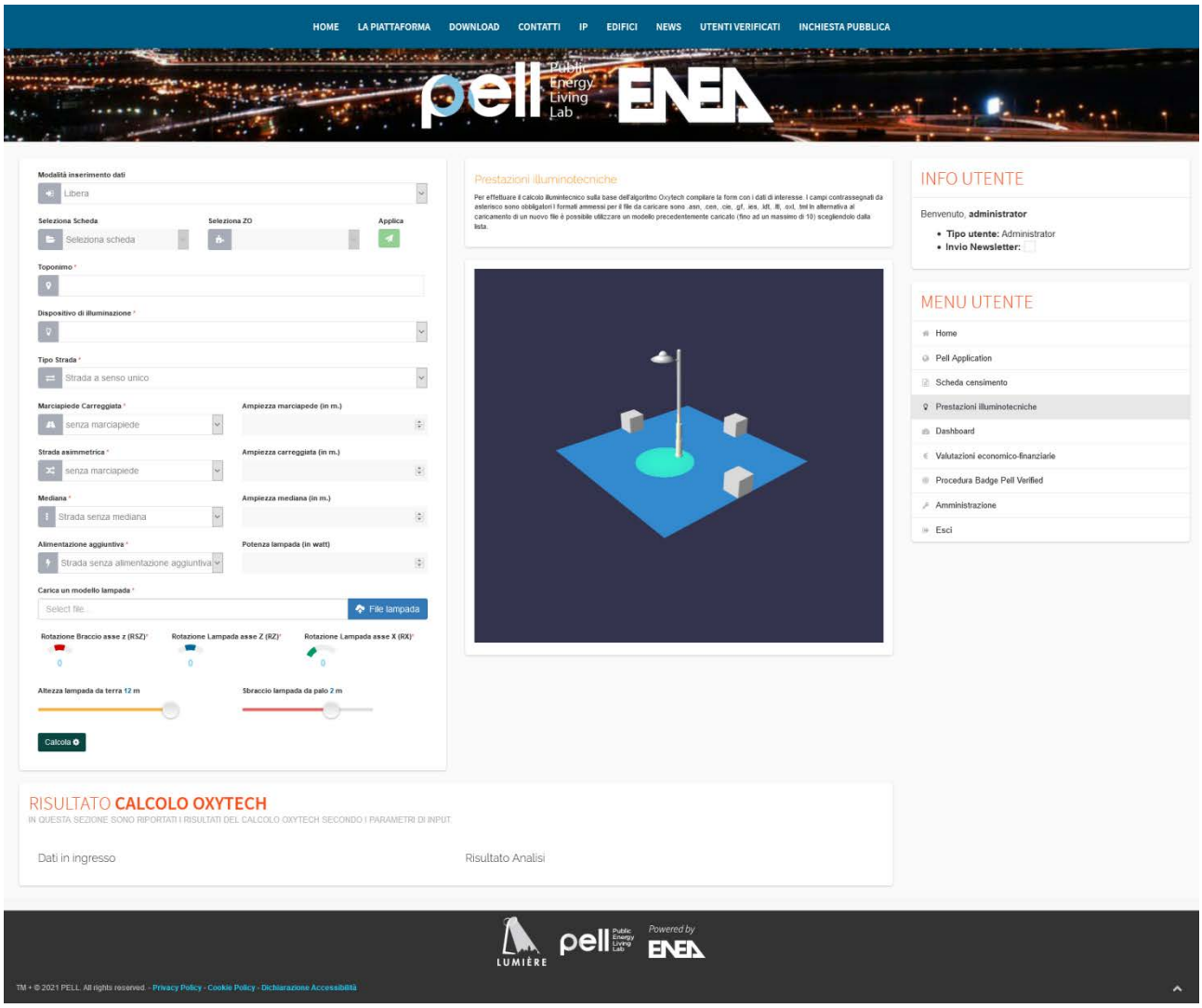
Il secret è configurabile a livello di config di CI al path *\application\config\enea\enea_pell.php* sotto la variabile ***jobSecret***

1.4 Attività 4 – Allaccio dati statici a tool Oxytech

1.4.1 Interfaccia

L'attività relativa al modulo Oxytech si è risolta in una estensione della capacità di input del modulo stesso.

Il modulo Oxytech è un modulo del sistema PELL IP che permette di effettuare la simulazione e il calcolo dell'efficienza illuminotecnica di un apparecchio (un palo della luce o una fonte di illuminazione posta su strada pubblica).



Il sistema permette di inserire i dati relativi alla sede stradale e una scheda illuminotecnica in formato .asn, .cen, .cie, .gf, .ies, .ldt, .ltl, .oxl

La combinazione di queste informazioni produce (tramite l'algoritmo oxytech) un report di efficienza energetica relativo alla fonte luminosa inserita.

Il modulo è stato esteso nella sua funzionalità di compilazione dei dati relativi alla sede stradale.

Una volta atterrati sulla pagina il sistema mette a disposizione la possibilità di selezionare una scheda censimento fra quelle caricate dall'utente; una volta selezionata la scheda una seconda query si preoccupa dell'estrazione delle Zone Omogenee (le strade) relative a quella Scheda e le propone come *option* in una seconda *select*.

Una volta selezionata la zona omogenea di riferimento viene avviata l'estrazione dei dati utili alla compilazione della scheda Oxytech che viene pertanto pre popolata con le informazioni necessarie, come ad esempio: la presenza del marciapiede, la tipologia di sede stradale, l'altezza dell'apparecchio (palo), lo sbraccio, inclinazione della lampada, ecc..

I dati raccolti oltre a popolare la form modificano dinamicamente anche il modello 3d in scala presente nella pagina che permette di avere una visualizzazione immediata e realistica dell'effettiva collocazione dell'elemento luminoso rispetto alla sede stradale.

1.4.2 Architettura MVC del modulo Oxytech

Il modulo Oxytech si sviluppa secondo i principi del paradigma MVC (Model View Controller) pertanto sia per quanto riguarda l'interfaccia di visualizzazione che quella di gestione viene seguito questo schema.

Controllers

- **Oxytech**
  - **Path:** *application\controllers\enea\Oxytech.php*
  - **Funzionalità:** in questa classe vengono definiti i metodi per la renderizzazione della pagina di frontend e tutte le dipendenza javascript e css della stessa
- **Oxytech API**
  - **Path:** *application\controllers\api\enea\Oxytech.php*
  - **Funzionalità:** in questa classe vengono definiti i metodi per la gestione delle chiamate asincrone relative all'interfacciamento con il server Oxytech e per l'estrazione dei dati relative alle Zone Omogenee

Models

- **Oxytech CRUD**
  - **Path:** *application\models\enea\Oxytech_model.php*
  - **Funzionalità:** gestisce il CRUD relativamente alla gestione dei dati Oxytech

View

- **FE**
  - **Path:** *application\views\enea\oxytech\content.php*
  - **Funzionalità:** si occupa della renderizzazione della pagina di frontend;

- **FE Javascript**
  - **Path:** *assets\js\enea\oxytech\oxytech.js*
  - **Funzionalità:** si occupa della gestione lato frontend della form e della generazione del modello tridimensionale di apparecchio e sede stradale

- **PLUGIN JS**
  Sono stati impiegati i seguenti plugin di terze parti per la realizzazione dei widget:
  - **Modello 3D:** Babylon; *assets/plugins/babylon/babylon*
  - **Selettori Gauge:** Knob; *assets/plugins/jQuery-Knob*

1.5 Test e Sicurezza

L'attività di TEST è integrata in quella precedente del sistema PELL-IP

In particolare relativamente agli sviluppi attuali:

Dashboard: registrazione dei permessi per la lettura e scrittura dei dati relativi sia al modulo di frontend e di backend

API caricamento asincrono: utilizzo della libreria JWT Token per la generazione sicura di un token JWT che permette di gestire il processo di autenticazione *stateless*. Le operazioni di lettura e scrittura su DB avvengono tutte sfruttando il sistema *Active Record* di CodeIgniter

(https://codeigniter.com/userguide2/database/active_record.html)

I dati di input di ogni chiamata vengono strettamente validati prima di essere passati ai metodi del modulo relativo.

JOB caricamento asincrono: il JOB è attivabile esclusivamente da shell e prevede la conoscenza di un *secret* per il suo innesco.

1.6 APPENDICE

1.6.1 Schema API caricamento schede

Login

General	
Servizio	[URL_BASE]/api/login
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Body	
Type	URLencoded form data
Parametri	username=[string/string]
	password=[string/string]

Esempio	{
	"username": "myusername",
	"password": "mypassword"
	}

Return	

Success	{
	"status": 200,
	"error": false,
	"code": "01",
	"message": "User logged In successfully",
	"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
	}

Failure	{
	"status": 401,
	"error": true,
	"code": "10",
	"message": "Authentication Failed"
	}

Sheets List

General

Servizio	[URL_BASE]/api/sheets
Tipo Chiamata	GET
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	
Esempio	
Return	
Success	{ "status": 200, "error": false, "code": "01", "message": "PELL sheets", "data": [{ "id": "154", "label": "SchedaBracciano", "owner": "MarioRossi", "an_ip_ist": "12058013", "an_ip_nom": "Bracciano", "n_pod": 1, "n_qe": 1, … …

Failure	{
	"status": 401,
	"error": true,
	"code": "10",
	"message": "Authentication Failed"
	}

User Data

General

Servizio	[URL_BASE]/api/user
Tipo Chiamata	GET
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	
Esempio	
Return	

Success	{
	"status": 200,
	"error": false,
	"messages": "User info",
	"data": {
	"id": "67",
	"username": "MarioRossi",
	"first_name": "Mario",
	"last_name": "Rossi",
	"email": "mario.rossi@example.com",
	"language": "it",
	"group": "45",
	"is_admin": false,
	…
	…

Failure	{
	"status": 401,
	"error": true,
	"code": "10",
	"message": "Authentication Failed"
	}

Download Sheet

General	
Servizio	[URL_BASE]/api/sheet/{id}

Tipo Chiamata	GET
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	
Esempio	
Return	
Success	<?xml version="1.0" encoding="utf-8"?> <CensusTechSheet xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gco="http://www.isotc211.org/2005/gco" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gsr="http://www.isotc211.org/2005/gsr" xmlns:gts="http://www.isotc211.org/2005/gts" xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <!--Begin::Area--> <Area> <!--META_SUP--> <gml:Polygon srsName="http://www.opengis.net/def/crs/EPSG/0/4936" gml:id="POL1"> <gml:exterior> <gml:LinearRing> <gml:posList srsDimension="2">42.1 12.17 42.03 12.17 42.03 12.2 42.1 12.2</gml:posList> … …
Failure	{ "status": 401, "error": true, "code": "10", "message": "Authentication Failed" }

Upload Sheet

General	
Servizio	[URL_BASE]/api/upload

Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	sheet_label=[string/string]
	sheet_type =[string/string] values: before_redevelopment, after_redevelopment, maintenance, simulation
	sheet_current =[int/int]

Esempio	{
	"sheet_label": "TestName",
	"sheet_type": "before_redevelopment",
	"sheet_current": 1
	}

Return	

Success	{
	"status": 200,
	"error": false,
	"code": "02",
	"message": "Upload done successfully"
	}

Failure	{
	"status": 401,
	"error": true,
	"code": "12",
	"message": "Upload Failed"
	}

Upload Sheets List

General	
Servizio	[URL_BASE]/api/uploaded_sheets
Tipo Chiamata	GET
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	
Esempio	
Return	
Success	{ "status": 200, "error": false, "code": "03", "message": "PELL list of uploaded sheets via API", "data": [{ "id": "3", "sheet_name": "EsempioCorretto20.xml", "sheet_code": "3e7dfbb6-977f-488a-a83b-1fd763799465", "sheet_label": "Prova", "sheet_current": "1", "sheet_type": "before_redevelopment", " … …

Failure	{
	"status": 401,
	"error": true,
	"code": "11",
	"message": "Access denied"
	}

Upload Sheet Delete

General	
Servizio	[URL_BASE]/api/uploaded_sheet/{sheet_code}
Tipo Chiamata	DELETE
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	
Esempio	
Return	
Success	{ "status": 200, "error": false, "code": "04", "message": "The sheet with code: 3e7dfbb6-977f-488a-a83b-1fd763799465 successfully deleted" }
Failure	{ "status": 401, "error": true, "code": "11", "message": "Access denied" }

Upload Sheet Delete

General	
Servizio	[URL_BASE]/api/load_on_db/{sheet_code}
Tipo Chiamata	GET
Header	
Content-Type	application/json
Accept	application/json
Authorization	"Bearer Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJQRUxMIiwiYXVkIjoiUEVMTF9hcGlfdXNlciIsImlhdCI6MTYxNjc1Nzc4NywibmJmIjoxNjE2NzU3Nzk3LCJleHAiOjE2MTY3NjEzODcsImRhdGEiOnsiaWQiOiI2NyIsImVtYWlsIjoiZWRvYXJkby5zY2F6em9jY2hpb0BnbWFpbC5jb20ifX0.FBlsZy3lMU2Rx7pjtFn7iVYQ9pBe2MJyFo42CMqwPgc"
Body	
Type	URLencoded form data
Parametri	
Esempio	
Return	
Success	{ "status": 200, "error": false, "code": "05", "message": " The sheet: 3e7dfbb6-977f-488a-a83b-1fd763799465 has been sucessfully loaded on db" }
Failure	{ "status": 401, "error": true, "code": "11", "message": "Access denied" }

2 Conclusioni

Il lavoro finora svolta ha migliorato l'integrazione e la fruizione dell'applicativo PELL sia dal punto di vista operativo che di usability.

L'interfaccia Dashboard nella sua plasticità permette di dare una visione immediata di contenuti di interesse.

Il sistema di caricamento asincrono tramite API ha permesso di introdurre una autenticazione JWT ed una gestione *stateless* degli utenti. Questa evoluzione apre la porta a possibili integrazioni e implementazioni sempre nell'ottica di esporre a entità esterne e terze parti i servizi offerti da PELL IP.

L'estensione del modulo Oxytech ha permesso di sfruttare i dati interni al sistema per facilitare le simulazioni illuminotecniche.

curriculum scientifico del gruppo di lavoro

Customer Management IT S.r.l. realizza progetti complessi e innovativi usando le più moderne tecnologie dell'Information and Communication Technology.

L'azienda nasce a giugno del 2010 da un gruppo di professionisti del settore informatico che, dopo una serie di esperienze diverse, hanno deciso di proporsi sul mercato con una propria realtà.

CM IT è una giovane azienda ricca di un maturo Know how, ma che porta con sé esperienza, innovazione, capacità e conoscenza delle peculiarità legate ai bisogni della Pubblica Amministrazione.

L'azienda è in grado di realizzare per i propri Clienti servizi "chiavi in mano" per la gestione dei fattori critici di successo, perché conosce a fondo il contesto di riferimento e le loro esigenze.

Il tratto distintivo di CM IT è la costante attenzione alla ricerca e alla sperimentazione di tecnologie emergenti, grazie al suo network di partner italiani e internazionali e alla disponibilità e curiosità dei propri Clienti.

CM IT ha un'idea d'innovazione molto concreta: sperimentare e realizzare ogni giorno soluzioni che rispondano in modo sempre più efficace e semplice alle esigenze dei Clienti.

CM IT ha un fatturato in crescita, nel 2017 è stato di 5,9 ML€ mentre nel corso del 2018 è salito a 6,8 M€; la previsione per l'anno 2019 è in ulteriore crescita.

L'azienda ha nel suo organico 105 professionisti informatici e una serie di consulenti con accordi di collaborazione stabili e continuativi.

Il personale operativo è dislocato nelle maggiori aree produttive ed economiche del paese, in particolare a Roma (sede legale e principale sede operativa), a Napoli e in altre aree del Sud Italia.

CM IT opera sui suoi Clienti sia in gestione diretta sia attraverso attività in subappalto. Il mercato di riferimento è la pubblica amministrazione centrale e locale, ma grazie alla collaborazione con i partner sono in corso anche attività di posizionamento commerciale e tecnico presso alcuni dei più importanti clienti che operano sul mercato Enterprise.

Il mix di competenze permette di ottimizzare la gestione dei gruppi di lavoro, mantenendo non solo un controllo costante sui costi ma assicurando anche un alto contenuto di professionalità.

Nel corso del 2018 si sono consolidati i rapporti con i due principali system integrator: Telecom Italia e Leonardo; con Telecom Italia sono stati sottoscritti nuovi contratti, alcuni pluriennali, che prevedono un fatturato costante per prossimi anni.

Nel corso del 2019 sono state attivate e avviate nuove collaborazioni con Accenture e Capgemini.

L'azienda si occupa di:

- ⇨ Progettazione e realizzazione di:
  - Sistemi informativi;
  - Soluzioni applicative;
- ⇨ Servizi di assistenza applicativa e sistemistica.

Aree di intervento

System and Application Integration

L'ICT mette oggi a disposizione un insieme di soluzioni vasto e complesso. L'innovazione può venire quindi non solo dalla ricerca, ma anche dall'integrazione di tecnologie già esistenti. Lo strumento principale è rappresentato dalle tecnologie di Application Integration che possono essere utilizzate per ottimizzare i processi interni e per abilitare nuovi modelli di business. Perché questo si traduca in effettivo valore per i clienti è necessario comprendere a fondo le loro esigenze e avere la capacità di progettare architetture di

sistema basate sulle nuove piattaforme di integrazione, conoscendo appieno le potenzialità delle nuove tecnologie.

CM IT mette a disposizione dei propri clienti un mix di esperienza e professionalità di eccellenza nelle tecnologie più diffuse, in quelle più innovative e negli ambienti di sviluppo più utilizzati e maggiormente produttivi.

Professional Services

CM IT supporta grandi organizzazioni nella gestione delle proprie infrastrutture ICT. Il successo dei progetti si basa sulla competenza organizzativa maturata in anni di esperienza e supportata da metodologie di project management, di application e change management. Grazie a una presenza capillare su tutto il territorio nazionale, CM IT è in grado di soddisfare le esigenze delle organizzazioni più complesse realizzando i seguenti servizi:

- Application and infrastructure maintenance;
- Desktop e server fleet management;
- Roll out di sistemi hardware e software su tutto il territorio nazionale;
- Progetti di e-learning e di formazione in aula.

Sicurezza Logica

CM IT mette a disposizione dei propri clienti le sue capacità di risk assessment, project management, system integration e network management per realizzare progetti chiavi in mano basati sulle migliori tecnologie di mercato e su partnership affidabili.

I progetti proposti coprono anche le problematiche della sicurezza logica dei sistemi, partendo dall'analisi del rischio e delle vulnerabilità, fino al dettaglio delle soluzioni applicative per realizzare e rendere disponibili in esercizio sistemi software con elevate caratteristiche di sicurezza applicativa.

System Rationalization

CM IT propone ai propri clienti consulenza e soluzioni per la razionalizzazione delle infrastrutture tecnologiche al fine di:

- semplificare le infrastrutture, attraverso la standardizzazione delle piattaforme di base, mediante il consolidamento dei server e dei dispositivi di storage;
- garantire la continuità di servizio, attraverso soluzioni e policy efficienti per il disaster recovery, per l'abbattimento dei tempi di manutenzione pianificata e non pianificata e per il monitoraggio e il tuning delle infrastrutture;
- garantire il provisioning istantaneo delle risorse hardware e software quando sono necessarie;
- minimizzare in modo misurabile il Total Cost of Ownership.

<u>Principali progetti</u>

- ✓ Active Directory Nazionale, Ministero dell'Interno
- ✓ Sistema Informativo della Polizia Stradale, Ministero dell'Interno
- ✓ MIPGWeb (Modello di Indagine Polizia Giudiziaria), Ministero dell'Interno.
- ✓ Sistema informativo per la gestione della sala operativa della Polizia Municipale di Bari
- ✓ Assistenza al SIDDA/SIDNA, Ministero della Giustizia
- ✓ E-commerce Interflora – TecFlora