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

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Raccolta dati energetici residenziali e di un distretto universitario

D.Masucci, S.Panzieri, C.Foglietta, F.Pascucci

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Report RdS/PTR(2021)/005

RACCOLTA DATI ENERGETICI RESIDENZIALI E DI UN DISTRETTO UNIVERSITARIO

Dario Masucci, Stefano Panzieri, Chiara Foglietta, Federica Pascucci (Università Roma Tre)

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - III annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: Implementazione piattaforma raccolta di dati energetici residenziali e predisposizione di metodologie per la connessione alla SCP
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Implementazione piattaforma raccolta di dati energetici residenziali e predisposizione di metodologie per la connessione alla SCP"
Responsabile scientifico ENEA: Fabio Moretti
Responsabile scientifico: Stefano Panzieri, Università Roma Tre

Indice

Indice delle figure

Sommario

In continuità rispetto alle precedenti annualità, le attività svolte durante questo anno si focalizzano sulla realizzazione di due diversi processi per la raccolta di dati energetici. Questi processi si differenziano tra di loro sulla base sia delle modalità di acquisizione e mantenimento delle informazioni, sia della tipologia di utente coinvolto. In particolare, il contributo apportato riguarda le due distinte progettualità introdotte di seguito.

District Database – riguarda la progettazione e realizzazione di un applicativo informatico in grado di gestire la raccolta, ciclica ed automatizzata, delle informazioni sui consumi energetici che caratterizzano il cluster di edifici afferenti al sistema di gestione energetica dell'Università di Roma Tre. Una volta acquisite, le informazioni devono essere inviate al servizio Smart City Platform (SCP) [1] messo a disposizione dal C.R. ENEA "La Casaccia". Partendo dalle specifiche di progetto definite nei precedenti report (per maggiori informazioni consulta il Report RdS/PTR2020/009 "Implementazione piattaforma raccolta di dati energetici residenziali e progettazione piattaforma raccolta dati energetici di un distretto universitario"), al termine di questa annualità è stato implementato il prototipo di applicativo web per la raccolta e lo scambio dei dati energetici.

Audit Energetico – riguarda la progettazione e la realizzazione di una piattaforma informatica in grado di acquisire e mantenere le informazioni sui consumi energetici di generici utenti residenziali. Viene, quindi, messo a disposizione degli utenti una *web application* grazie alla quale è possibile inserire in maniera strutturata e puntuale tutti i dati necessari per effettuare una auto valutazione sull'efficienza energetica del proprio appartamento. In questa annualità sono stati perfezionati alcuni comportamenti sia statici che dinamici della piattaforma implementata precedentemente in modo tale da migliorarne l'efficacia e renderla ulteriormente *compliant* ai requisiti progettuali definiti.

La raccolta di entrambe le tipologie di dati di consumo permetto la loro successiva elaborazione ad alto livello per fornire indicazioni e benchmark di riferimento.

1 Introduzione

La collaborazione tra l'Università Roma Tre e il C.R. ENEA "La Casaccia" ha l'obiettivo di sviluppare tecnologie innovative e soluzioni informatiche in grado di supportare i processi di pianificazione strategica degli interventi di riqualificazione energetica degli edifici, sia residenziali che ad uso ufficio. Quindi, le due attività su cui si è concentrato il contributo di Roma Tre riguardano la realizzazione di strumenti software per implementare e sostenere una raccolta, standardizzata ed omogenea, dei consumi energetici. Questi software devono poter interagire e interoperare con i tool di monitoraggio e valutazione delle prestazioni dei servizi messi a disposizione da ENEA. In particolare, la ricerca è stata suddivisa in due linee progettuali principali, ossia il *District Database*, sulla raccolta automatica dei consumi energetici di un distretto universitario, e l'*Audit Energetico*, sulla raccolta dei consumi energetici residenziali. Gli sviluppi e i risultati ottenuti da queste due linee progettuali verranno presentati separatamente.

L'attività svolta all'interno della progettualità *District Database* ha avuto come scopo principale quello di gestire l'interscambio in tempo reale dei dati riguardanti i dispositivi energivori che compongono le *essential utilites* di un ambiente urbano. In particolare, si è voluto creare un applicativo informatico per la trasmissione automatizzata dei consumi energetici imputabili al cluster di edifici appartenenti a Roma Tre, e che sono sottoposti a monitoraggio, verso la piattaforma informatico centrale che gestisce questo ecosistema, ossia la Smart City Platform (SCP). E'stato, perciò, definito e realizzato un sistema in grado di acquisire i consumi energetici orari aggregati per edificio e di manipolare i dati per organizzarli in un formato adatto all'inserimento nel sistema SCP dell'ENEA. Deve, quindi, creare il messaggio in formato JSON [2] per essere compliant con le specifiche di comunicazione [3], e trasmettere il payload alla SCP sfruttando il protocollo di comunicazione *Message Queue Telemetry Transport* (MQTT) [4] attualmente adottato come standard ISO di messaggistica. Operativamente, il contributo è consistito nella realizzazione di un applicativo informatico in grado di accedere al database che contiene i consumi energetici di Roma Tre, organizzare i dati secondo le modalità di immissione nel sistema SCP, avviare ciclicamente un *publisher* (server) MQTT che, dopo aver organizzato il messaggio in un *Urban Dataset*, lo potrà inviare al sistema SCP sul quale risiede un *subscriber* (client) MQTT. Per fare ciò sono state sfruttate le potenzialità del linguaggio di programmazione ad oggetti Java [5], grazie al quale è stato implementato un *thread* informatico in grado di espletare le funzionalità sopramenzionate. Una descrizione più accurata della soluzione proposta e dei test effettuati su di essa è riportata nella sezione 2 del presente report.

L'attività svolta all'interno della progettualità *Audit Energetico* ha avuto come scopo principale quello di acquisire ed elaborare i dati energetici riguardanti i consumi energetici riconducibili alle abitazioni private. Si è, quindi, pensato di realizzare un portale web in cui gli utenti, su base volontaria, possono fornire tutte le informazioni utili e necessarie per individuare e segnalare le zone geografiche o gli edifici con maggiore necessità di interventi di riqualificazione energetica. L'utilizzo di questo strumento mira a rendere più semplice poter fornire un feedback all'utente che sia puntuale e mirato. Sfruttando valutazioni statistiche e considerando benchmark di riferimento, risulta più agevole segnalare quali sono le criticità su cui intervenire per migliorare il profilo energetico dell'abitazione. Inoltre, la piattaforma informatica permette di raccogliere e memorizzare in modo omogeneo i dati provenienti da tutti gli utenti e quindi di consentire una loro corretta elaborazione e valutazione statistica. E' stata quindi realizzata, in versione beta, una *web application* che prevede una sezione di gestione, *back-end*, che elabora e supervisiona la produzione dei contenuti, e una sezione applicativa, *front-end*, che gestire il processo di immissione e visualizzazione dei contenuti da parte dell'utente. Seguendo la procedura che un generico utente deve effettuare per accreditarsi al servizio e fornire le informazioni necessarie, verranno mostrati nella sezione 3 del presente report gli elementi che compongono la piattaforma con cui è possibile interagire. Per implementare le funzionalità richieste dalla piattaforma si è deciso di utilizzare tre linguaggi classici di programmazione, molto usati, e che, se ben maneggiati e coordinati consentono di definire l'aspetto e il comportamento dell'interfaccia grafica e, più in generale, della web application. Quindi è stata programmata in HTML [6] la struttura statica delle pagine, personalizzandola tramite il CSS [7] grazie al quale è possibile dotare la pagina di particolari colori, font, immagini di sfondo, ecc. Mentre, utilizzando il linguaggio Javascript [8] si è fornita

la pagina di tutti quei meccanismi e di tutte quelle azioni che permettono l'interazione dinamica da parte dell'utente.

2 District Database

L'obiettivo di questa progettualità è di definire e implementare un applicativo informatico in grado di mettere in comunicazione il database dell'Università Roma Tre, contenente i dati riguardanti il consumo energetico degli edifici di Ateneo, con l'aggregatore presente al centro di ricerca ENEA La Casaccia. Considerando gli standard definiti nel progetto PELL, si vuole, quindi, creare un accesso al database dell'Università per prelevarne i dati energivori e memorizzarli in formato JSON nel database ausiliario NoSQL orientato agli oggetti e, contestualmente inviarli alla Smart City Platform dell'ENEA. Si è deciso di adottare una strategia che comportasse un intervento minimamente invasivo sul preesistente sistema di gestione energetica operativo sul cluster di edifici di Roma Tre, e che mantenesse un grado di complessità di implementazione basso. Di seguito viene riportato il contesto applicativo in cui si è operato e si descrive l'implementazione e le funzionalità del thread informatico realizzato per effettuare lo scambio di dati tra Roma Tre e la SCP.

2.1 Strumenti e standard SCP

Per poter coordinare al meglio le funzionalità che deve svolgere il processo automatizzato di invio dati è necessario analizzare la struttura informatica e logica che caratterizza la SCP implementata in ENEA. Dallo studio della documentazione associata al rilascio [9], emerge che si tratta di una piattaforma orizzontale che aggrega e integra i dati di piattaforme verticali indipendenti tra loro e che si occupano del monitoraggio di uno specificato asset. Per poter strutturare un canale di comunicazione tra Roma Tre e la SCP che sia stabile e il più possibile esente da malfunzionamenti è stato necessario che l'applicativo informatico utilizzato per lo scambio dati fosse compliant con gli standard definiti da ENEA.

2.1.1 Smart City Platform

La Smart City Platform è costituita da una piattaforma centrale di governance chiamata *inter-Smart City Platform*, o *inter-SCP*, in grado di stabilire e mantenere una comunicazione interoperabile con le varie istanze di SCP che agiscono su scala urbana. L'i-SCP permette di monitorare su scala nazionale i dati urbani energetici, osservandone e confrontando i valori, offrendo, quindi, molteplici vantaggi. Operativamente, si attiva l'inter-SCP a raccogliere dati dalle varie SCP e, contestualmente, si abilitano le SCP a raccogliere attivamente i dati dalle Solution verticali urbane, il tutto utilizzando il protocollo MQTT.

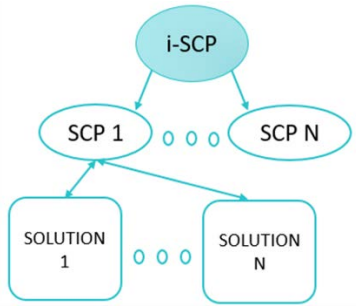


Figura 1 - Schema logico del progetto Smart City Platform

In questa progettualità si è sfruttata la piattaforma urbana SCP Smart Village Casaccia con la quale è stato possibile implementare la comunicazione. Per fare ciò è necessario sfruttare le funzionalità di un *SCP-Bridge* che mette a disposizione dei moduli software, denominati *services*, i quali possono essere utilizzati per implementare molti scenari di comunicazione. I services possono eseguire diverse azioni, quali la lettura e scrittura, locale o remota, di *UrbanDataset*, e anche l'implementazione di semplici operazioni di DataFusion. L'esecutore, o scheduler, di questi services, è chiamato *SCP-Scheduler* in grado di registrare, avviare, fermare, monitorare e gestire con periodicità predefinita i vari software applicativi.
Una volta che i dati vengono elaborati dalla inter-SCP, essi potranno essere visualizzati attraverso la *SCP-Dashboard*, che mostra, tramite tabelle e diagrammi, i dati nazionali ed urbani relativi al caso cittadino o distrettuale. La SCP, quindi, non è solo uno strumento per il monitoraggio dell'intero distretto, ma anche un broker per l'interscambio di dati e informazioni tra piattaforme verticali, tra le quali si evidenzia un'interdipendenza. I componenti chiave relativi alla SCP, di interesse per questa attività, sono:

- *Solution verticale*, che corrisponde ad una piattaforma locale, ossia ad un particolare contesto applicativo nella città o nel quartiere. Il modello di Solution descrive un insieme di caratteristiche comuni che permettono di individuare dei processi per la gestione dei dati.
- *Comunicazione interoperabile* tra i sistemi, basata sullo scambio di dati eterogenei, e benché siano diversi tra loro, sono comunque caratterizzati da una collocazione spazio-temporale ben precisa e che li distingue.
- *Urban Dataset (UD)*, che riporta le informazioni di stato della piattaforma da cui è esportato. Operativamente, un Urban Dataset viene creato da una Solution tramite le seguenti fasi:
    1. Esportazione dei dati dal proprio database locale per poi aggregarli nell'UD configurato;
    2. Rappresentazione dell'UD secondo il formato JSON o XML;
    3. Utilizzo del Web Service UDG per inviare l'UD alla SCP;
    4. Ricezione da parte della SCP dell'UD e immagazzinamento nella banca dati.

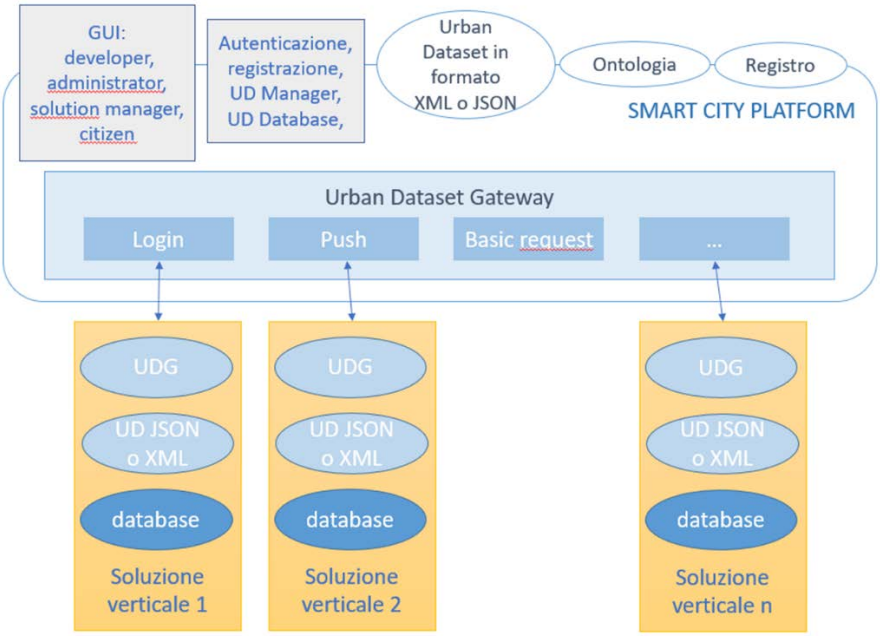


Figura 2 - Elementi che compongono l'ecosistema SCP

La definizione centralizzata degli Urban Dataset viene data da ENEA, in particolare dal laboratorio dell'C.R. ENEA CROSS, che analizza, certifica e registra tutto ciò nell'*Ontologia* centrale del progetto PELL. Il template dell'UD è disponibile in vari formati tra i quali, come già detto, il formato JSON che verrà utilizzato nella fase di esportazione. Quindi, riassumendo, l'Urban Dataset è composto da:

- Una descrizione semantica chiara, definita con l'Ontologia;
- Un modello dati astratto;
- Un'implementazione sintattica in JSON o XML;
- Un servizio di trasporto dati, tramite il suo Gateway.

Il servizio Web dell'*Urban Dataset Gateway*, in acronimo UDG, è formato da:

- Tre pattern: PUSH, Request/Response, Publish/Subscribe;
- Un'interfaccia comune che prevede i metodi: Login, Push, BasicRequest, SearcingRequest;
- Un'implementazione REST dell'interfaccia per trasmettere in formato JSON e una WDSL per trasmettere in formato XML.

Ognuno di questi elementi è stato sfruttato in fase di implementazione del processo automatico per supportare l'invio ciclico dei dati energetici degli edifici di Roma Tre secondo quelli che sono gli standard previsti per la comunicazione interoperabile con la SCP.

2.1.2 Soluzione implementata

La soluzione proposta consiste nell'implementare un processo ciclico, chiamato *Data Exchange Manager*, o DEM, che acceda al database energetico locale di Roma Tre, ne estragga le informazioni di interesse, per poi assemblarle a seconda degli standard della Smart City Platform, e, successivamente, invii il messaggio contenente le informazioni al database di ENEA tramite l'Urban Dataset Gateway. Per realizzare l'applicativo software capace di effettuare lo scambio di Urban Dataset tra il contesto di Roma Tre e la SCP si è deciso di procedere secondo step successivi:

1. **Analisi dell'UB che si vuole implementare e del *Modello Dati Astratto* (MDA)**
   si è ricercato tra gli UD esistenti in Ontologia individuando l'UD *Builging Elettric Consumption* [10] che risulta adeguato agli scopi prefissati. L'MDA, che ha l'obiettivo di aiutare la mappatura delle strutture che comunicano con la SCP e di tradurre i messaggi destinati all'UD, è composto da tre parti principali:
   - *Specification*: contiene le informazioni riguardanti le proprietà che caratterizzano e definiscono il particolare UD utilizzato. In particolare, si possono distinguere i seguenti campi:
     a) *specificationReferences*, i riferimenti che consentono l'identificazione della specifica, quali: la versione della specifica, *version*; il codice identificativo della specifica, *id*; il nome associato all'UB, *name;* e, infine, l'url associato alla specifica.
     b) *propertyDefinition*, un insieme di proprietà, sia elementari sia aggregate, che caratterizzano l'UD. Una proprietà elementare viene definita da: nome, *propertyNeme*; descrizione, *propertyDescription*; tipo di dato primitivo con cui viene espresso il valore, *dataType;* unità di misura del dato, *unitOfMeasure*.
   - *Context*: fornisce le informazioni che contestualizzano i valori trasmessi. In particolare, il blocco prevede le seguenti informazioni:
     a) *producer*, riferimento alla piattaforma che produce i dati tramite l'identificatore univoco del sistema, *id*, e l'identificativo dello schema di riferimento, *schemeID*;
     b) *timeZone*, fuso orario della zona di interesse;
     c) *timeStamp*, istante di generazione del particolare UD;
     d) *coordinates*, coordinate WGS84 del centro geometrico del sistema di monitoraggio, espresse tramite tre campi, ossia latitude, longitude e height;
     e) *language*, per la lingua utilizzata;
   - *Values*: è composto da una o più righe, dove ad una riga corrisponde un gruppo di proprietà, ognuna contenente le seguenti informazioni:
     a) *id*, identificatore della riga;
     b) *description*, descrizione testuale dei valori monitorati;
     c) *coordinates*, coordinate geografiche dell'edificio a cui sono riferiti i dati rilevati;
     d) *period*, periodo temporale di acquisizione dei dati definito da un tempo di inizio, *tStart*, e un tempo di fine, *tEnd*
     e) property, stabilisce come definire i valori delle proprietà introdotte, ossia: il nome identificativo della proprietà, *name*; il valore acquisito, *value*.

   In base al MDA si definisce il contenuto e la struttura dei messaggi scambiati tramite UD per stabilire quali proprietà specificare. Quindi, deve essere effettuata un'operazione di mapping tra Modello Dati e le informazioni rese disponibili dal sistema di gestione energetica degli edifici di Roma Tre. Questo è stato necessario per definire come devono essere popolati i messaggi creati a partire dal Modello Dati scelto e mappare ogni elemento del Modello con un campo della sorgente dati. Questa deve essere facilmente rintracciabile all'interno del sistema di archiviazione.
2. **Implementazione della struttura di archiviazione ausiliaria**
   Si è definita e implementata la struttura del database ausiliario sfruttando le potenzialità di MongoDB [11], ossia un database non relazionale basato su documenti, che premette di avere tempi di accesso rapidi in lettura e scrittura, una gestione automatica dello sharding e della

ridondanza della memoria. E' stata prevista una sola collezione i cui documenti memorizzati possono avere la seguente struttura:

- *Buildings*, contenente le informazioni anagrafiche dell'edificio, i campi sono:
  - *Building_id*, codice identificativo dell'edificio;
  - *Building_name*, nome dell'edificio;
  - *Latitude*, latitudine geografica dell'edificio;
  - *Longitudine*, longitudine geografica dell'edificio;
  - *Height*, altitudine sul livello del mare dell'edificio.
- *Measurements*, contiene le informazioni riguardanti la grandezza misurata, i campi sono:
  - *Measurement_id*, codice univoco della grandezza misurata;
  - *Measurement_description*, tipo della grandezza misurata;
  - *Measurement_type*, unità di misura della grandezza misurata;
  - *Measurement_primitive*, tipo di dato primitivo con cui è memorizzato il valore;
- *Data*, contenente le informazioni sui dati prelevati, si suddivide in due categorie di campi, quelli storici, memorizzati in un array, e quelli attuali:
  - *Historian*, di tipo array, contenente vari oggetti ognuno dei quali con i seguenti campi:
    - *Start_acquisition*, momento in cui si inizia ad acquisire il dato,
    - *End_acquisition*, momento in cui si finisce di acquisire il dato,
    - *Value*, valore del dato acquisito.
  - *Actual*, un oggetto contenente i seguenti campi:
    - *Start_acquisition*, data e ora di inizio acquisizione del dato attuale;
    - *End_acquisition*, data e ora di fine acquisizione del dato attuale;
    - *Value*, valore del dato attuale acquisito.

Risulta ora evidente come ad ogni edificio di Roma Tre deve essere, quindi, associato un documento, ossia un record del database, all'interno del quale sono riconoscibili due tipi di dati:

- Immutabili, per quanto riguarda i dati anagrafici dell'edificio e delle grandezze misurate;
- Real time, per il valore assunto dalla grandezza e il periodo temporale associato ad ogni acquisizione, motivo per il quale, viene utilizzato un array per memorizzare le acquisizioni passate.

L'implementazione del database ausiliario è avvenuta sfruttando le funzionalità messe a disposizione dalla GUI grafica *MongoDB Compass* [12], versione 8.0.23, instaurando una connessione in localhost. E' stato scelto questo particolare database non relazionale poiché risulta essere largamente utilizzato per memorizzare, gestire e interrogare grandi quantità di dati attraverso uno schema estremamente flessibile. Oltre a questo, utilizza il formato JSON per l'archiviazione dei documenti, il che ha reso più efficiente la creazione del messaggio da inviare alla SCP.

3. Implementazione del processo informatico

Il DEM deve svolgere ciclicamente una serie di attività sequenziali con l'obiettivo di interrogare la struttura di archiviazione di Roma Tre, di creare gli UD, e di inviare il messaggio alla SCP. In particolare, le azioni implementate in ordine temporale nel processo informatico sono:

a) Acquisizione dei dati energetici dal database di Roma Tre
b) Trasformazione dei dati in formato JSON per essere compatibili con gli standard della Smart City Platform;
c) Aggregazione dei dati all'interno di un Urban Dataset associato alla Solution;
d) Configurazione dei parametri per la trasmissione dati all'Urban Dataset Gateway;
e) Invio dell'Urban Dataset creato al sistema Smart City Platform tramite il Web Service Urban Dataset Gateway utilizzando il protocollo MQTT;
f) Memorizzazione dei dati acquisiti all'interno del database ausiliario, discriminando tramite un flag i dati trasmessi con successo all'Urban Dataset Gateway.

Il Data Exchange Manager, o DEM, dovrà, quindi, accedere in lettura al database energetico di Roma Tre e in lettura e scrittura al database ausiliario.

Figura 3 - Passi algoritmici che esegue il DEM

Per le funzionalità di quest'applicazione si è deciso di sviluppare il processo comunicativo tra le piattaforme di Roma Tre ed ENEA, tramite il noto linguaggio di programmazione Java, ed utilizzando come ambiente di sviluppo Eclipse [13]. I principali motivi per cui è stato deciso di utilizzare questo linguaggio di programmazione risiedono nel fatto che esso è estremamente performante e che risulta essere: semplice robusto e sicuro; indipendente dalla piattaforma in cui verrà utilizzato; progettato per eseguire codice da sorgenti remote prevedendo strumenti e librerie per il networking.

2.2 Applicativo implementato

Per questa attività, in base alla quantità e alla tipologia di dati che si devono inviare, è stata prevista una Solution verticale di test per un edificio di Roma Tre afferente al Dipartimento di Ingegneria. Il processo di configurazione della comunicazione con la SCP ha previsto diversi passaggi riportati di seguito in sequenza logica-temporale:

1. E' stata formalizzata la collaborazione tra i due Enti interessati accordandosi sulla possibilità di avviare il flusso di dati. Roma Tre ha richiesto la creazione di un'istanza della SCP dedicata.
2. E' stata creata da ENEA l'istanza della SCP richiesta ed è stato definito l'utente Administrator.
3. Roma Tre si è occupata di individuare e configurare l'Urban Dataset da inviare e la Solution da utilizzare.
4. Roma Tre ha progettato e implementato l'applicativo informatico in grado di inviare, in modo automatico, i consumi energetici del proprio cluster di edifici alla SCP Casaccia. Il processo informatico risiede e vive all'interno di una macchina appartenente all'intranet di Roma Tre.
5. E' stata avviata la fase di test della comunicazione interoperabile, tramite l'Urban Dataset Gateway, tra i due Enti.

Al fine di sviluppare lo scambio di dati tra il database di Roma tre e la SCP, si è implementato un programma, attraverso il linguaggio Java, utilizzando come IDEE Eclipse. Per svolgere e testare le funzionalità del programma, sono stati utilizzati due software quali: MongoDB Compass, nella versione 1.29.4, e MySQL Workbench [14], nella versione 8.0.23. Questi ultimi facilitano la creazione, la gestione e svariate altre operazioni sui database, anche in locale. Nel programma, sono stati previsti metodi per la connessione e la disconnessione ai due database, metodi per la creazione di un documento, e molti altri, ognuno dei quali ha una precisa gestione delle eccezioni, tutto ciò all'interno di un'unica classe detta *Connessioni*. Per le varie proprietà, quali nome e password dei database e nomi delle tabelle relazionali, ed altro come si vedrà in seguito, è stato previsto l'utilizzo di un file *properties*. I dati vengono prelevati dal database relazionale, attraverso la query effettuata tramite l'operazione di JOIN sulle tre tabelle, e memorizzati i differenti valori in variabili globali. Viene poi creato, per ogni nuova acquisizione, un documento in formato JSON, e memorizzato all'interno della collezione Edificio del database non relazionale. Se da quell'edificio è già stato precedentemente prelevato un dato energetico, si aggiorna semplicemente il documento a lui relativo. Inoltre, per inviare, attraverso il protocollo MQTT, un messaggio alla SCP in formato JSON, contenente i valori effettivi prelevati dal database relazionale e inseriti all'interno

dell'Urban Dataset messo a disposizione dall'ENEA, è stato implementato il codice all'interno del metodo *run()*. Si procede riportando alcuni dettagli di realizzazione.

2.2.1 Variabili globali definite

Per chiarezza, si riportano di seguito le variabili globali utilizzate all'interno del programma e suddivise in:

- Variabili per la gestione ottimale del flusso del programma:
  - *Delay*, int, intero che indica il tempo dopo il quale si controlla nuovamente la misura;
  - *TempoCiclo*, int, intero che indica il tempo dopo il quale si può instaurare una nuova connessione e prelevare dei nuovi dati;
  - *JsonDoc*, di tipo Document, contenente il riferimento al documento JSON creato;
  - Token, di tipo String, memorizza il token relativo all'UD di riferimento;
  - *PropertiesPath*, che memorizza il nome del file di configurazione delle proprietà;
- Variabili per gestire l'interazione con il database energetico di Roma Tre:
  - *ServerSQLDBname*, contenente il nome del database relazionale;
  - *ServerSQLPath*, che identifica il path del server;
  - *ServerSQLUser*, memorizza l'username nei per accedere al database relazionale;
  - *ServerSQLPassword*, memorizza la password per accedere al database relazionale;
  - *UrlSQL*, utilizzato per definire l'url di connessione al database relazionale;
  - *Acquisizioni*, per memorizzare il nome della tabella del database relazionale che riguarda le acquisizioni energetiche;
  - *Edificio*, per memorizzare il nome della tabella del database relazionale che riguarda gli edifici delle varie acquisizioni;
  - *Grandezza*, per memorizzare il nome della tabella del database relazionale che riguarda le grandezze delle acquisizioni effettuate;
  - *DbSQLstConnection*, di tipo Connection, utilizzata per instaurare la connessione al database relazionale;
  - *DbSQLstStatement*, di tipo Statement, utilizzata per eseguire la query SQL;
  - *RsMain*, di tipo ResultSet, memorizza il risultato della query SQL.
- Variabili globali utilizzate per acquisire il valore della misurazione energetica prelevata e inserita nel resultSet, così che l'informazione resti in memoria in un'esecuzione del programma:
  - *M_val*, di tipo double, per memorizzare il valore della misura acquisita;
  - *M_tIn*, di tipo TimeStamp, per memorizzare il tempo di inizio acquisizione;
  - *M_tFn*, TimeStamp, per memorizzare il tempo di fine acquisizione;
  - *Id_m*, int, per l'id identificativo della misura;
  - *Id_e*, int, per l'id identificativo dell'edificio;
  - *Id_g*, int, per l'id identificativo della grandezza relativa all'acquisizione;
  - *G_nome*, stringa, per il nome della grandezza dell'acquisizione;
  - *G_unit*, stringa, per l'unità di misura dell'acquisizione;
  - *G_prim*, stringa, per la primitiva della misura dell'acquisizione;
  - *E_nome*, stringa, per il nome dell'edificio da cui proviene l'acquisizione;
  - *E_alt*, double, per l'altitudine dell'edificio;
  - *E_lat*, double, per la latitudine dell'edificio;
  - *E_lon*, double, per la longitudine dell'edificio.
- Variabili per gestire l'interazione con il database ausiliario:
  - *ServerMongoDBname*, per il nome del database l'ausiliario;
  - *ServerMongoUser*, per il nome dell'utente che accede al database;
  - *ServerMongoPassword*, memorizza la password d'accesso;
  - *ServerMongoIp*, memorizza l'indirizzo IP per accedere al database ausiliario;
  - *UrlMongo*, per definire l'url di connessione al database ausiliario;
  - *NomeCollectionEdificio*, per memorizzare il nome della collezione degli edifici di Roma Tre.
  - *MongoClient,* di tipo MongoClient, contenente il riferimento alla connessione del database mongoDB;

- *MongoDB*, di tipo MongoDatabase, contenente il riferimento al database;
- *RiferimentoCollection*, di tipo MongoCollection<Document>, che contiene il riferimento alla collezione edificio del database;

2.2.2 File properties

Il file properties è un semplice file di tipo testuale, con delle parole chiave che vengono riconosciute dal programma, alle quali viene associato un valore che il programma acquisisce. È così più semplice modificare nomi, o password, o indirizzi, poiché non è necessario modificare nulla all'interno dell'implementazione del codice. Sono stati quindi valutati i livelli di sicurezza, in quanto, ogni accesso ai database è vincolato dall'Username e dalla Password. Sono inoltre presenti i tempi di attesa e di ciclo di 15 minuti, facilmente modificabili, ed anche i nomi relativi alle tabelle del database relazionale dell'Università ed il nome della collezione presente all'interno del database ausiliario. Segue esempio del file inserito all'interno del progetto, i valori in futuro potrebbero cambiare.

```
# File properties associato al progetto ComunicazioneRomaTre_SCP

serverDBRomaTreName = romatredb
serverDBRomaTreUser = root
serverDBRomaTrePassword = Creative1234!
serverDBRomaTrePath =

acquisizioniTab = acquisizioni
edificioTab = edificio
grandezzaTab = grandezza

serverMongoDBname = mydb
serverMongoUser =
serverMongoPassword =
serverMongoIp = localHost
serverMongoPort = 27017

nomeCollezioneEdifici = edificio

delay = 10000
tempoCiclo = 60000
```

Figura 4 - Esempio di file properties utilizzabile

Il costruttore all'interno del programma che sfrutta le informazioni prelevate dal file properties prende il nome della classe, *Connessioni(String pPath)*, e riceve in input una stringa relativa al file. Al suo interno viene creato un oggetto *Properties()*, e successivamente assegnata, alla variabile globale *propertiesPath*, la stringa che viene passata come parametro. Vengono, quindi, assegnati i valori prelevati dal file properties alle variabili globali. In alternativa al precedente costruttore, ne viene utilizzato un altro, al fine di verificare il corretto funzionamento in locale. Esso non necessita di nessuna stringa passata come parametro, ed assegna dei valori prestabiliti alle variabili globali.

2.2.3 Il metodo MAIN

Al fine di rendere chiaro e semplice il metodo *main()*, e per sfruttare le potenzialità della programmazione su thread, si è utilizzato il metodo *run()*, descritto più avanti, che conterrà tutti i metodi che consentono il corretto funzionamento del programma. Nel corpo del metodo *main()* viene innanzi tutto stampato un messaggio di "Avvio scambio dati", successivamente viene creato un riferimento *cn* di tipo connessioni. Se è assente il file properties, viene creato un oggetto invocando il costruttore che imposta dei valori prestabiliti. Se invece è presente il file properties, viene invocato l'altro costruttore che ne fa uso. Infine, viene avviato, con il comando *start()*, il metodo *run()*, contenente le istruzioni sul thread che eseguirà il processo.

2.2.4 Il metodo RUN

Il metodo *run()*, contenuto nella classe Thread, viene chiamato se il thread è stato costruito utilizzando un oggetto di tipo *Runnable*, e viene poi eseguito il codice specificato al suo interno. L'interfaccia Runnable fornisce maggior flessibilità, in quanto, il thread diventa una sottoclasse di qualsiasi altra classe. Questo metodo può essere chiamato più volte, o usando il metodo *start()* o invocandolo semplicemente con il suo nome, ma, in quest'ultimo caso, non è prevista la restituzione di alcun valore.

Oltre al metodo *start()*,la classe Thread offre altri vari metodi per il controllo, quali:

- *stop()*, per forzare la terminazione dell'esecuzione di un thread, e quindi, le risorse utilizzate dal thread vengono subito liberate;
- *suspend()*, per bloccare l'esecuzione di un thread, ma non libera le risorse impegnate;
- *resume()*, per riprendere l'esecuzione di un thread precedentemente sospeso, il thread riattivato ha una priorità maggiore di quello correntemente in esecuzione;
- *sleep(long t)*, per bloccare per un tempo specificato l'esecuzione di un thread;
- *join()*, per bloccare il thread chiamante in attesa della terminazione del thread di cui si invoca il metodo;
- *yield()*, per sospendere l'esecuzione del thread invocante, lasciando il controllo della CPU agli altri thread in coda d'attesa.

L'esecuzione del metodo *run()* all'interno della classe Connessioni, invoca l'esecuzione di ulteriori metodi, al fine di eseguire, in ordine temporale, le seguenti azioni:

1. **Connettersi al database relazionale di Roma Tre**, attraverso il metodo *connessioneDBRomaTre()*
   Java può effettuare azioni CRUD su un database MYQSL sfruttando la libreria *mysql-connector-java-8.0.26*, disponibile open source e inserita nel building path del progetto.
2. **Interrogare il database relazionale** attraverso il linguaggio SQL
   Viene implementata una stringa leggibile nel linguaggio SQL, che rappresenta la query necessaria per ottenere i dati di interesse contenuti nel database SQL. L'interrogazione restituirà un *resultSet*, ossia un oggetto con struttura tabellare creato appositamente per prelevare i dati dai database relazionali. Per acquisire il resultSet, viene inizializzato un oggetto *statement* sulla connessione. Nel caso in cui i dati sono assenti, il resultSet non acquisirà nessun nuovo dato, sarà stampata in output la stringa "resultSet = empty".
3. **Acquisire i dati**, memorizzando le informazioni in variabili globali
   Il resultSet ottenuto a seguito dell'interrogazione SQL al database relazionale contiene la nuova acquisizione energetica dell'edificio dell'Università. L'ordine dei campi del resultSet dipende dall'interrogazione implementata, ossia, da quali attributi vengono selezionati ed ottenuti dalla query sulle tabelle del database relazionale. Vengono quindi ordinatamente associati i valori ottenuti nel resultSet alle variabili globali elencate in precedenza.
4. **Disconnettersi dal database di Roma tre**, attraverso il metodo *disconnessioneDBRomaTre()*
   Se la connessione è stata effettivamente instaurata, allora, viene chiusa e viene stampato il messaggio di avvenuta chiusura. Nel caso in cui si dovessero presentare malfunzionamenti verrà generata un'eccezione.
5. **Connettersi al database l'ausiliario** MongoDB, attraverso il metodo *connessioneClientMongoDB()*
   Java può effettuare azioni CRUD su un database MongoDB sfruttando la libreria mongo-java-driver-3.4.2, disponibile open source e inserita nel building path del progetto. In particolare, viene creato un oggetto di tipo *Logger* che si occupa di inizializzare le funzionalità di gestione dei database MongoDB, permettendo la connessione tramite l'indirizzo IP ed il numero di porta associati al database. Viene poi preso il riferimento per il database e per la collezione che devono essere acceduti.
6. **Creare o aggiornare un documento del database ausiliario**, attraverso il metodo *creaDocumento()*
   Il metodo agisce direttamente sulla collezione Edificio del database MongoDB. In particolare, si verifica che nella collezione sia presente un documento che abbia nel campo *id_edificio*, il valore corrispondente a quello della nuova acquisizione. In caso di verifica positiva dell'identificativo dell'edificio, il documento viene aggiornato, altrimenti ne viene creato uno nuovo con i dati della nuova acquisizione.
7. **Disconnettersi dal database ausiliario**, attraverso il metodo *disconnessioneClientMongoDB()*
   Come per il precedente metodo di disconnessione dal database relazionale, anche in questo caso, se la connessione è stata effettivamente instaurata, allora viene chiusa e viene stampato il messaggio di avvenuta chiusura connessione. Nel caso in cui si dovessero presentare malfunzionamenti, viene lanciata un'eccezione.

8. **Creare il documento JSON da inviare alla SCP**, attraverso il metodo *creaJson()*
   La definizione del formato dati stabilisce una lingua comune fra i diversi fornitori di dati e le piattaforme municipali, in termini di modello astratto e sintassi. La semantica del contenuto deve, quindi, essere conforme alla specifica SCPS Semantic, definita nell' Ontologia del progetto PELL, mentre la sintassi, che può essere o in formato JSON o XML, deve essere valida rispetto alla specifica SCPS Information [15].
9. **Controllare la comunicazione con la SCP**
   Verifica la sussistenza del token associato alla particolare Solution all'interno della SCP, attraverso l'invocazione dei metodi *isAlive()*, *test()*, e *login()* forniti dal client sviluppato dall'ENEA.
10. **Inviare il documento JSON** creato all'SCP, attraverso il metodo *push()*
    Il pattern del metodo push, disponibile nel client sviluppato dall'ENEA, prevede l'invio di un Urban Dataset da parte del Client al Server, il quale riceve l'Urban Dataset, lo processa e risponde con un ack al Client. La connessione, quindi, resta aperta fino a quando il Server invia l'ack, il Client lo riceve e chiude il canale di comunicazione.

Queste operazioni vengono eseguite ciclicamente grazie ad una condizione while, contenete una variabile detta *flagRun*, sempre verificata, la quale permette l'attivazione del processo ogni 15 minuti.

2.2.5 Gestione delle eccezioni

Varie sono le eccezioni che vengono raccolte e altrettanti sono i messaggi ad esse relativi. Per ogni funzionalità o metodo eseguito all'interno della classe principale del processo informatico, vengono catturate le eccezioni associate. Di seguito si riporta un elenco delle eccezioni riconosciute, segnalate attraverso la stampa di un messaggio a video, e gestite attraverso l'arresto del processo automatico.

- ClassNotFoundException "ClassNotFoundException - JDBC driver download failed"
- *SQLException*, per quanto riguarda la comunicazione con il database SQL, che si dividono in:
  - "SQLException - connection Roma Tre failed";
  - "SQLException - main query in run() failed";
  - "SQLException - data acquisition from Roma Tre in run() failed";
  - "SQLException - disconnection Roma Tre failed";
- *MongoException*, per quanto riguarda la comunicazione con il database No-SQL che si dividono in:
  - "MongoException - connection MongoDB failed";
  - "MongoException - insert or update documents in MongoDB in run() failed";
  - "MongoException - disconnection MongoDB failed";
- *JsonDocumentException*, per quanto riguarda la creazione e l'invio del JSON, che si dividono in
  - "JsonDocumentException - Json creation failed";
  - " JsonDocumentException – Json sending to SCP failed".

2.3 Test funzionali

In questo paragrafo del report vengono descritte le fasi salienti riguardanti la fase di testing effettuata per verificare l'efficacia e l'efficienza dell'applicativo informatico implementato. In particolare, sono state testate tutte le varie funzionalità che deve garantire il programma. In un primo luogo, si è deciso di approfondire le attività di connessione al database relazionale, di acquisizione dei dati attraverso la query in SQL, e di disconnessione dal database relazionale. Terminata questa prima valutazione, si è proceduto ad effettuare un secondo test utilizzando il tool MongoDB Compass per verificare il comportamento dell'applicativo e l'effettiva connessione/disconnessione al database MongoDB in locale. La terza tipologia di test effettuati ha avuto l'obiettivo di verificare la corretta creazione del documento JSON, secondo il formato dell'*Urban Dataset 2.0* definito dall'ENEA. Il file JSON corrisponde al contenuto del messaggio inviato tramite protocollo MQTT alla SCP, perciò, le informazioni che contiene devono avere una corrispondenza uno a uno con le informazioni previste nel template *BuildingElectricConsumption*. Come ultimo test è stata osservato il comportamento ciclico dell'applicativo che deve essere in grado di gestire il collegamento con la SCP e inviare l'UD.

2.3.1 Collegamento e acquisizione dati dal database energetico

Per testare queste funzionalità dell'applicativo è stato deciso di utilizzare una copia del database energetico di Roma Tre. La copia fedele in termini di strutturazione del database e delle sue tabelle è stata riprodotta in locale nel PCserver su cui risiede anche l'applicativo implementato. Questo per evitare che eventuali malfunzionamenti che potevano presentarsi durante questa fase avrebbero avuto un impatto sul database energetico dell'Ente.

La verifica sul comportamento è avvenuta attraverso l'inserimento di dati pseudoreali nel database relazionale implementato in MySQL Workbench tenendo in considerazione lo schema tabellare che rappresenta la realtà del database dell'Università Roma Tre. La connessione e la disconnessione di quest'ultimo è avvenuta con successo e, tramite un metodo di stampa su console, sono stati visionati i dati effettivamente inseriti, come evidenziato nella figura che segue.

```
Avvio scambio dati
Missing configuration file's path. Procedure enabled by default value.
============================================================
Check started activation 1 - waiting settling time
Connessione con Roma Tre avvenuta con successo!
Chiusura connessione con Roma Tre avvenuta con successo!
Connessione con MongoDB avvenuta con successo!

Documenti contenuti in collezione Edificio:
Document{{_id=619ba99033f0ac2448fb6c23, Building=Document{{Building_id=1,
Document{{_id=619caaea3970266080012fed, Building=Document{{Building_id=11

Chiusura connessione con MongoDB avvenuta con successo!
```

Figura 5 - Stampa su video per il controllo delle connessioni con i databases

2.3.2 Collegamento e scrittura dati sul database ausiliario

Sfruttando l'IDE messa a disposizione dal tool open source MongoDB Compass è stato possibile valutare immediatamente la gestione del collegamento con il database, figura 5, e la bontà delle operazioni di inserimento e aggiornamento. In particolare, è stato verificato che un nuovo documento viene creato solo se si verificano due casi: il primo è che la collezione sia inizialmente vuota, figura 6a; il secondo è che la collezione non contenga già un documento contenente l'id dell'edificio associato alla nuova acquisizione energetica, figura 6b.

a. b. c.

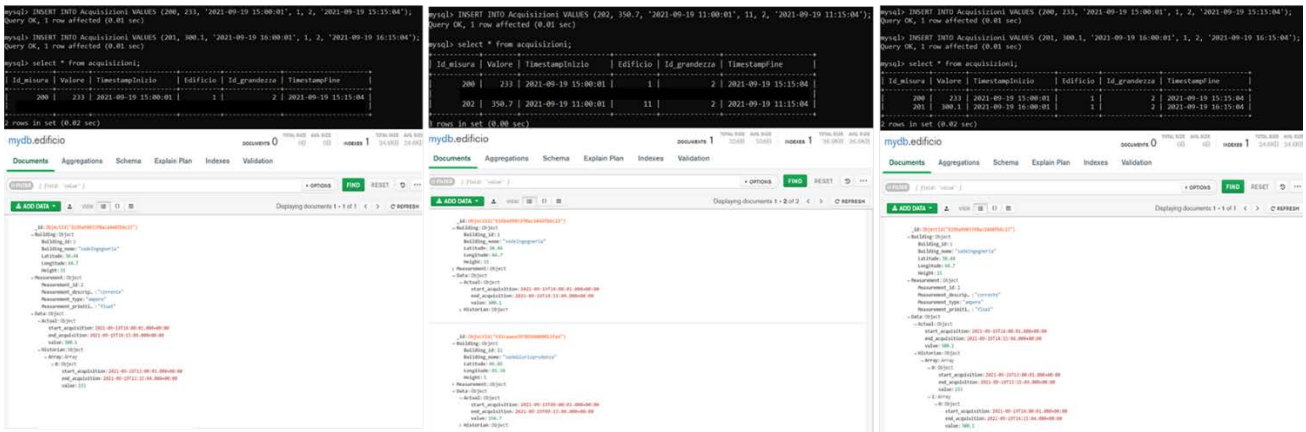


Figura 6 - a) creazione di un documento associato all'edificio 1; b) creazione di un documento associato all'edificio 11; c) aggiornamento dei valori riguardanti l'edificio 1

Per quanto riguarda l'aggiornamento del documento, che si ricorda deve essere effettuato qualora venga trovato il documento con l'identificativo dell'edificio corrispondente a quello della nuova acquisizione, è stato verificato inserendo nel database relazionale due acquisizioni provenienti dallo stesso edificio. Attraverso due esecuzioni del programma lo stesso documento è stato modificato, quindi aggiornato, figura 6c. La struttura di un documento memorizzato nel database relazionale MongoDB è riportata nella figura seguente, e si riferisce al binomio creato edificio-grandezza.

Figura 7 - Struttura di esempio di documento memorizzato in MongoDB

2.3.3 Creazione del JSON

Sfruttando le funzionalità della libreria *JsonObject* disponibile open source è stato creato l'oggetto JSON che corrisponde al corpo dell'UD inviato alla SCP. E' stato, perciò, verificato che le informazioni vengano strutturate presentando una corrispondenza uno a uno con le informazioni previste nel template *BuildingElectricConsumption (BEC)* sopramenzionato. In particolare, si è osservato come nel UD-BEC, vi è una sezione standard che deve essere riportata nello stesso modo all'interno del documento, ed una sezione contente i valori che di volta in volta devono essere aggiornati.

Figura 8 - Stampa pretty del JSON creato

La validazione è avvenuta analizzando le stampe su console aggiunte per questa fase di test ottenute grazie ad un metodo definito *ad hoc*. Nella figura 8 è riportata la stampa *pretty* dell'oggetto JSON creato.

2.3.4 Collegamento e invio dell'UD alla SCP

Secondo le best practice riportate dall'ENEA [3], è consigliato utilizzare quattro metodi per connettere l'utente all'Urban Dataset Gateway. Ognuno di questi metodi prevede un array di ritorno contenente l'esito dell'operazione richiesta. Nella figura 9 sono riportati: nella parte superiore lo schema utilizzato per l'array di ritorno e nella parte inferiore due esempi di risposta che si possono ottenere invocando il *login REST method*: a sinistra si può notare come l'operazione abbia successo, il contrario a destra. Tutti i casi di errore, per tutti i vari metodi offerti nella nuova libreria fornita dall'ENEA, sono stati individuati e catalogati in una lista, per le diverse categorie, con i codici ed i relativi messaggi, così che il client possa riconoscere, interpretare e gestire le risposte del web service.

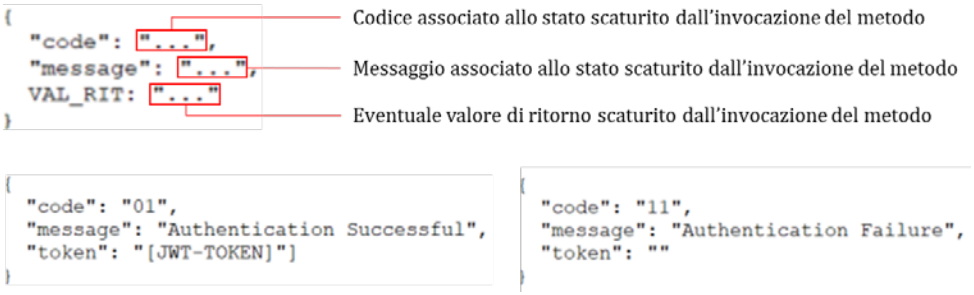


Figura 9 - Struttura ed esempi di ritorno della WS

I metodi hanno un preciso ordine nel quale devono essere richiamati:

1. metodo *test*: verifica se il Web Service (WS) SCPCasaccia è attivo. Se fallisce una prima volta, deve essere invocato nuovamente.
2. metodo *isAlive*: per capire se il *JSON Web Token* (JWT), ricevuto precedentemente è ancora valido. In caso affermativo si deve saltare il punto 3 e procedere con il punto 4.
3. metodo *login*: salva localmente il token JWT ricevuto nella risposta.
4. metodo *push*: invia un Urban Dataset alla SCP di destinazione.

Durante questi test si è monitorato il comportamento dell'applicativo di fronte a due diversi scenari. Nel primo scenario figura 10 si presuppone che il JWT non sia più valido, perciò, l'applicativo dovrà tentare di effettuare nuovamente il login, di utilizzare il nuovo token fornito per il collegamento con il WS, e di inviare l'UD.

Figura 10 - Screenshot dei messaggi stampati dal MDA (Scenario 1)

Come si può osservare il comportamento dell'applicativo è quello richiesto. Infatti:

- Viene invocato il metodo test ottenendo come risposta che la connessione con la SCP è ancora in atto;
- Viene invocato il metodo isAlive ottenendo come risposta che il token non è più valido;
- Viene invocato il metodo login con autenticazione ottenendo come risposta un nuovo token valido;
- Il nuovo token viene memorizzato ed utilizzato per le comunicazioni future. Nel caso riportato viene invocato il metodo push con successo.

Nell'ultimo scenario, figura 11, si è ipotizzato che il WS sia attivo e che il JWT sia ancora valido, perciò l'applicativo dovrà: inviare l'UD alla WS, attendere il tempo di riattivazione, e iniziare un altro ciclo di funzionamento. Di seguito le stampe di verifica ottenute tramite il metodo di stampa su consolle.

Figura 11 - Screenshot dei messaggi stampati dal MDA (Scenario 2)

3 Audit Energetico

Durante questa attività è stata implementata una piattaforma informatica in grado di acquisire, memorizzare ed elaborare i dati energetici riguardanti le unità abitative private, ed è basata su una architettura modulare che dal punto di vista strutturale è suddivisa in di macro-elementi interoperabili In particolare, si possono distinguere: l'Engine che si occupa di elaborare i dati e produrre simulazioni energetiche per fornire benchmarks e suggerimenti di riqualificazione energetica; l'Orchestrator che si occupa di gestire e supervisionare il flusso di informazioni che devono essere inserite ed utilizzate all'interno del blocco Engine e di recuperare da esso i risultati delle elaborazioni effettuate; lo Storage si occupa di mantenere in memoria i dati energetici degli utenti e i risultati delle elaborazioni effettuate dall'Engine; l'Application si occupa di fornire un'interfaccia semplice e intuitiva che accompagna l'utente nel processo di immissione dei dati energetici e nella visualizzazione dei risultati ottenuti a seguito delle elaborazioni del blocco Engine. Il presente report fornisce una descrizione delle funzionalità che realizza il macro-elemento Application di cui si è occupato il gruppo di ricerca di Roma Tre. Simulando il comportamento di un utente che vuole usufruire del servizio di *Audit Energetico*, verranno evidenziate le principali funzionalità e dinamiche che caratterizzano il front-end dell'interfaccia di acquisizione dati.

3.1 Aspetti grafici e funzionalità del front-end

L'elemento *Application* è inserito all'interno dell'ecosistema con il compito di fungere da interfaccia tra l'utente finale del servizio e la piattaforma informatica. Per questo motivo deve essere in grado di comunicare con gli altri componenti con cui deve scambiare informazioni. Lo scopo principale di questo elemento è quello di tradurre i meccanismi di inserimento dati utilizzati nel macro-elemento *Engine* in elementi grafici che permettono una rappresentazione user friendly migliorando così l'accessibilità della piattaforma. Quindi, le funzionalità messe a disposizione dell'utente risultano essere più comprensibili grazie all'utilizzo di *form* per la gestione e manipolazione dei dati messi a disposizione dal linguaggio HTML utilizzato in fase di sviluppo software della web application. In particolare, sono stati implementati elementi per l'inserimento dati (campi di testo, liste dropdown, checkbox, steppers, sliders, bottoni con icona), elementi per la compilazione facilitata (bottoni con testo, tooltip), e elementi per attivare ulteriori selezioni (interruttori toggle, checkbox).

Figura 12 - Elementi grafici di inserimento dati

Questi strumenti sono utilizzati all'interno delle pagine web che permettono all'utente di navigare agevolmente le domande del questionario di audit. Tutte le pagine presentano il medesimo template di base che di volta in volta può subire delle leggere modifiche in base alle esigenze di visualizzazione. La struttura della pagina, che è suddivisa in aree funzionali, tiene conto del contenuto grafico e testuale visualizzato, e della gestione ottimale dello spazio.

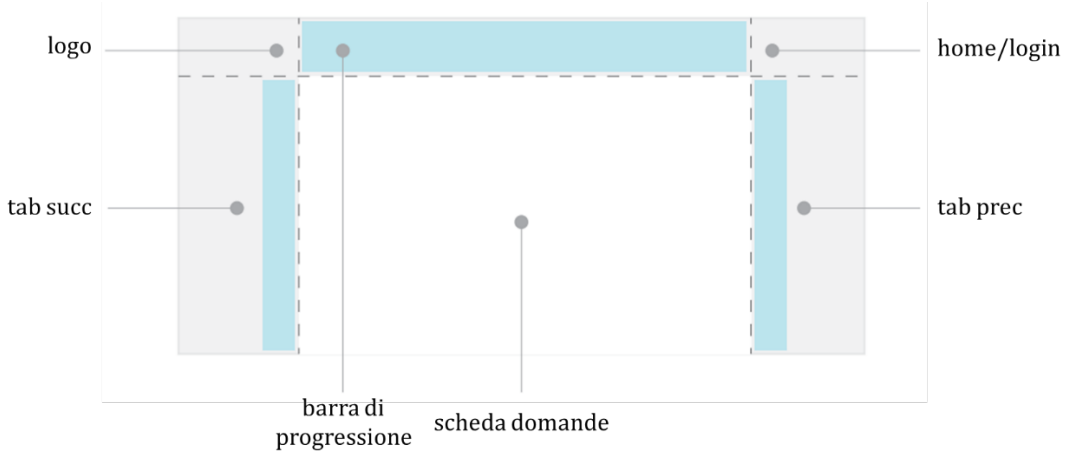


Figura 13 - Layout delle pagine

In particolare, si possono distinguere: l'area *scheda domande* in cui vengono riportate le domande del questionario, e l'area dedicata alla *barra di progressione*, grazie alla quale tenere traccia visiva sullo stato d'avanzamento del questionario. Nelle due aree denominate *tab succ* e *tab perc* sono stati implementati due pulsanti con i quali è possibile navigare orizzontalmente all'interno del questionario, ossia spostarsi da una macro-tematica all'altra. In alto a sinistra è stato riservato uno spazio per mantenere in sovraimpressione il logo del capoprogetto ENEA, oppure del progetto stesso, a seconda dei casi. Infine, in alto a destra, nell'area funzionale *home/login* è stato implementato un pulsante di *shortcut* verso la pagina iniziale oppure la pagina di login, a seconda dei casi. A chiudere le pagine, nella parte inferiore e presente una banda orizzontale all'interno della quale sono riportati i loghi dei partners che partecipano al progetto di *Audit Energetico* ossia, da sinistra a destra, Università La Sapienza, Università Roma Tre e il capo progetto ENEA.
Il comportamento dinamico che deve assumere ogni pagina della piattaforma web è gestito attraverso degli script sviluppati in Javascript grazie ai quali sono stati definiti i meccanismi di controllo e di coerenza dei valori inseriti dall'utente. Infatti, è stato previsto che se un dato necessario per l'elaborazione non sia stato inserito il processo non permette di proseguire alla pagina successiva segnalando l'errore e riportando la visualizzazione sull'elemento di input mancante. Stesso comportamento viene assunto nel caso in cui un valore inserito non sia coerente con le informazioni rese in precedenza, o conforme con i valori limiti previsti per quel particolare dato. Grazie a questo tipo di programmazione si gestiscono anche le funzionalità di auto completamento e l'attivazione di parti del questionario associate alla dinamica degli interruttori toggle. Per maggiori dettagli si rimanda al report riguardante la precedente annualità.

3.2 Soluzione realizzata

In questa sezione si vuole presentare la web application implementata per realizzare le funzionalità di audit energetico per le utenze residenziali private. Come detto in precedenza, è stato necessario tradurre le specifiche utilizzate dall'elemento *Engine* e suddividere il processo di compilazione in sei step consecutivi che corrispondono ai cinque macro-argomenti con cui è possibile classificare le domande proposte all'utente, e con l'aggiunta del percorso di benvenuto che introduce alla compilazione del questionario. Per chiarezza di esposizione, viene seguito il processo di accesso e compilazione messo a disposizione dell'utente per evidenziare le soluzioni grafiche e dinamiche utilizzate.

3.2.1 Percorso di benvenuto e accesso alla piattaforma

Prima di procedere con l'accesso registrato al processo di audit, vengono fornite tutta una serie di informazioni utili per la compilazione del questionario. In questo modo l'utente avrà chiaro quali sono le possibili azioni da effettuare durante l'inserimento dei dati e le informazioni necessarie per ottenere un'elaborazione corretta da parte dell'elemento *Engine*.
La prima visualizzazione che viene presentata ad un utente che accede al servizio di audit, denominata *home*, presenta una breve descrizione degli obiettivi che si prefigge il processo di audit.

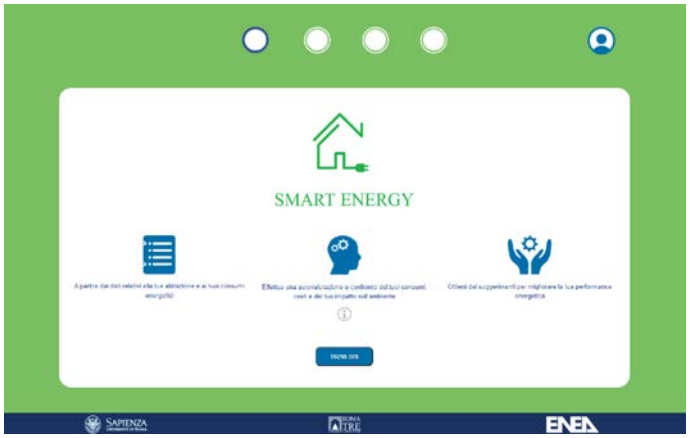


Figura 14 - Prima schermata di Benvenuto

Quindi, si può procedere cliccando sul pulsante *Inizia ora* e visualizzare la seconda schermata, *come funziona*, in cui viene illustrato per punti la procedura di compilazione del questionario e quali sono i suggerimenti e gli strumenti che vengono messi a disposizione come ausilio per l'utente.
Come si può notare, viene comunque sempre lasciata la possibilità di saltare il percorso di benvenuto premendo il pulsante nell'area *home/login* ed essere indirizzati direttamente alla pagina di accesso.

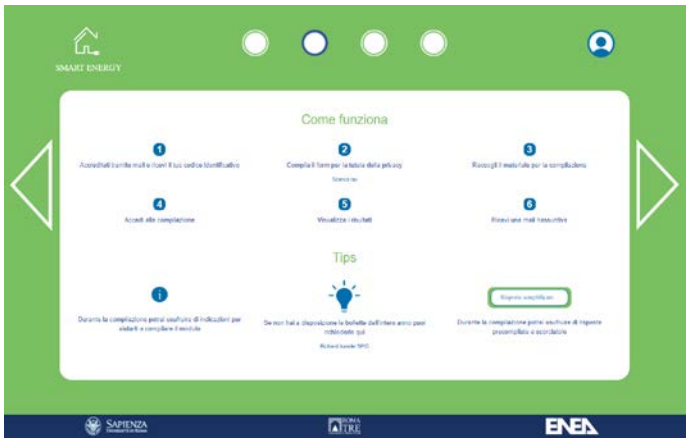


Figura 15 - Seconda schermata di Benvenuto

Utilizzando le frecce presenti nelle aree funzionali *prec/succ* si può navigare all'interno del percorso introduttivo all'audit. Cliccando sulla freccia *suc* si passa alla schermata successiva, *cosa otterrai*, nella quale sono brevemente descritti gli output per l'utente che ci si aspetta vengano forniti dall'elaborazione dell'audit energetico, come ad esempio individuare gli interventi consigliati per migliorare l'efficienza energetica.

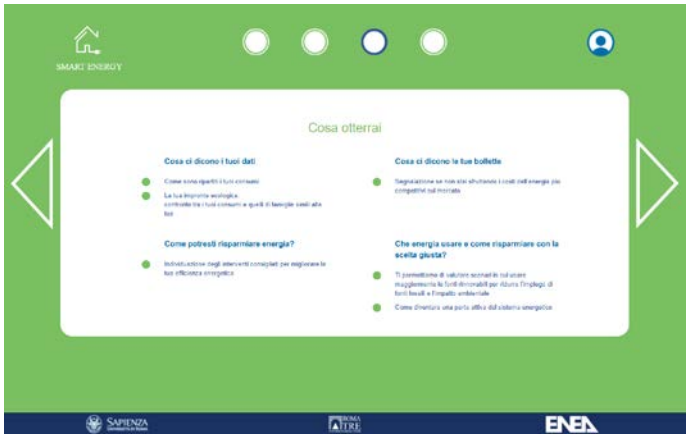


Figura 16 - Terza schermata di Benvenuto

Cliccando sulla freccia *suc* si passa alla schermata successiva, *cosa serve sapere*, in cui si riportano tutti gli strumenti e i dati necessari di cui premunirsi prima dii iniziare l'audit in modo tale da velocizzare e facilitare il processo di compilazione per l'utente. Si consiglia, ad esempio, di reperire le bollette mensili della luce e le caratteristiche tecniche dell'edificio in cui l'abitazione è immersa.

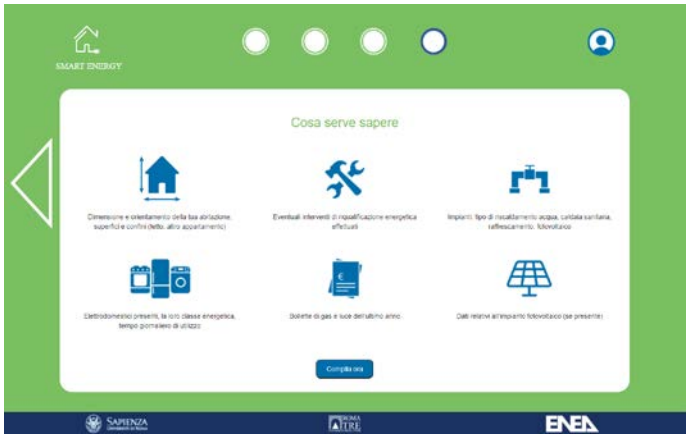


Figura 17 - Quarta schermata di Benvenuto

Una volta terminato il percorso di benvenuto, si è indirizzati alla pagina di accesso al sistema di simulazione dei consumi di ENEA, *accesso*. Tramite questa interfaccia è possibile creare una nuova utenza oppure accedere come un utente registrato, come indicato testualmente.

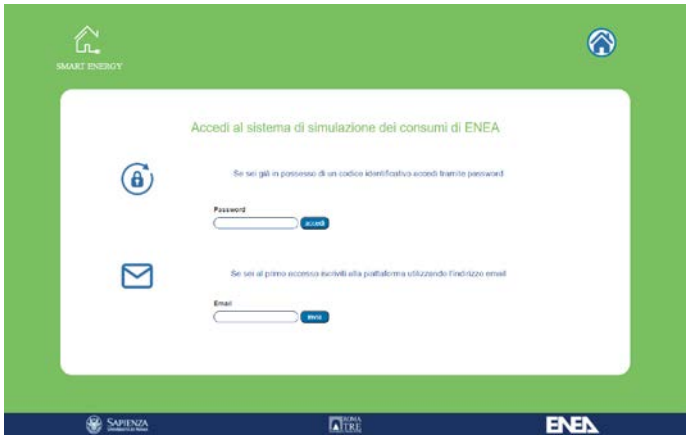


Figura 18 - Pagina di Login

L'utente può creare un nuovo profilo inserendo una e-mail di riferimento e premere il tasto invio. Se l'indirizzo è già presente tra gli utenti registrati viene riportato un messaggio testuale che invita ad effettuare l'accesso tramite password, figura 19a, altrimenti viene fornito un codice alfanumerico che l'utente dovrà appuntare e inserire nel campo password, figura 19b. Una volta premuto il pulsante di invio corrispondente, si viene indirizzati alla prima pagina del questionario, *scheda1*, riguardante il primo macro-argomento del questionario.
Invece, per accedere come utente registrato è sufficiente inserire nel campo password il codice alfanumerico che è stato visualizzato durate la procedura di registrazione e, dopo aver premuto sul pulsante di invio corrispondente, si viene indirizzati alla pagina associata al primo macro-argomento in ordine logico-sequenziale per cui non si è ancora compilato il questionario.
Se l'utente è verificato apparirà un pop-up testuale "Inizia il questionario. Vieni indirizzato alla pagina di riferimento" e per proseguire con la compilazione si dovrà cliccare su *ok*, figura 19c.
Cliccando sul pulsante presente nell'area funzionale *home/login* si viene indirizzati nuovamente alla prima visualizzazione presentata, ossia la pagina *home* della piattaforma.

a.

b.

c.

Figura 19 - a) Tentativo di accesso con una mail già registrata; b) Accesso con una mail non registrata; c) Accesso tramite password

3.2.2 AuditScheda1 – Dati generali

In questa parte del questionario si raccolgono le informazioni riguardanti l'ubicazione dell'edificio in termini di provincia e comune, e l'occupazione in termini di numero di persone presenti in casa a seconda della fascia oraria. Si mantiene traccia di queste ultime informazioni che verranno sfruttate nelle schede successive per permettere, ove possibile, la compilazione automatica di alcuni campi.

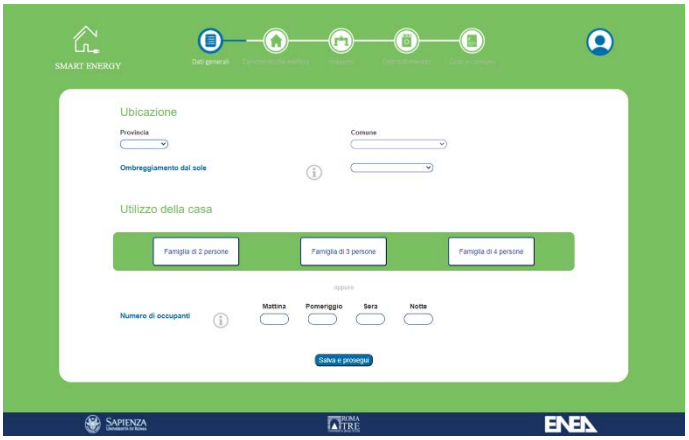


Figura 20 - Prima pagina del questionario

La procedura di compilazione del questionario prevede l'inserimento delle informazioni riguardanti:

- L'*ubicazione* dell'appartamento di cui si sta effettuando l'audit. In particolare:
  - La *provincia* – implementato tramite una lista dropdown contenente tutte le province d'Italia. In base alla scelta fatta viene abilitata la possibilità di inserire il comune di riferimento.

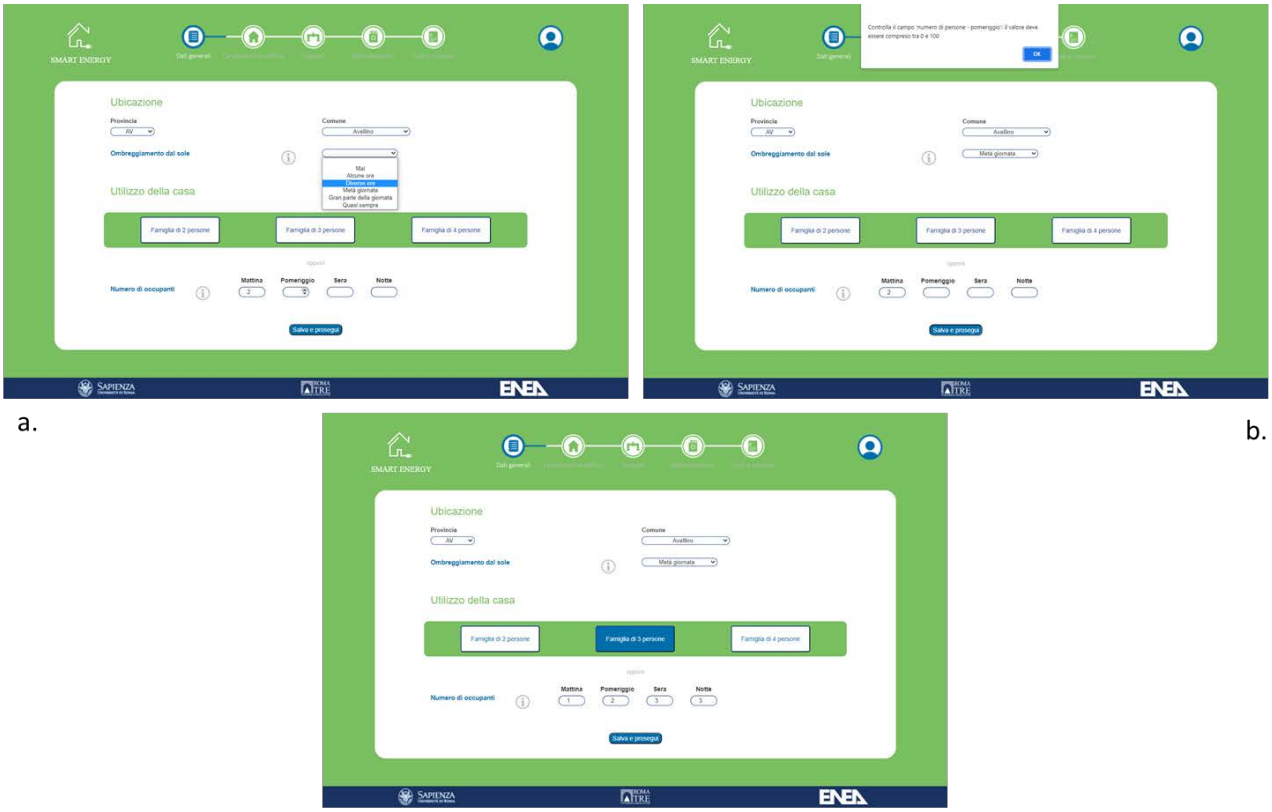
- Il *comune* – implementato tramite una lista dropdown contenente tutti i comuni appartenenti alla provincia selezionata.

- L'*ombreggiamento del sole* – implementato tramite una lista dropdown che riporta le opzioni previste dal questionario.

Queste informazioni, figura 21a, sono necessarie all'elemento engine per elaborare correttamente gli output che verranno forniti in base alla posizione geografica e all'esposizione dell'abitazione.

- **Il** *numero di occupanti* – implementato tramite quattro form di tipo steppers, incrementale e decrementale, suddivisi in base alla fascia oraria ossia *mattina*, *pomeriggio*, *sera*, *notte*. Il numero più alto all'interno di questi form determina il numero di occupanti. Questo valore viene memorizzato dinamicamente per poter essere di nuovo disponibile durante il proseguo del questionario. Per agevolare l'inserimento di questa informazione sono disponibili tre bottoni con testo ognuno dei quali corrisponde ad un profilo di occupazione in base al numero di persone che usufruiscono dell'appartamento, figura 21c.

Terminato l'inserimento dei dati richiesti in questa prima scheda, si può procedere cliccando sul bottone *salva e prosegui* che avvierà il controllo di coerenza sui valori immessi. Nel caso venga riscontrato un errore, figura 21b, questo viene prontamente segnalato e motivato attraverso un pop-up, permettendo così una facile correzione da parte dell'utente. Nel caso non vi siano incoerenze sui dati, l'elemento *application* li salva all'interno del database, l'elemento *storage*, e indirizza alla visualizzazione della scheda successiva del questionario.



a. b.

c.

Figura 21 – Prima pagina del questionario a) Esempio di compilazione della prima pagina del questionario; b) Esempio di comunicazione di un errore di compilazione; c) Esempio utilizzo del bottone per l'autocompilazione

3.2.3 AuditScheda2 – Caratteristiche architettoniche

In questa parte del questionario si raccolgono le informazioni riguardanti le caratteristiche architettoniche dell'abitazione e dell'edificio che la contiene. In particolare, si richiedono:

- I *dati generali* dell'edificio, figura 22a, come:
  - Il *periodo di costruzione* – implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
  - Il *numero di piani dell'intero edificio* – implementato tramite un form di tipo steppers, incrementale e decrementale, con valore minimo 0 e valore massimo 100.
  - Il *numero di piani dell'abitazione* – implementato tramite un form di tipo steppers, incrementale e decrementale, con valore minimo 0 e senza un valore massimo predefinito. In fase di controllo di coerenza verrà segnalato se il numero di piani dell'abitazione eccede il numero di piani dell'edificio, limite che evidentemente non si deve superare.
  - L'*altezza media di ogni piano* – implementato tramite un form di tipo steppers, incrementale e decrementale, con valore minimo 2m e valore massimo 6m.
  - Il *numero di stanze* – implementato tramite un form di tipo steppers, incrementale e decrementale, con valore minimo 1 e valore massimo 100.
  - Il *numero di finestre* – implementato tramite un form di tipo steppers, incrementale e decrementale, con valore minimo 1 e valore massimo 100.
  - Il *colore dei muri esterni* – implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
  - Il *colore della copertura* – implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
- I dati sulla *pianta e i confini* dell'abitazione, figura 22b, come:
  - La *forma della pianta dell'abitazione* – questa informazione può essere inserita selezionando uno dei quattro bottoni con icona al cui interno è riportata la rappresentazione grafica della pianta, mentre nella parte inferiore è presente la rappresentazione testuale. Le possibili scelte sono quelle previste dall'elemento *engine*. L'avvenuta selezione di uno dei bottoni con icona permette di inserire i dati riguardanti questa sezione e che sono riportati di seguito.
  - La *posizione del nord* – una volta selezionata la forma della pianta, una sua rappresentazione grafica compare all'interno della rosa dei venti. Utilizzando lo slider posto appena sopra è possibile modificare la posizione del nord rispetto ai lati dell'abitazione.
  - La *lunghezza* di ogni lato – a seconda della pianta selezionata vengono abilitate le form di tipo steppers associate ad ogni lato esistente, ed è quindi possibile inserirvi il valore misurato o che si ha a disposizione
  - Il tipo di locali con cui è *confinante* ogni lato dell'edificio – questa scelta è implementata tramite dei pulsanti di selezione mutuamente esclusivi e rispecchiano le possibilità previste dall'elemento *engine*.
- I dati sugli interventi di riqualificazione effettuati sull'abitazione, figura 22c – questa sezione, inizialmente disabilitata, può essere attivata attraverso l'uso del toggle associato. A questo punto è possibile selezionare gli interventi attraverso i toggles corrispondenti. Una volta attivati, si possono inserire le informazioni, per ogni intervento, riguardanti:
  - Il *periodo* di realizzazione – implementato tramite una lista dropdown che riporta le scelte previste dall'elemento *engine*.
  - La *percentuale* di realizzazione – implementata tramite uno slider con valore minimo 0% e valore massimo 100%.

Viene mantenuta traccia di alcune di queste informazioni poiché saranno sfruttate nelle schede successive per permettere, ove possibile, la compilazione automatica di alcuni campi.
Oltre alle sezioni di inserimento appena descritte, è presente una parte di *riepilogo dei dati geometrici*, figura 22c, che permette all'utente di visualizzare il calcolo delle superfici calpestabili in base ai dati inseriti sopra, dopo aver premuto il bottone *calcola*.
Terminato l'inserimento dei dati richiesti in questa seconda scheda, si può procedere cliccando sul bottone *salva e prosegui* che avvierà il controllo di coerenza sui valori immessi. Nel caso venga riscontrato un

errore, figura 22d, questo viene prontamente segnalato e motivato attraverso un pop-up, permettendo così una facile correzione da parte dell'utente. Nel caso non vi siano incoerenze sui dati, l'elemento *application* li salva all'interno del database, l'elemento *storage*, e indirizza alla visualizzazione della scheda successiva del questionario. Vale la pena segnalare che in questa sezione il controllo di coerenza riguarda, tra le altre cose, anche la verifica di due importanti aspetti:

- Il numero di piani dell'abitazione deve essere necessariamente uguale o inferiore al numero di piani che compongono l'intero edificio o stabile in cui l'abitazione è immersa.
- La coerenza tra i dati immessi riguardanti la lunghezza dei lati che compongono l'abitazione e la pianta che è stata selezionata. In particolare, a seconda della pianta si ha che:
    - Pianta quadrata – la lunghezza del lato A deve essere uguale alla lunghezza del lato B.
    - Pianta rettangolare – la lunghezza del lato A deve essere diversa dalla lunghezza del lato B.
    - Pianta ad angolo:
        - la lunghezza del lato C deve essere maggiore della lunghezza del lato A
        - la lunghezza del lato D deve essere minore della lunghezza del lato B
    - Pianta a doppio angolo:
        - la lunghezza del lato E deve essere minore della lunghezza del lato C
        - la somma della lunghezza dei lati D e F deve essere minore della lunghezza del lato B

a. b.

c. d.

Figura 22 - Seconda pagina del questionario a) dati dell'edificio; b) pianta e confini dell'abitazione; c) interventi di riqualificazione; d) esempio errore di compilazione

3.2.4 AuditScheda3 – Impianti

In questa parte del questionario si raccolgono le informazioni riguardanti la tipologia di impianti che caratterizzano l'abitazione, le principali specifiche tecniche, e la modalità di utilizzo. Più precisamente, si distinguono cinque diverse sezioni:

- Impianto di *riscaldamento*, figura 23a - i dati di interesse che vengono raccolti sono:

- Il *tipo di impianto* – implementato tramite pulsanti mutuamente esclusivi che rendono disponibili le opzioni definite dall'elemento *engine*.
- Il *tipo di generatore* – implementato tramite pulsanti mutuamente esclusivi che rendono disponibili le opzioni definite dall'elemento *engine*.
- La *classe energetica media* – implementato tramite lista dropdown che contiene le opzioni definite dall'elemento *engine*. Inizialmente è disabilitato, e viene abilitato solo se il tipo di generatore selezionato è la "pompa di calore elettrica".
- La *modalità di regolazione* - implementato tramite lista dropdown che contiene le opzioni definite dall'elemento *engine*.
- Il tipo di *terminali in ambiente* – implementato tramite pulsanti mutuamente esclusivi che rendono disponibili le opzioni definite dall'elemento *engine*.
- L'*utilizzo del condizionatore per il riscaldamento* – implementato tramite un toggle che permette la scelta tra *si* e *no*.

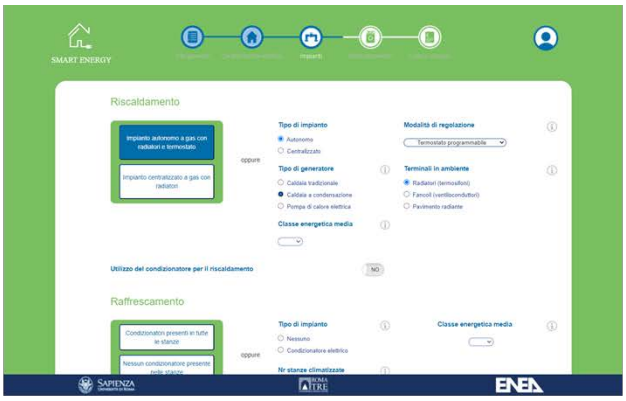
Sono presenti anche due bottoni con testo tramite i quali è possibile usufruire dell'opzione di auto-compilazione di alcuni dei principali campi di questa sezione in base a due profili preimpostati, ossia "impianto autonomo a gas con radiatori e termostato" e "impianto centralizzato a gas con radiatori".

- Impianto di *raffrescamento*, figura 23b – i dati di interesse che vengono raccolti sono:
  - Il *tipo di impianto* – implementato tramite pulsanti mutuamente esclusivi che rendono disponibili le opzioni definite dall'elemento *engine*.
  - Il numero di stanze climatizzate – implementato tramite un form di tipo steppers, incrementale e decrementale, con valore minimo 0 e con valore massimo pari al numero di stanze dichiarato nella pagina precedente del questionario. Inizialmente è disabilitato, e viene abilitato solo se il tipo di impianto selezionato è il "condizionatore elettrico".
  - La *classe energetica media* – implementato tramite lista dropdown che contiene le opzioni definite dall'elemento *engine*. Inizialmente è disabilitato, e viene abilitato solo se il tipo di impianto selezionato è il "condizionatore elettrico".
  - L'utilizzo di *altri apparecchi* per il raffrescamento – implementato tramite un toggle che permette la scelta tra *si* e *no*. Impostando il toggle su *si*, viene fornita la possibilità di segnalare l'utilizzo di altri apparecchi, in particolare ventilatori e deumidificatori, abilitando l'interazione con i due toggle corrispondenti. Ponendo come valore *si* a uno dei toggles vengono abilitate le form per l'inserimento delle informazioni associate a quel particolare dispositivo. I dati da inserire sono:
    - La *quantità* intesa come il numero di dispositivi – implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 1 e un valore massimo pari al numero di stanze dichiarato nella pagina precedente del questionario.
    - L'*uso giornaliero* – implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 0,1 e un valore massimo pari a 24 ore.

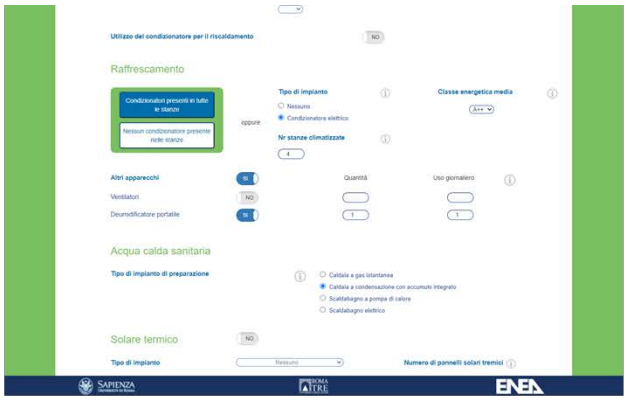
  Sono disponibili anche due bottoni con testo tramite i quali è possibile usufruire dell'opzione di auto-compilazione di alcuni dei principali campi di questa sezione in base a due profili preimpostati, ossia "nessun condizionatore presente nelle stanze" e "condizionatori presenti in tutte le stanze". Selezionando quest'ultimo bottone verrà impostato il tipo di impianto come "condizionatore elettrico" e compilato il campo "numero di stanze climatizzate" con il numero di stanze presenti nell'abitazione che è stato dichiarato nella pagina precedente del questionario.

- impianto di erogazione dell'*acqua calda sanitaria*, figura 23b – il dato di interesse che viene raccolto riguarda il *tipo di impianto di preparazione* ed è implementato tramite pulsanti mutuamente esclusivi che rendono disponibili le opzioni definite dall'elemento *engine*.

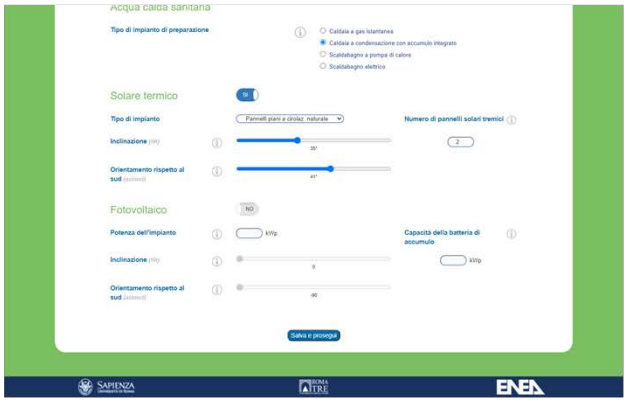
- impianto *solare-termico*, figura 23c – per poter inserire i dati riguardanti questo impianto è necessario che il toggle associato sia impostato su *si*, in questo modo si abilitano le form per l'inserimento delle seguenti informazioni:
    - Il *tipo di impianto* – implementato tramite una lista dropdown che contiene le opzioni definite dall'elemento *engine*.
    - L'*inclinazione* (tilt) dei pannelli – implementato tramite una form di tipo slider con valore minimo pari a 0 gradi e valore massimo pari a 90 gradi.
    - L'*orientamento rispetto al sud* (azimut) dei pannelli - implementato tramite una form di tipo slider con valore minimo pari a -180 gradi e valore massimo pari a 180 gradi.
    - Il *numero di pannelli solari termici* – implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 1 e un valore massimo pari a 100 unità.
- impianto *fotovoltaico*, figura 23c – per poter inserire i dati riguardanti questo impianto è necessario che il toggle associato sia impostato su *si*, in questo modo si abilitano le form per l'inserimento delle seguenti informazioni:
    - La *potenza dell'impanto* – implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 0,1 e un valore massimo pari a 100 kWp.
    - L'*inclinazione* (tilt) dei pannelli – implementato tramite una form di tipo slider con valore minimo pari a 0 gradi e valore massimo pari a 90 gradi.
    - L'*orientamento rispetto al sud* (azimut) dei pannelli - implementato tramite una form di tipo slider con valore minimo pari a -90 gradi e valore massimo pari a 90 gradi.
    - La *capacità della batteria di accumulo* – implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 0,1 e un valore massimo pari a 100 kWp.



a.



b.



c.

Figura 23 - Terza pagina del questionario a) riscaldamento; b) impianto di raffrescamento e acqua sanitaria; c) solare termico e fotovoltaico

3.2.5 AuditScheda4 – Elettrodomestici

In questa parte del questionario si raccolgono le informazioni riguardanti la tipologia e quantità di elettrodomestici presenti nell'abitazione e una stima sul loro utilizzo medio settimanale o giornaliero. In particolare, i dispositivi energivori domestici vengono suddivisi in base al loro utilizzo per facilitare l'operazione di inserimento da parte dell'utente. Più precisamente, si distinguono cinque diverse sezioni:

- La *cucina*, figura 24a, - i dati di interesse che vengono raccolti riguardano l'utilizzo del piano cottura, del forno e del forno a microonde, e sono:
    - La *tipologia di dispositivo* – implementato tramite pulsanti mutuamente esclusivi che rendono disponibili le opzioni definite dall'elemento *engine.*
    - I *minuti di uso giornaliero* – implementato tramite un form di tipo steppers, incrementale e decrementale, che prevede un valore minimo pari a 0 e un valore massimo pari a 1440 minuti.

  Sono previsti tre bottoni con testo tramite i quali è possibile usufruire dell'opzione di auto-compilazione del campo riguardante l'uso giornaliero a seconda di tre profili preimpostati e basati sul numero di occupanti dichiarato nelle pagine precedenti del questionario.

- La *refrigerazione*, figura 24a – i dati di interesse che vengono raccolti riguardano l'utilizzo di dispositivi per la refrigerazione del cibo, e sono:
    - Il *tipo* di dispositivo - implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
    - Il *volume* del dispositivo – implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 1 litro e un valore massimo non definito.
    - La *classe energetica* dichiarata – implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.

  Viene fornita all'utente la possibilità di inserire le informazioni riguardanti al più tre dispositivi di refrigerazione del cibo. L'inserimento dei dati riguardanti il singolo dispositivo può avvenire solo a seguito dell'attivazione del toggle corrispondente.

- Il *lavaggio, pulizia e stiratura*, figura 24b – i dati di interesse riguardano i dispositivi utilizzati per il lavaggio e la stiratura dei vestiti, il lavaggio delle stoviglie, e la pulizia di casa. Le informazioni di un particolare dispositivo possono essere inserite solo ponendo il toggle associato sul valore *si*. In base ai dispositivi selezionati le informazioni di interesse variano di poco. In particolare, si ha;
    - Per *lavatrice*, *asciugatrice*, *lavastoviglie*, *lavasciuga*
        - I *cicli settimanali*, implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
        - La *classe energetica*, implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
        - La *capacità* disponibile, implementato tramite una lista dropdown in cui sono riportate le opzioni che prevede l'elemento *engine*.
    - Per *aspirapolvere*, *scopa elettrica*, *ferro da stiro*, *ferro da stiro con caldaia*, si deve inserire l'utilizzo giornaliero in minuti. E' implementato tramite quattro sliders, associati ai quattro dispositivi, che possono assumere valore minimo 0 e valore massimo 240 minuti.

  E' previsto un bottone con testo tramite il quale è possibile usufruire dell'opzione di auto-compilazione dei campi riguardanti i *cicli settimanali* di utilizzo dei dispositivi in base al numero di occupanti dichiarato nelle pagine precedenti del questionario.

- L'*illuminazione* dell'abitazione, figura 24c - i dati di interesse riguardano i tipi di dispositivi utilizzati per l'illuminazione all'interno dell'abitazione, e il loro numero. Questo inserimento è implementato tramite un form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un

valore minimo pari a 1 e un valore massimo pari a 100 unità. Le informazioni di un particolare dispositivo possono essere inserite solo ponendo il toggle associato sul valore *si*.
Sono disponibili anche tre bottoni con testo tramite i quali è possibile usufruire dell'opzione di auto-compilazione in base a tre diversi profili preimpostati di illuminazione, ossia "molto efficiente", "poco efficiente" e "media efficienza".

- L'utilizzo di *computer/internet*, figura 24c - i dati di interesse riguardano la quantità e l'utilizzo giornaliero di computer fissi e portatili connessi all'interno dell'abitazione. Questi inserimenti sono implementati tramite forms di tipo steppers, incrementale e decrementale, che, quando abilitati, prevedono un valore minimo e un valore massimo. In particolare, si ha che:
  - La *quantità* può assumere valore minimo pari a 1 e un valore massimo pari a 100.
  - L'*utilizzo giornaliero* in ore può assumere valore minimo 0,1 e valore massimo 24 ore.

  Le informazioni di un particolare dispositivo possono essere inserite solo ponendo il toggle associato sul valore *si*.
  Sono disponibili anche tre bottoni con testo tramite i quali è possibile usufruire dell'opzione di auto-compilazione in base a tre diversi profili preimpostati di utilizzo giornaliero, ossia "alto", "medio" e "basso".

- L'utilizzo di apparecchi per la *cura della persona*, figura 24d - i dati di interesse riguardano l'utilizzo giornaliero di asciugacapelli e di piastra per i capelli. Questi inserimenti sono implementati tramite sliders, che, quando abilitati, prevedono un valore minimo pari a 1 e un valore massimo pari a 1440 minuti.

a. b. c. d.

Figura 24 - Quarta pagina del questionario a) cucina e refrigerazione; b) lavaggio pulizia e stiratura; c) illuminazione e utilizzo del computer; d) cura della persona e altri apparecchi

Oltre a queste informazioni, viene data la possibilità all'utente di inserire altri tre apparecchi che non sono presenti nel questionario, fornendo le stesse indicazioni richieste per quelli presenti, ossia indicandone il nome, la quantità e l'utilizzo medio giornaliero o settimanale, figura 24d. Le informazioni di un particolare dispositivo possono essere inserite solo ponendo il toggle associato sul valore *si*. In particolare, si ha che:

- Il *tipo* di apparecchio può essere inserito tramite una form testuale.
- La *quantità* di apparecchi di quella particolare tipologia prevede un inserimento tramite form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevedono un valore minimo pari a 1 e un valore massimo pari a 100 unità.
- La potenza erogata dall'apparecchio può essere inserita sfruttando una form di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 1 e un valore massimo pari a 10000 Watt.
- L'inserimento riguardante *l'uso giornaliero* del dispositivo è implementato tramite forms di tipo steppers, incrementale e decrementale, che, quando abilitato, prevede un valore minimo pari a 1 e un valore massimo pari a 1440 minuti.

Terminato l'inserimento dei dati si può procedere cliccando sul bottone *salva e prosegui* che avvierà il controllo di coerenza sui valori immessi. Nel caso venga riscontrato un errore questo viene prontamente segnalato e motivato attraverso un pop-up, permettendo così una facile correzione da parte dell'utente. Nel caso non vi siano incoerenze sui dati, l'elemento *application* li salva all'interno del database, l'elemento *storage*, e indirizza alla visualizzazione della scheda successiva del questionario.

3.2.6 AuditScheda5 – Consumi e costi

In questa parte del questionario si raccolgono le informazioni, desumibili dai dati contenuti nella bolletta dell'utente, riguardanti due servizi essenziali principali, ossia l'elettricità ed il gas. Per entrambi, è stata strutturata una form per facilitare l'inserimento dei dati e si è deciso di lasciare all'utente la scelta del mese dal quale iniziare l'inserimento dei dati, l'importante è che questi siano basati su un arco di tempo annuale. In particolare, si ha:

- *Consumi elettrici* mensili e *spesa annuale*, figura 25, le cui informazioni richieste sono:
  - Il *tipo di utente*, da distinguere tra residenziale e non residenziale. Questa scelta è implementata tramite pulsanti mutuamente esclusivi.
  - Il *mese di partenza* della lettura, implementati tramite una lista dropdown contenente i mesi dell'anno. A seconda della scelta dell'utente si modificheranno automaticamente le intestazioni delle form dedicate all'inserimento dei dati da bolletta.
  - Il *consumo mensile*, implementato tramite 12 form di tipo stepper, incrementale o decrementale, che può assumere valore minimo pari a 0 e valore massimo pari a 5000 kWh.
  - La *spesa annuale*, implementata anch'essa tramite una form di tipo stepper, incrementale o decrementale, che può assumere valore minimo pari a 0,01 e valore massimo pari a 10000 euro.
- *Consumi del gas* mensili e *spesa di consumo* annuale, figura 25, le cui informazioni da inserire sono:
  - Il *mese di partenza* della lettura, implementati tramite una lista dropdown contenente i mesi dell'anno. A seconda della scelta dell'utente si modificheranno automaticamente le intestazioni delle form dedicate all'inserimento dei dati da bolletta.
  - Il *consumo mensile*, implementato tramite 12 form di tipo stepper, incrementale o decrementale, che può assumere valore minimo pari a 0 e valore massimo pari a 5000 Smc.
  - La *spesa annuale*, implementata anch'essa tramite una form di tipo stepper, incrementale o decrementale, che può assumere valore minimo pari a 0,01 e valore massimo pari a 10000 euro.

Terminato l'inserimento dei dati si può procedere cliccando sul bottone *salva e prosegui* che avvierà il controllo di coerenza sui valori immessi. Nel caso venga riscontrato un errore questo viene prontamente segnalato e motivato attraverso un pop-up, permettendo così una facile correzione da parte dell'utente. Nel caso non vi siano incoerenze sui dati, l'elemento *application* li salva all'interno del database, l'elemento *storage*. L'utente viene, quindi, avvisato tramite messaggio testuale dell'avvenuto salvataggio di tutte le informazioni, e viene indirizzato alla pagina di accesso alla piattaforma. Infatti, i risultati saranno disponibili solo dopo l'elaborazione da parte dell'elemento *Engine*, che solitamente richiede qualche minuto.

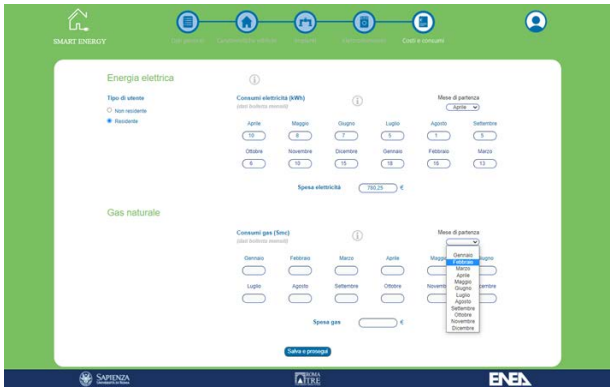


Figura 25 - Quinta pagina del questionario

4 Conclusioni

La collaborazione in atto tra l'Università Roma Tre e la Divisione Energia del Centro Ricerche ENEA "La Casaccia" è incentrata sullo sviluppo di tecnologie innovative e soluzioni informatiche volte alla pianificazione strategica degli interventi di riqualificazione energetica degli edifici, siano questi residenziali o ad uso ufficio. Quindi, si punta a creare una mappatura e raccolta standardizzata ed omogenea dei consumi e porre le basi per un catasto nazionale, strutturato in un DB interoperabile e in un tool di monitoraggio e valutazione delle prestazioni dei servizi. In particolare, la ricerca è stata suddivisa in due linee progettuali principali: District Database e l'Audit Energetico.
La prima, District Database, ha come obiettivo quello di gestire l'interscambio in tempo reale dei dati riguardanti gli edifici che compongono il distretto universitario dell'Università di Roma Tre. Lo scambio di dati viene realizzato grazie ad un applicativo informatico automatizzato, thread, definito tramite il linguaggio di programmazione Java che:

1. Preleva i dati energetici riguardanti un edificio afferente al Dipartimento di Ingegneria dell'Università Roma Tre;
2. Organizza i dati secondo una struttura json compliant con gli standard definiti da ENEA;
3. Tenta l'invio dell'UD al WS della SCPCasaccia;
4. Memorizza i dati all'interno di un database ausiliario non relazionale

Il software implementato risulta essere efficace, in quanto sono state testate le funzionalità previste, ed efficiente, volutamente concepito con bassa complessità computazionale, quindi veloce, e con un buon uso delle risorse a disposizione. Le connessioni ai database sono minime, vengono infatti ottimizzate dal punto di vista temporale. Inoltre, ono stati utilizzati tutti strumenti open source per la progettazione, quindi, il programma può essere facilmente replicato. Tramite il file properties, è molto semplice modificare gli indirizzamenti dei databases, rendendo l'applicativo estremamente portabile. È uno strumento scalabile orizzontalmente, poiché, può essere facilmente ampliato, tramite l'aggiunta di query di lettura, diverse da quella che si è implementata, per altri dati del database dal quale si prelevano, e query diverse di inserimento nel database MongoDB. Inoltre, lo strumento risulta essere minimamente invasivo rispetto all'ecosistema tecnologico preesistente, poiché il database dal quale si prelevano i dati, quindi quello di Roma Tre, non viene mai modificato, e viene richiesto solo l'accesso in lettura. Un possibile, ed importante, sviluppo futuro di questo progetto, è l'utilizzo del database ausiliario nel caso in cui la connessione con la SCP viene accidentalmente interrotta. Il database, memorizzando i dati delle varie acquisizioni in un semplice array di oggetti, ossia il campo historian, all'interno di ciascun documento, alla instaurazione della nuova connessione con la SCP, invierà tutte quelle acquisizioni che per quel determinato tempo non sono state inviate. Ciò è utile al fine di non perdere alcuna informazione ed assicurare continuità nello scambio di dati. Quindi, grazie alla collaborazione tra Roma Tre ed ENEA, si è potuto implementare una effettiva comunicazione tra un esempio di edifici cittadini, in questo caso universitari, e la SCP del progetto PELL.

L'altra necessità emersa è quella di acquisire ed elaborare i dati energetici riguardanti le abitazioni private. Utilizzando un portale web, gli utenti, su base volontaria, potrebbero agevolmente inserire tutte le informazioni utili e necessarie per individuare e segnalare le zone geografiche o gli edifici con maggiore necessità di interventi di riqualificazione energetica. In questo modo sarebbe più semplice fornire un feedback mirato all'utente tramite statistiche e benchmark di riferimento, consigliando dove poter intervenire per migliorare il profilo energetico della sua abitazione. Adottando questa strategia si rendono omogenei i dati provenienti da tutti gli utenti e si consente una loro corretta elaborazione e valutazione statistica. Tenendo presente l'obiettivo dell'attività, la soluzione proposta è basata su una architettura modulare che dal punto di vista strutturale è suddivisa in di macro-elementi interoperabili, come emerge dalle specifiche funzionali e di progettazione definite nelle annualità precedenti.
E' stata, quindi, implementata un'interfaccia utente che si occupa della gestione dei seguenti processi:

1. Registrazione di un nuovo utente, che consiste anche nella generazione di un codice univoco per il riconoscimento dell'utente;

2. Controllo degli accessi alla piattaforma, per consentire l'utilizzo del servizio di Audit Energetico solo ad un utente registrato;
3. Compilazione del questionario, per guidare l'utente nel processo di compilazione informandolo delle funzionalità disponibili;
4. Scrittura e lettura dal database, che deve gestire l'inserimento delle risposte e la lettura dei risultati dell'Audit;

In particolare, nell'ultima annualità, ci si è concentrati sull'integrazione di importanti funzionalità dinamiche e di autocompilazione del questionario, e sul perfezionamento dell'interfaccia grafica. Le pagine web utilizzate per la compilazione del questionario hanno tutte un layout fisso e sono caratterizzate dalla presenza di moduli per l'inserimento del testo e anche dei moduli per l'inserimento dei dati a risposta chiusa. Ad ogni pagina di compilazione è stato associato un macro-argomento relativo all'Audit Energetico e contiene le relative domande. Le risposte a ciascun macro-argomento vengono salvate nel database solo dopo che l'utente le ha completate e inviate. I dati vengono poi archiviati tenendo traccia del macro-argomento appena trattato. Gli utenti vengono quindi indirizzati alla pagina di compilazione successiva, portando ovviamente con sé il codice identificativo e i dati necessari per l'eventuale funzione di autocompilazione messa a disposizione dal front end.
L'interfaccia così progettata presenta una struttura non troppo articolata ma estremamente disaccoppiata presentando una spiccata modularità. Questa caratteristica assicura una buona portabilità e riusabilità della soluzione proposta. Infatti, si può notare che è possibile inserire, modificare, o eliminare qualsiasi sezione del questionario in modo chiaramente poco invasivo, senza pregiudicare la consistenza e la disponibilità dell'audit energetico stesso.
Come già detto, la piattaforma è stata implementata nei server ENEA presenti presso il R.C. "La Casaccia" ed è accessibile via [http://192.107.92.38/audit/Smart_Sim/AuditBenvenuto]. Attualmente la piattaforma è completa all'80% in quanto deve essere ancora implementata la pagina relativa alla visualizzazione dei risultati, che non sono ancora disponibili nel database. Con il lancio della versione beta dello strumento, abbiamo raccolto importanti informazioni per migliorare le dinamiche di compilazione e testare l'efficacia dello strumento finora implementato.
A valle di quanto riportato, gli obbiettivi prestabiliti per questa annualità sono stati raggiunti, fermo restando che su entrambe le progettualità devono essere realizzati numerosi test di robustezza e di utilizzo massivo prima di poter entrare a pieno regime.

5 Riferimenti

[1] ENEA, «Smart City Platform - La Casaccia,» [Online]. Available: https://smartcityplatform.enea.it/casaccia/.

[2] M. Conway, «Using JSON,» 2021.

[3] ENEA, «SCPS Communication 2.0,» [Online]. Available: https://smartcityplatform.enea.it/#/it/specification/communication/2.0/index.html.

[4] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh e R. Al-Hatmi, «Internet of Things: Survey and open issues of MQTT protocol,» in *International Conference on Engineering and MIS 2017*, 2018.

[5] K. Arnold, B. Kennedy e G. M. Novak, The Java programming language, Addison Wesley Professional, 2005.

[6] C. Muscino, B. Kennedy e G. M. Novak, «HTML: The Definitive Guide, Second Edition,» in *Computers in Physics*, 1998.

[7] E. A. Mayer, CSS: The Definitive Guide, O'Reilly Media Inc, 2006.

[8] D. Flanagan e W. S. Like, JavaScript: The Definitive Guide, 2006.

[9] C. Novelli, «Smart City Platform Specification Functional Level 1.0,» ENEA - RdS/PAR2017/104, 2017.

[10] ENEA, «UrbanDataset WEB LIBRARY,» [Online]. Available: https://smartcityplatform.enea.it/UDWebLibrary/it/urbandataset?name=BuildingElectricConsumption.

[11] K. Chodorow, MongoDB : the definitive guide.

[12] MongoDB, «MongoDB Compass - documentation,» [Online]. Available: https://docs.mongodb.com/compass/current/.

[13] E. Burnette, Eclipse IDE: pocket guide, O'Reilly.

[14] M. J. McLaughlin e S. Mikolaitis, MySQL Workbench: Data Modeling & Development, McGraw-Hill Education, 2013.

[15] ENEA, «SCPS Information 2.0,» [Online]. Available: https://smartcityplatform.enea.it/#/it/specification/information/2.0/.

Dario Masucci è un collaboratore di ricerca presso l'Università degli Studi "Roma Tre" dal 2015, anno in cui ha conseguito la laurea magistrale in Automazione. I suoi interessi di ricerca includono lo sviluppo di strumenti di supporto alle decisioni (DSS) incentrati sull'utilizzo di algoritmi di ottimizzazione multi-obiettivo, e soluzioni basate sull'Internet of Things (IoT) e la Big Data Analysis. Negli ultimi anni ha lavorato a diversi progetti europei ideando e realizzando algoritmi decisionali multicriterio, modelli di interdipendenze per le Infrastrutture Critiche (CI), e strumenti ad alto livello di gestione delle emergenze e degli asset coinvolti.