Sviluppo servizi di comunità e flessibilità

Ivano Turi

Report RdS/PTR2021/040

SVILUPPO SERVIZI DI COMUNITÀ E FLESSIBILITÀ

Ivano Turi (Harpa Italia srl)

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero per la Transizione Ecologica - 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: Energy Communities: Implementazione servizi LEC

Responsabile del Progetto: Claudia Meloni, ENEA

Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno del Contratto "*Sviluppo servizi base di flessibilità e comunità per portale Comunità Energetiche"*

Responsabile Unico del Procedimento ENEA: Claudia Meloni

Responsabile del Contratto per il Contraente: Vincenzo Pascucci

Indice

Sommario

Questa attività ha riguardato lo sviluppo dei primi moduli software propedeutici alla fruizione dei servizi offerti dal portale LEC per le comunità energetiche. A tal fine il modulo principale è stato quello relativo alla realizzazione di un Identity Provider (IDP) che consentisse la fruizione di tutti i servizi con la medesima identità digitale.

La filosofia di fondo del portale è quella della interoperabilità dei servizi e delle soluzioni, pertanto si sono riutilizzati e customizzati moduli sw relativi alla smart City platform.

Le attività condotte sono state articolate nei seguenti task:

Task 1: Progettazione Tecnica e Sviluppo Identity Provider (IDP) che ha riguardato la definizione dell'architettura software dell'IDP, lo sviluppo delle interfacce API (Application Programming Interface) M2M (Machine To Machine) e dell'SDK (Software Development Kit) e le interfacce utente HDMI (High-Definition Multimedia Interface) e Admin e la realizzazione di Open API per l'interazione M2M.

Task 2: Aggiornamento SCP che ha riguardato l'integrazione della GUI della Smart City Platform con l'IDP, l'integrazione del modulo UD Gateway con l'IDP.

Task 3: Modulo Data Fusion per gestione flessibilità energetica, relativo alla implementazione prototipale di acquisizione della flessibilità degli utenti della comunità energetica e calcolo della flessibilità potenziale complessiva della comunità.

Task 4: Sviluppo KPI relative alla premialità per l'autoconsumo (caso Prosumer) in cui vengono raccolti e storicizzati i dati aggregati di autoconsumo della EC.

Task 5: Test, Manutenzione e Coaching on-the-job.

1 Introduzione

Questa attività ricade nell'ambito delle attività relative alla Linea di Attività 1.48 "Energy Communities: Implementazione servizi LEC", che prevede lo sviluppo di un nuovo portale sulle Comunità Energetiche.

La attività di ricerca sulle Comunità Energetiche prende spunto dal voler fornire ai vari utenti di una comunità adeguati servizi e strumenti in grado di creare un ecosistema energetico intelligente e interattivo.

In tale contesto si mettono in relazione aspetti correlati dal punto di vista ambientale, sociale, tecnologico ed economico con l'obiettivo di migliorare la sostenibilità del sistema energetico.

Il modello ENEA propone la definizione e lo sviluppo di strumenti per la nascita di Energy Community a supporto dei differenti profili energetici che caratterizzano il tessuto urbano (abitazioni, terziario, piccole imprese).

L'obiettivo è lo sviluppo di metodologie, infrastrutture tecnologiche, modelli gestionali ed economici per sostenere e sviluppare l'aggregazione e l'iniziativa dei prosumers e la messa a punto di un processo auto-organizzativo (codificato) di una comunità di cittadini che sviluppa la capacità di auto-gestire una serie funzionalità connesse alla rete energetica; il modello si basa su tecnologie abilitanti e servizi aggregati e fa uso della tecnologia blockchain e di metodologie per la remunerazione della flessibilità e del comportamento virtuoso della comunità attraverso l'utilizzo di valute complementari.

Questa visione si implementa su una scala di medio periodo attraverso lo sviluppo di un portale dedicato alle EC, pertanto il punto di partenza consiste nella definizione dello stack tecnologico ICT in cui vengono individuati gli attori con l'aggiunta degli stream di dati che possono arrivare da altre sorgenti.

2 Descrizione delle attività svolte e risultati

3 Task 1 - Progettazione tecnica e sviluppo identity provider

Un Identity Provider (IDP in seguito) è definito come “un'entità di sistema che crea, conserva e gestisce le informazioni sull'identità degli utenti e fornisce anche servizi di autenticazione per affidare le applicazioni all'interno di una federazione o di una rete distribuita”.

Questo task ha comportato l’implementazione dell’IDP secondo quanto qui di seguito indicato.

3.1 Architettura software dell’IDP

È qui descritta l’architettura software adottata (Figura 1) per l’implementazione dell’IDP.

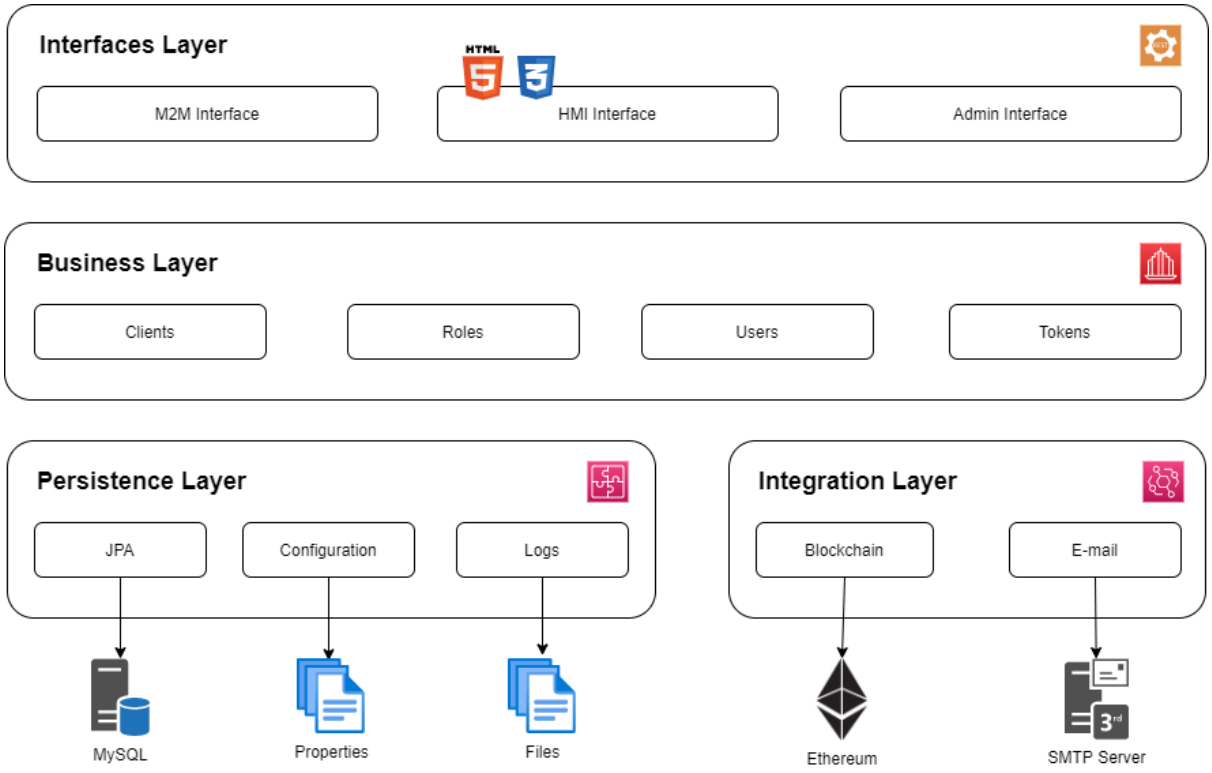

Figura 1 - Architettura Software

Si identificano diversi layer funzionali:

- Interfaces: rappresenta le interfacce tramite cui i diversi client accedono all’IDP. In particolare, sono identificate tre classi di interfacce:
  - M2M (machine to machine) interface: utilizzate per le integrazioni server to server. Sostituisce l’IdentityGateway indicato nell’appendice A dell’allegato tecnico ENEA.
  - HMI (human machine interface): utilizzate per le integrazioni da client che richiedono l’interazione dell’utente. Include le interfacce HTML (HyperText Markup Language) relative ai seguenti flussi:
    - Registrazione utente;

        - Login/logout;
        - Modifica profilo;
        - Conferma e-mail;
        - Cambio password.
    - Admin interface: utilizzate dal client di amministrazione dell'IDP.

  In tutti i casi indicati viene adottato il paradigma RESTful (REpresentational State Transfer). I dati scambiati fra i client e l'IDP (e viceversa) sono codificati in JSON.

- Business logic: il cuore del sistema, include i componenti che si occupano dei cicli di vita delle entità gestite dal sistema e degli algoritmi dell'IDP, fra cui il SSO (single sign-on). In particolare, le entità sono:
    - Clients (applications): rappresentano le applicazioni che possono utilizzare i servizi esposti dall'IDP. Ogni applicazione web, pertanto, viene censita nel sistema ed associata ad un id (client id) ed a una password (secret), generati automaticamente. Vanno inoltre specificati gli URL di callback per i flussi di login, logout, registrazione e modifica del profilo.
    - Roles: i suoli assegnabili agli utenti. I ruoli sono personalizzabili e trasversali rispetto ai diversi client presenti nel sistema. Il ruolo permette di definire un semplice sistema di autorizzazione, la cui reale implementazione è a carico delle diverse applicazioni client. Il nome del ruolo è un identificativo univoco.
    - Users: gli utenti che possono autenticarsi nel sistema. Oltre alle credenziali (e-mail e password), sono inclusi gli attributi: name, surname, organization, phone, address, emailConfirmed. Gli utenti sono in relazione con i ruoli e le applicazioni. L'email è un identificativo univoco.
    - Tokens: sono generati in tre diversi flussi:
        - SSO: nel flusso di SSO, descritto nei prossimi capitoli, viene generato un token che rappresenta l'avvenuto login dell'utente.
        - Validazione e-mail: in seguito alla registrazione di un nuovo utente, l'IDP invia una e-mail all'utente, affinché questa venga validata. Nel corpo dell'e-mail è presente un link che permette di avviare la procedura di verifica. Questo link include il token di validità della richiesta.
        - Cambio password: nel caso in cui l'utente abbia dimenticato la propria password, il sistema genera una e-mail con un link per avviare la procedura di cambio password. Questo link include il token di validità della richiesta.

  Oltre ai servizi di CRUD (Create Read Update Delete) per le entità sopra indicate, sono qui presenti le implementazioni dei servizi esposti dalle interfacce RESTful e HTML. Tali servizi saranno descritti nei capitoli successivi.

- Persistence: si occupa della memorizzazione permanente di diversi tipi di informazioni:
    - Entities: le entità sono persistite su database MySQL (Structured Query Language).
    - Configuration: la configurazione del sistema è su file properties.
    - Logs: i log delle diverse azioni eseguita nell'IDP sono memorizzate su file.
- Integration: include le integrazioni verso sistemi esterni. In particolare, è qui implementato un sistema di eventi generati dall'IDP contestualmente al ciclo di vita delle entità e alle azioni compiute dagli utenti. Questo sistema è utilizzabile per estendere dinamicamente le funzionalità dell'IDP, andando ad implementare opportuni listener di tali eventi. Tramite questo sistema sono fornite le seguenti funzionalità di integrazione verso servizi esterni:

- E-mail: modulo di integrazione con il server SMTP. Le e-mail sono generate per comunicare particolari eventi agli utenti e agli amministratori. Sono utilizzati i template HTML presenti nel sistema. Il servizio di spedizione delle e-mail fa uso del server SMTP messo a disposizione da ENEA. Gli eventi che producono la generazione di una comunicazione via e-mail sono:
  - Registrazione utente: viene inviata una e-mail in modo da verificarne la proprietà.
  - Reset della password: nel caso in cui l'utente abbia dimenticato la password, viene inviata un'e-mail contenente un link per il reset della password.
  - Cambio password: al cambio della password viene inviata un'e-mail che notifica l'avvenuta modifica.

3.2 Interfacce M2M e SDK

Le API (Application Programming Interface) M2M (Machine To Machine) sono utilizzabili per le integrazioni server to server. Queste API sono riservate ad utenti con ruolo M2MAgent. Benché sia possibile utilizzare direttamente le API M2M, è consigliata l'adozione dell'SDK (Software Development Kit), poiché semplifica diversi dettagli implementativi.

L'SDK è distribuito tramite un repository presente su GitLab. Periodicamente possono essere generati nuovi SNAPSHOT. È possibile indicare l'SDK come dipendenza diretta nel proprio pom.xml.

```
<repositories>
    <repository>
        <id>gitlab-maven</id>
        <url>https://gitlab.com/api/v4/packages/maven</url>
    </repository>
</repositories>
<dependencies>
    <dependency>
        <groupId>it.enea.scp.idp</groupId>
        <artifactId>scp-idp-sdk</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>
</dependencies>
```

Le dipendenze di runtime dell'SDK sono:

- org.apache.httpcomponents.client5:httpclient5:5.1
- org.codehaus.jackson:jackson-jaxrs:1.9.13
- org.apache.logging.log4:log4j-api:2.14.1
- org.apache.logging.log4j:log4j-core:2.14.1
- org.apache.logging.log4j:log4j-slf4j-impl:2.14.1
- io.jsonwebtoken:jjwt-api:0.11.2
- io.jsonwebtoken:jjwt-impl:0.11.2
- io.jsonwebtoken:jjwt-jackson:0.11.2

La classe che fornisce accesso alle API M2M è IdpManager. Va istanziata indicando:

- baseUrl dell'identity provider: attualmente è https://idp.smartcityplatform.enea.it/
- clientId e clientSecret: fornite al system integrator al momento del censimento dell'applicazione

LE API M2M richiedono che sia presente l'header HTTP (Hypertext Transfer Protocol) Authorization, valorizzato con il token di un utente M2MAgent associato al clientId.

E' possibile salvare il token che verrà utilizzato per invocare le API tramite il metodo setCredentials, specificando le credenziali dell'utente M2MAgent fornite. In questo modo verrà immediatamente eseguito un login e il token sarà memorizzato nell'istanza dell'IdpManager creata.

Nel caso in cui il token scada, l'SDK provvederà ad ottenerne uno nuovo tramite le credenziali precedentemente utilizzate.

Per l'interfaccia M2M sono presenti i metodi:

- login: flusso di login per gli accessi M2M.
- logout: logout dell'utente. Viene rimosso il token dell'utente.
- isValid: verifica che il token dell'utente sia valido.
- userList: ritorna la lista degli utenti associati ad una certa applicazione identificata dal proprio client id. Poichè un utente può essere rimosso in modo globale da tutte le applicazioni, è buona norma invocare periodicamente la userList, che ritorna l'elenco degli utenti associati all'applicazione. In questo modo l'applicazione può rimuovere dal proprio database gli utenti non più presenti nell'IDP.
- getUser: ritorna lo user data la sua e-mail.
- createUser: crea un nuovo utente nell'IDP e lo associa ad una applicazione ed un ruolo.
- updateUser: aggiorna un utente già esistente.
- deleteUser: rimozione dell'utente.
- changePassword: modifica della password di un utente.
- status: permette di verificare lo stato del servizio IDP.

I diversi metodi utilizzano il token memorizzato o possono essere invocati specificandone uno.

Il servizio di creazione degli utenti accetta il parametro HTTP enc (true | false). Se true, la password dell'utente sarà memorizzata as-is nel database dell'IDP. La password va generata con l'algoritmo di hashing bcrypt, versione $2y$ e costo 10.

3.3 Interfacce HMI

Le interfacce HMI sono utilizzate dall'utente finale. Tramite esse, può autenticarsi e accedere alle applicazioni censite nell'IDP.

3.3.1 Login e SSO

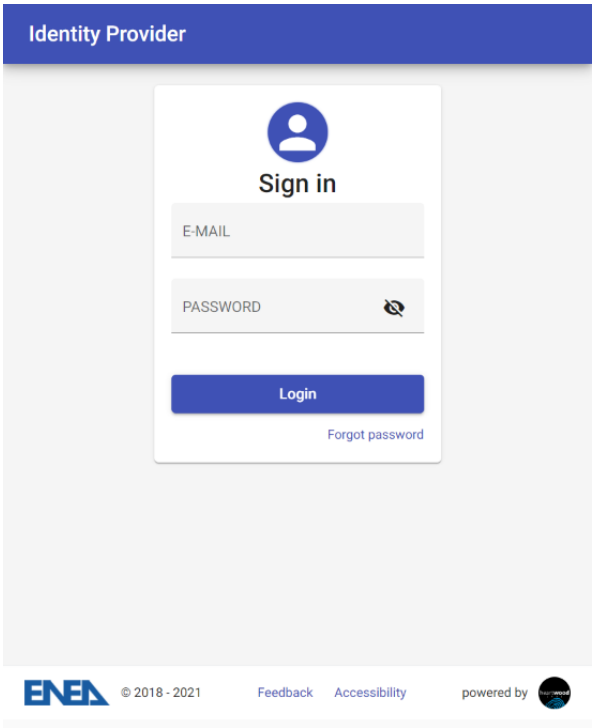

Figura 2 - Login

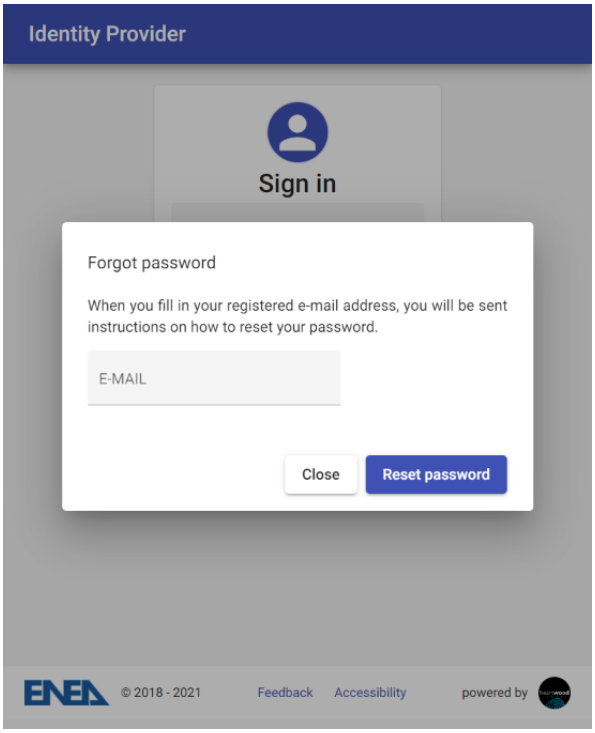

Figura 3 - Reset Password

Nel momento in cui un utente deve essere autenticato dall'applicazione, questo viene rediretto alla pagina di login: https://idp.smartcityplatform.enea.it/#/login/{clientId} (Figura 2) ove può anche richiedere il reset della password (Figura 3)

L'identity provider verifica se è presente nel local storage un token valido (non scaduto) e che l'utente sia abilitato all'accesso del clientId. Se queste condizioni sono verificate, viene attivato il flusso di SSO, in caso contrario viene mostrato il form di login.

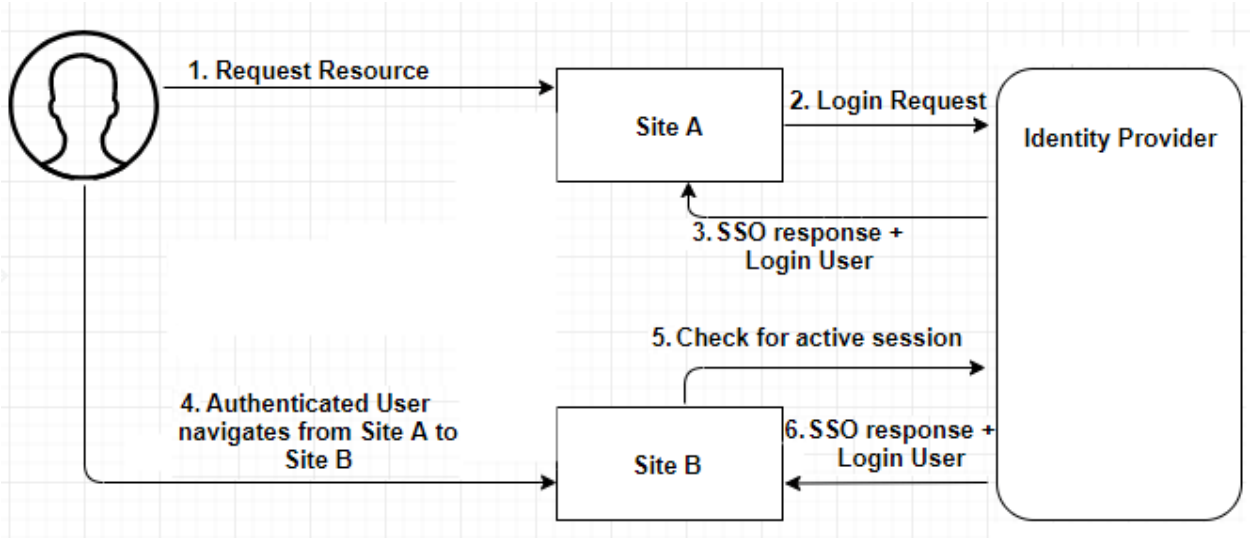

Figura 4 – Flusso autenticazione

A seguito dell'attivazione del flusso di SSO (Figura 4) o di un login avvenuto con successo, l'identity provider invoca (GET) la callback di login del clientId. Viene inviato il token JWT, firmato con il clientSecret.

L'applicazione verifica il token JWT, che contiene: issuer (ENEA), subject (user|userId_IDP), i dati del profilo utente (fra cui l'e-mail), il token da usare nelle chiamate successive, la data di creazione e data di scadenza. È possibile utilizzare la classe TokenManager dell'SDK. L'applicazione infine può effettuare programmaticamente il login dell'utente.

3.3.2 Logout

Il logout è centralizzato. Viene rimosso il token di sessione dell'utente, per cui l'utente viene sloggato da tutte le applicazioni. L'utente viene rediretto all'url: https://idp.smartcityplatform.enea.it/#/logout/{clientId}

L'identity provider invoca (GET) la callback di logout del clientId, a cui passa il token JWT relativo alla sessione appena eliminata. L'applicazione effettua programmaticamente il logout dell'utente.

3.3.3 Signup

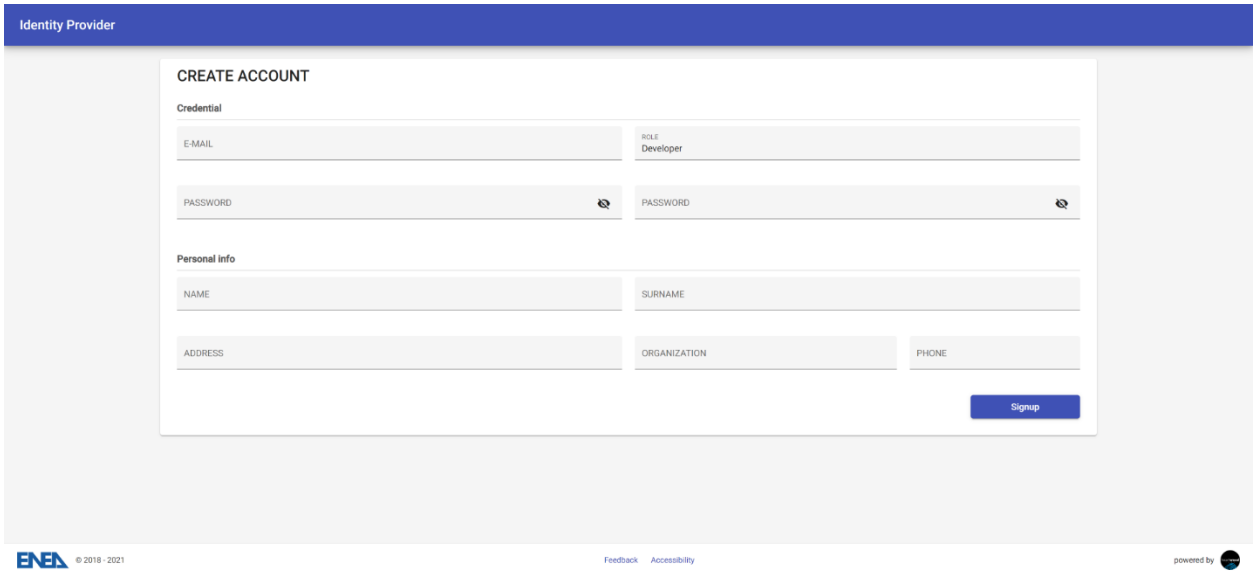

Figura 5 – Registrazione

Per creare un nuovo profilo utente, l'applicazione mostra all'utente la pagina: https://idp.smartcityplatform.enea.it/#/signup/{clientId}/{role} (Figura 5)

L'identity provider, all'inserimento dell'e-mail, verifica se questa è già presente o meno:

- In caso lo sia, il form chiede di inserire solo la password, in modo da associare l'utente al clientId.
- Se l'utente è già associato al clientId, viene mostrato un messaggio che chiede di ritornare all'applicazione di partenza ed effettuare il login.

Al salvataggio dell'utente, l'identity provider invoca (GET) la callback di profile del clientId, a cui invia i dati dell'utente appena registrato (status=create). L'applicazione potrà a questo punto mostrare un proprio form per chiedere all'utente di inserire ulteriori informazioni, peculiari dell'applicazione e non presenti nel form di signup dell'identity provider.

Al salvataggio dell'utente, l'identity provider genera una e-mail contenente un link con cui l'utente potrà validare la propria email. La validazione valorizza a true l'attributo emailConfirmed dell'utente. Questa informazione può essere utilizzata o meno dalle applicazioni per impedire agli utenti non verificati il login o l'accesso a particolari risorse.

3.3.4 Modifica profilo utente

Per permettere all'utente di modificare il proprio profilo, l'applicazione deve mostrare la pagina: https://idp.smartcityplatform.enea.it/#/profile/{clientId}

Al salvataggio dell'utente, l'identity provider invoca (GET) la callback di profile del clientId, a cui invia i dati dell'utente appena modificato (status=update). L'applicazione, come nel caso della signup, potrà mostrare all'utente un proprio form per permettere l'editazione di ulteriori attributi.

3.3.5 Rimozione profilo

La rimozione del profilo utente può essere:

- Relativa all'applicazione: viene rimossa l'associazione fra l'utente e l'applicazione. Se l'utente risulta non più associato ad alcuna applicazione, viene rimosso: https://idp.smartcityplatform.enea.it/#/delete/{clientId}
- Globale: l'utente viene rimosso dal sistema: https://idp.smartcityplatform.enea.it/#/delete-all/{clientId}

Alla rimozione dell'utente, l'identity provider invoca (GET) la callback di profile del clientId, a cui invia i dati dell'utente appena rimosso (status=delete).

Le funzionalità sopra elencate per l'interfaccia HMI fanno uso dei seguenti servizi:

- login: flusso di login dell'utente. Usa l'algoritmo di SSO. Viene generato un token per l'utente che effettua correttamente il login.
- logout: logout dell'utente. Viene rimosso il token dell'utente.
- createUser: crea un nuovo utente nell'IDP e lo associa ad una applicazione ed un ruolo.
- updateUser: aggiorna l'anagrafica dell'utente.
- deleteUser: rimozione dell'utente. La cancellazione è relativa al client id specificato. Nel caso in cui venga rimossa l'ultima associazione fra l'utente e i client, l'utente viene rimosso dall'IDP.
- validateEmail: permette di validare l'e-mail dell'utente a seguito della sua registrazione.
- forgotPassword: attiva il flusso di reset della propria password.
- resetPassword: modifica la password dell'utente in seguito all'attivazione del flusso di reset.

3.4 Interfacce Admin

L'utente con ruolo Developer accede all'identity provider nella propria pagina home (Figura 6) per:

- Censire le applicazioni verticali che dovranno integrarsi con il sistema.
- Modificare il proprio profilo.
- Modificare e creare utenti.
- Creare i ruoli degli utenti dell'applicazione.

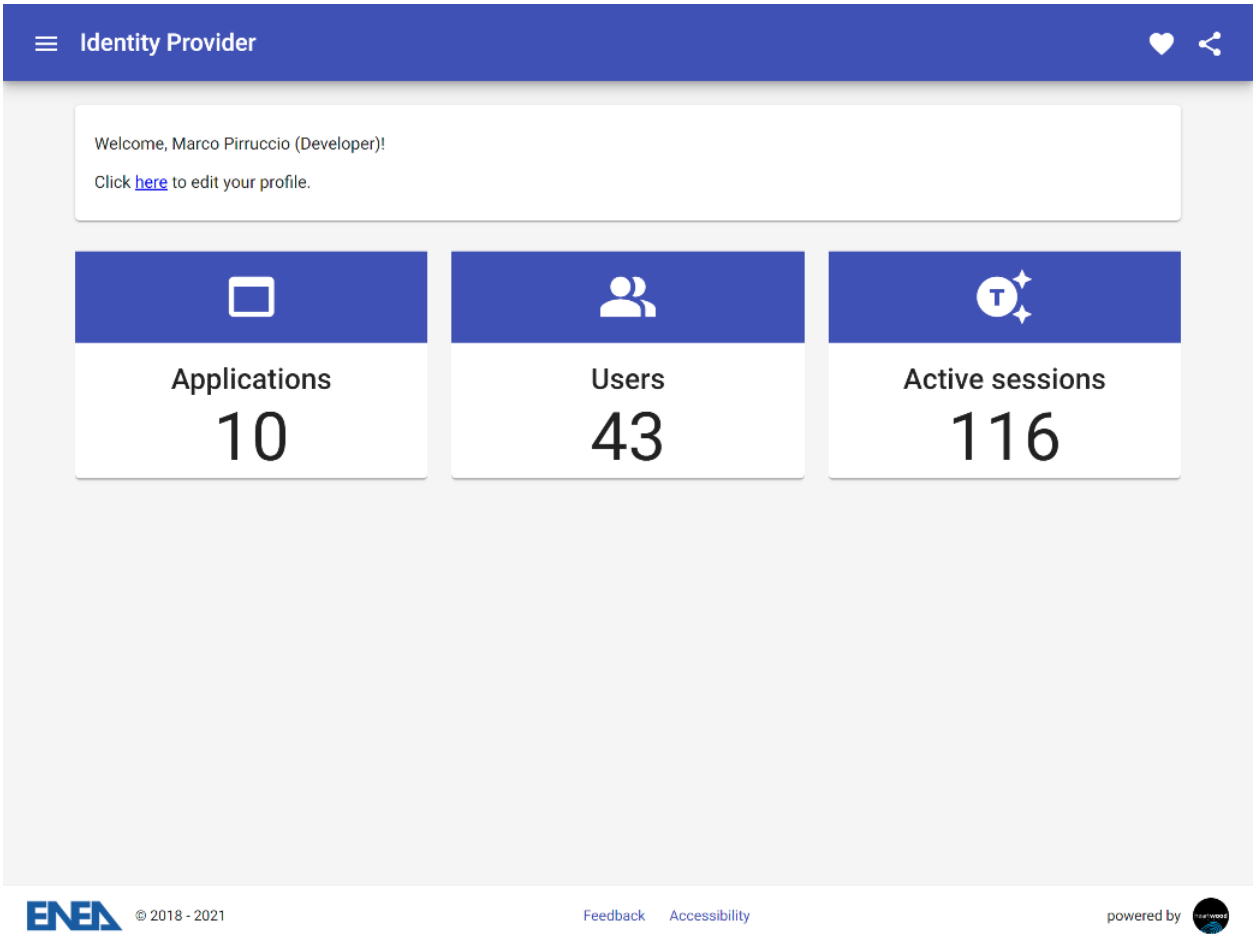

Figura 6 - Pagina home IDP

3.4.1 Censimento di un'applicazione

Affinché un'applicazione possa integrarsi all'IDP, deve essere stata precedentemente creata da un utente Developer (Figura 7).

Se viene censita una SCP (Smart City Platform), va impostato il flag SCP a true. In questo caso, al salvataggio, il sistema:

- Associa questa applicazione a tutti gli utenti già associati all'applicazione SCP-Template.
- Crea un nuovo utente con ruolo M2MAgent, utilizzabile per invocare i servizi server-to-server.
    - L'utilizzo di questo utente è opzionale e dipende dal tipo di applicazione che viene sviluppata. In ogni caso è possibile creare un utente con questo ruolo in un secondo momento.
    - Le credenziali di questo utente vanno comunicate dal Developer al system integrator, che dovrà memorizzarle all'interno della propria applicazione.

Client id e client secret: sono generati dall'IDP e devono essere utilizzati dall'applicazione che si integra con l'IDP, sia nei flussi HMI che in quelli M2M.

La sezione callbacks permette di specificare gli URL che vengono utilizzati dall'IDP per notificare all'applicazione gli eventi:

- Login: l'utente ha effettualo il login. L'IDP effettua una GET all'url indicato, inviando il token JWT, firmato con il clientSecret. L'applicazione potrà leggere il token ed effettuare programmaticamente il login dell'utente.
- Logout: l'utente ha effettualo il logout. L'IDP effettua una GET all'url indicato, inviando il token JWT relativo alla sessione appena rimossa. L'applicazione potrà leggere il token ed effettuare programmaticamente il logout dell'utente.
- Profile: questo URL viene invocato per tutti i flussi che comportano l'editazione del profile dell'utente:
    - Signup: creazione di un nuovo profilo. L'IDP invoca (GET) la callback a cui invia i dati dell'utente appena registrato (status=create).
    - Modifica profilo. L'IDP invoca (GET) la callback a cui invia i dati del profilo utente modificato (status=update).
    - Rimozione profilo. L'IDP invoca (GET) la callback a cui invia i dati dell'utente appena rimosso (status=delete).

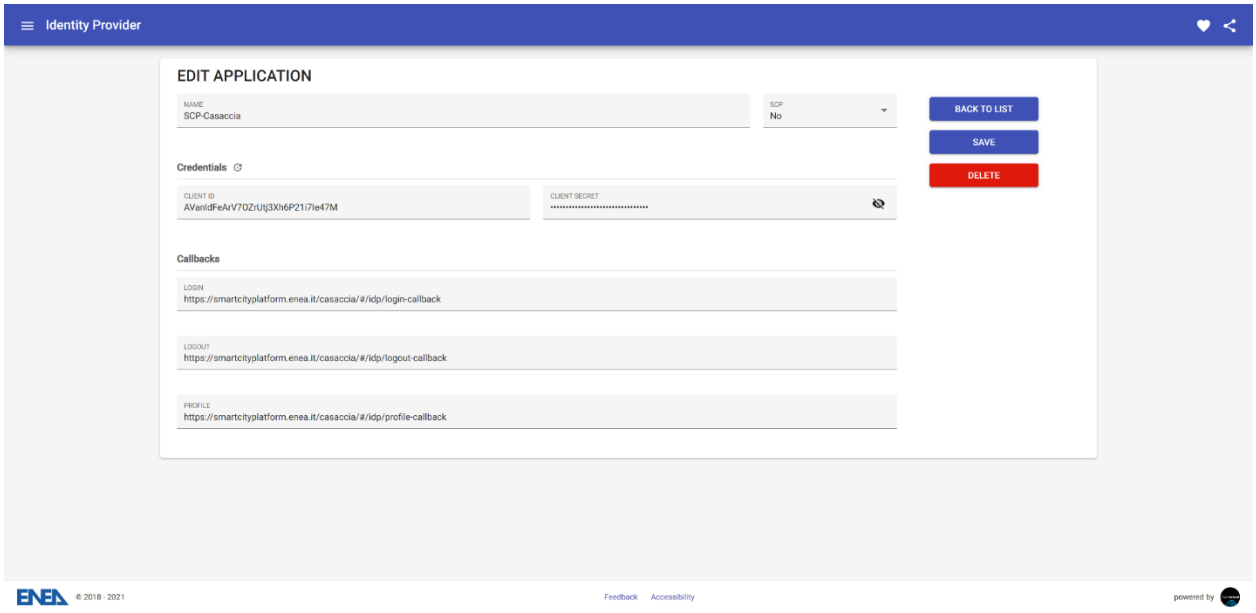

Figura 7 - Censimento Applicazione

3.4.2 Censimento di un utente

Oltre ai dati anagrafici dell'utente, tramite questa interfaccia (Figura 8) è possibile specificare:

- E-mail e password, che rappresentano le credenziali dell'utente
- Il ruolo (Figura 9)
- Le applicazioni a cui è abilitato

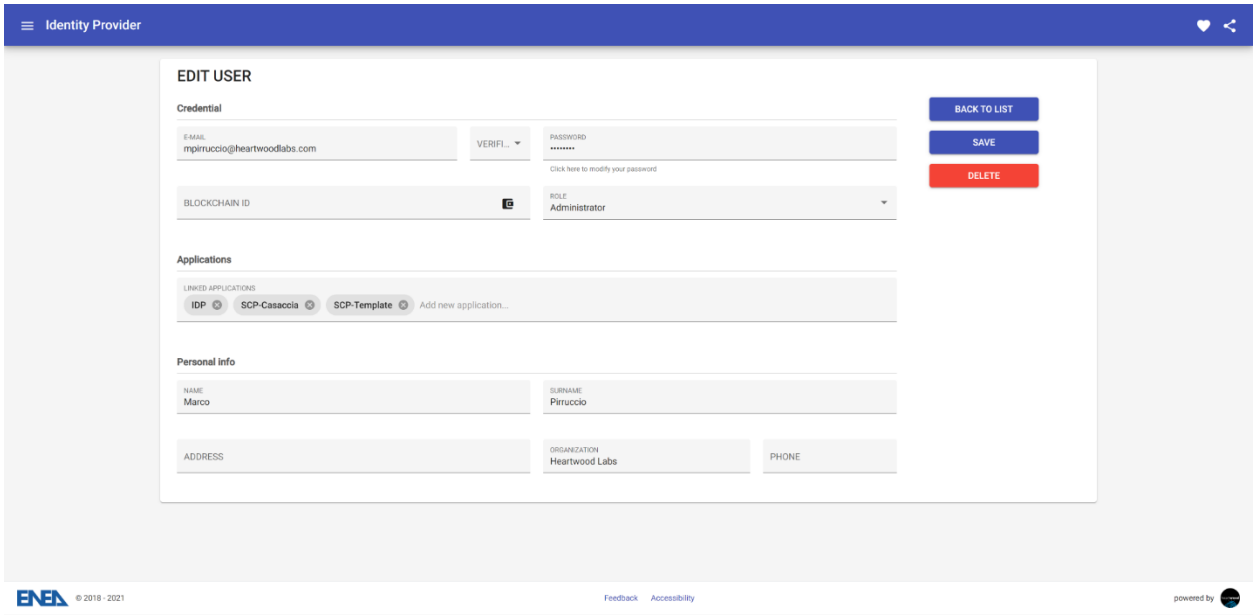

Figura 8 - Modifica Utente

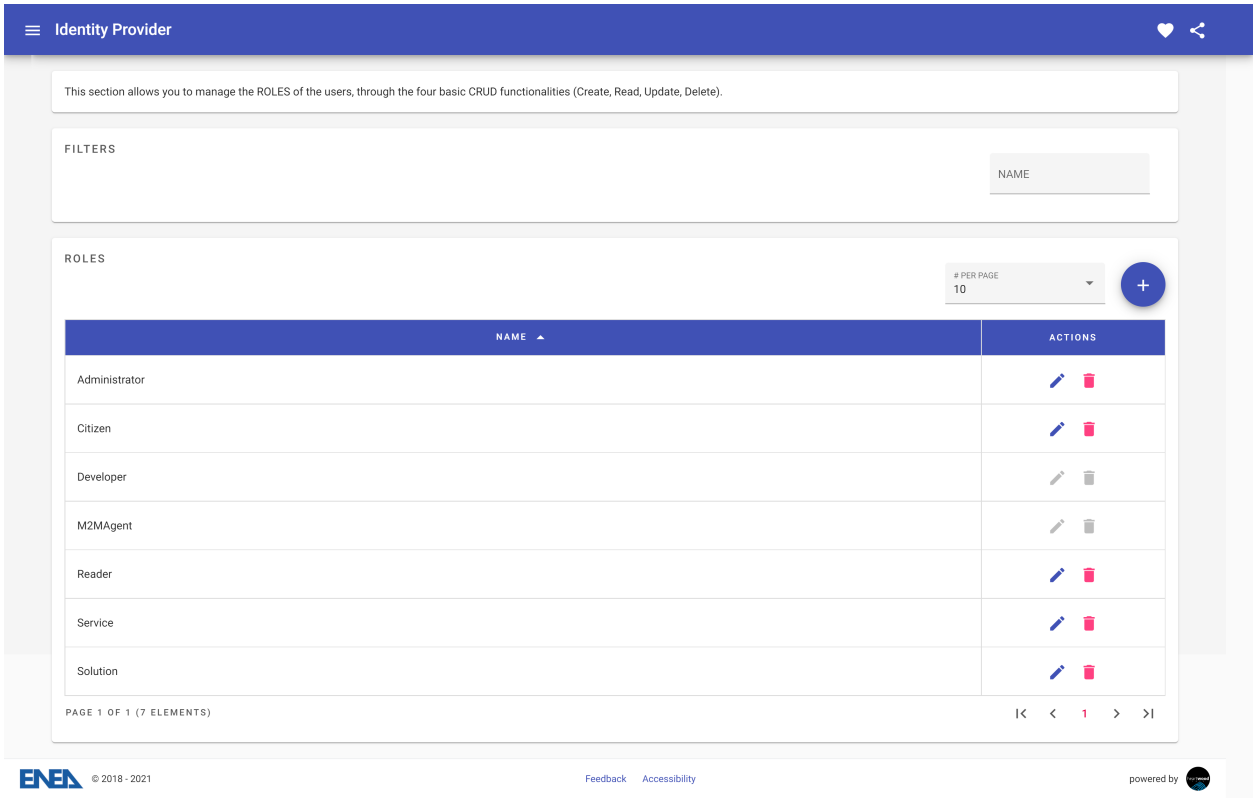

Figura 9 - Modifica Ruoli

Le funzionalità sopra elencate per l'interfaccia Admin fanno uso dei seguenti servizi:

- Gestione utenti:
  - user: ritorna l'utente indicato.
  - userList: ritorna la lista degli utenti associati ad una certa applicazione identificata dal proprio client id.

  - createUser: crea un nuovo utente nell'IDP e lo associa ad una applicazione ed un ruolo.
  - updateUser: aggiorna l'anagrafica dell'utente.
  - deleteUser: rimozione dell'utente. La cancellazione rimuove tutte le associazioni fra l'utente e i client, e infine rimuove l'utente.
- Gestione ruoli:
  - role: ritorna il ruolo indicato.
  - roleList: elenco dei ruoli presenti dell'IDP.
  - createRole: creazione di un nuovo ruolo.
  - updateRole: aggiornamento del ruolo.
  - deleteRole: cancellazione del ruolo.
- Gestione client o applicazioni:
  - client: ritorna il client indicato.
  - clientList: elenco dei client presenti dell'IDP.
  - createClient: creazione di un nuovo client.
  - updateClient: aggiornamento del client.
  - deleteClient: cancellazione del client.

3.5 Open API

Le API dell'identity provider sono esposte tramite OpenAPI. È fornito un minisito tramite cui interagire con i diversi endpoint (Figura 10). È accessibile all'URL: https://idp.smartcityplatform.enea.it/swagger/

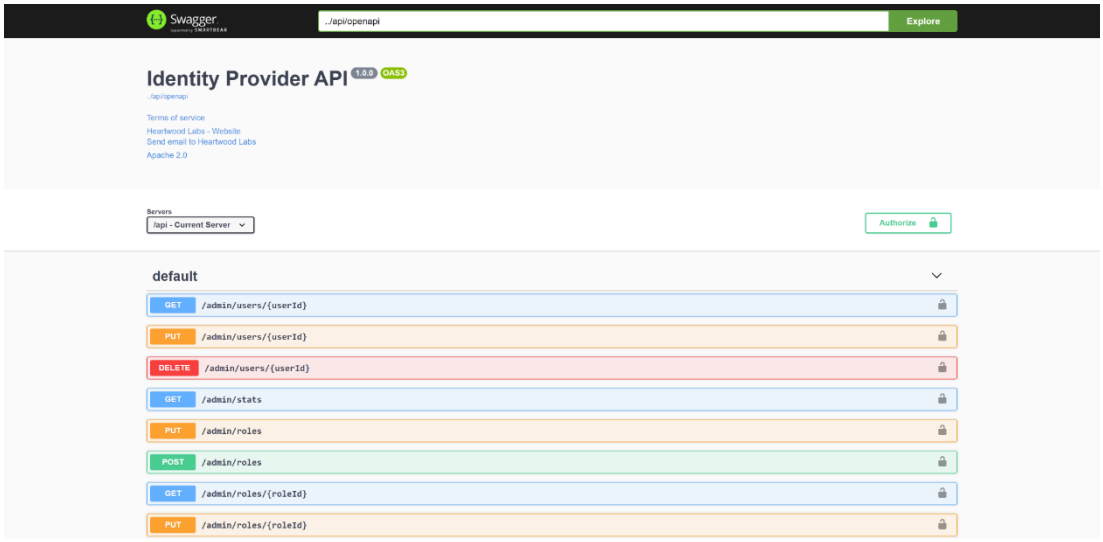

Figura 10 - API Provider

3.6 Accessibilità

Le interfacce HTML, sia quelle di gestione, riservate agli amministratori, sia quelle per gli utenti che effettuano la registrazione, login, etc. nell'IDP, rispettano i requisiti di accessibilità richiesti da AGID. La verifica di tali requisiti è stata verificata tramite i validatori:

- Vamolà: http://www.validatore.it/vamola_validator/checker/index.php
- WAVE Web Accessibility Evaluation Tool: https://wave.webaim.org

Il footer delle diverse pagine contiene un link verso l'informativa fornita da ENEA.

3.7 Privacy e GDPR

Per quanto riguarda la privacy degli utenti e il GDPR (Garante per la protezione dei dati personali), l'IDP fornisce un sistema di speudoanonimizzazione dei dati identificativi trattati. In particolare, seguendo quanto indicato dall'ENISA (European Union Agency for Cybersecurity) in "Recommendations on shaping technology according to data protection and privacy provisions", sono state seguite le linee guida indicate per la pseudoanonimizzazione dei dati raccolti direttamente dagli utenti.

Le tabelle dell'anagrafica degli utenti, quindi, contiene nella colonna dell'e-mail un valore anonimizzato tramite RNG (random number generator). È inoltre presente una ulteriore tabella in cui è indicata l'associazione fra tale valore e l'e-mail.

La password degli utenti è crittografata con bcrypt.

Inoltre, il footer delle diverse pagine contiene un link verso l'informativa della privacy fornita da ENEA.

3.8 Tecnologie adottate

Dal punto di vista tecnologico, l'IDP è realizzato tramite i seguenti linguaggi, tecnologie e framework:

- Interfacce HTML tramite Angular e Sass. Il processo di build è realizzato tramite la CLI di Angular.
- Java 8 per i servizi RESTful, il ciclo di vita delle entità, la persistenza dei dati e le integrazioni con le piattaforme esterne. I test di unità sono implementati tramite JUnit. Il processo di build è realizzato tramite Maven.
- Persistenza dei dati su MySQL, file properties e file di log.

Il sito relativo alla documentazione è realizzato tramite Swagger.

3.9 Codici sorgente

I sorgenti sono disponibili su GitLab e accessibili alle utenze ENEA dei laboratori TERIN-SEN-SCC e TERIN-SEN-CROSS:

- Componente server: https://gitlab.com/marcopirruccio/scp-idp-server
- Componente client: https://gitlab.com/marcopirruccio/scp-idp-client
- SDK: https://gitlab.com/marcopirruccio/scp-idp-sdk

4 Task 2 – Aggiornamento SCP

4.1 Integrazione SCP-GUI

A seguito del rilascio dell'identity provider, la gestione degli utenti della SCP-GUI (Graphic User Interface) è stata riscritta in modo da adottare il nuovo sistema.

Il processo di integrazione ha comportato l'adozione dell'SDK dell'identity provider, che è stato aggiunto come dipendenza del progetto SCP-GUI, componente server, nel pom.xml.

Dal registry sono state rimosse, dalla tabella user, le colonne name, surname, organization, phone, address, adesso gestite dall'identity provider.

Ogni istanza di SCP viene così censita nell'IDP e viene associata agli attributi clientId e clientSecret, generati dall'IDP al momento dell'inserimento. Queste informazioni sono state memorizzate nel file scp-gui.properties, in modo da essere utilizzate nei flussi di integrazione con l'IDP.

Viene generata una utenza con ruolo M2MAgent, associata alla SCP-GUI, utilizzata per consumare le API M2M. Le credenziali di questo utente di sistema sono salvate nel file scp-gui.properties.

Sono stati quindi integrati i flussi HMI e M2M.

I flussi HMI hanno comportato la creazione di 3 URL di callback, utilizzati per ricevere le seguenti notifiche da parte dell'identity provider:

- Login: /#/idp/login-callback
- Logout: /#/idp/logout-callback
- Editazione profilo utente: /#/idp/profile-callback

È inoltre stato aggiunto il supporto al flusso di SSO dell'identity provider.

Le API M2M sono invece state utilizzate dai servizi di editazione degli utenti da parte degli amministratori della SCP-GUI:

- Lista degli utenti: al fine di generare tale lista è stata rimossa la colonna contenente gli attributi name e surname di User (Figura 11), mantenendo le altre informazioni reperite dal Registry.
- Scheda utente e scheda solution (Figura 12): nella sezione è stata data evidenza che le informazioni lì presenti non sono gestite localmente, ma dall'IDP. Al salvataggio, questi dati sono inviati all'IDP tramite il metodo updateUser dell'SDK.

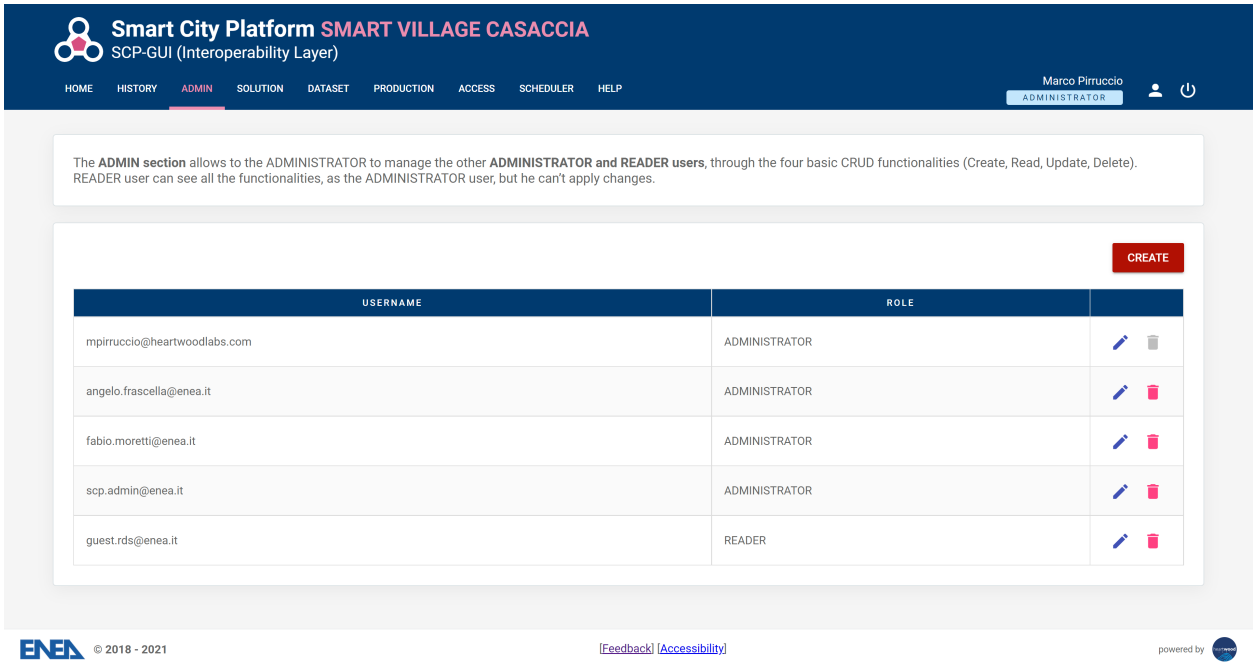

Figura 11 - Livello Interoperabilità Admin

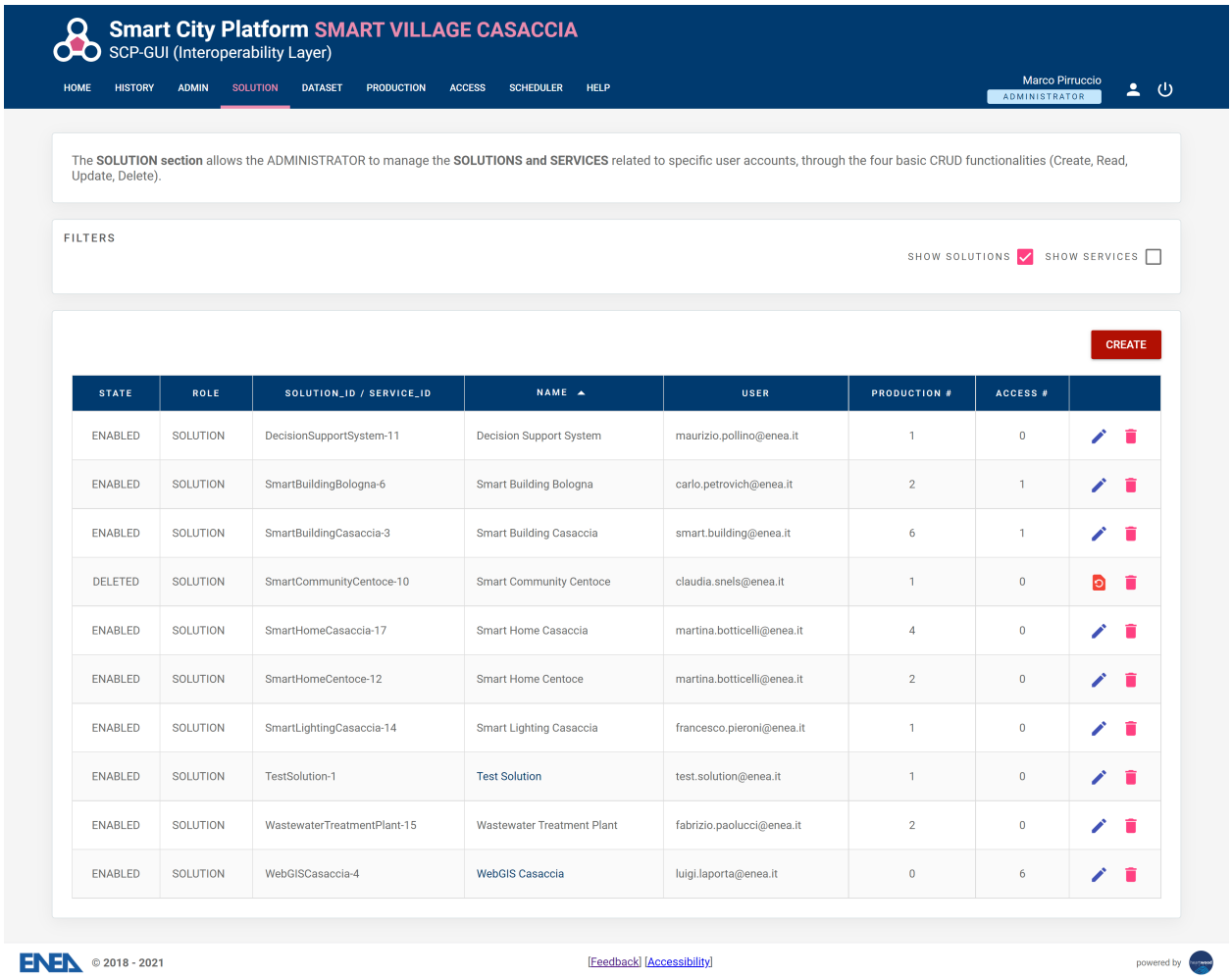

Figura 12 - Livello Interoperabilità Soluzioni

È stato inoltre aggiunto un servizio batch temporizzato, eseguito periodicamente, che di occupa di tenere sincronizzate le anagrafiche utenti della SCP-GUI con quelle dell'IDP. Questo servizio fa uso del metodo userList presente nell'SDK e si occupa di rimuovere dal registry utenze non più presenti nell'IDP poiché rimosse in flussi esterni alla SCP-GUI.

I sorgenti del componente sono inclusi nei progetti della SCP-GUI e sono accessibili su GitLab, nel branch master:

- https://gitlab.com/marcopirruccio/enea-gui
- https://gitlab.com/marcopirruccio/enea-gui-client

4.2 Integrazione UD Gateway

Il componente UD Gateway espone i servizi di login e logout, usati dai client per autenticarsi prima di effettuare le operazioni di request e push.

Anche questo componente, quindi, è stato integrato con il nuovo identity provider utilizzando le API M2M tramite SDK. Il processo di integrazione ha comportato l'adozione dell'SDK dell'identity provider, che è stato aggiunto come dipendenza del progetto nel pom.xml (Figura 13).

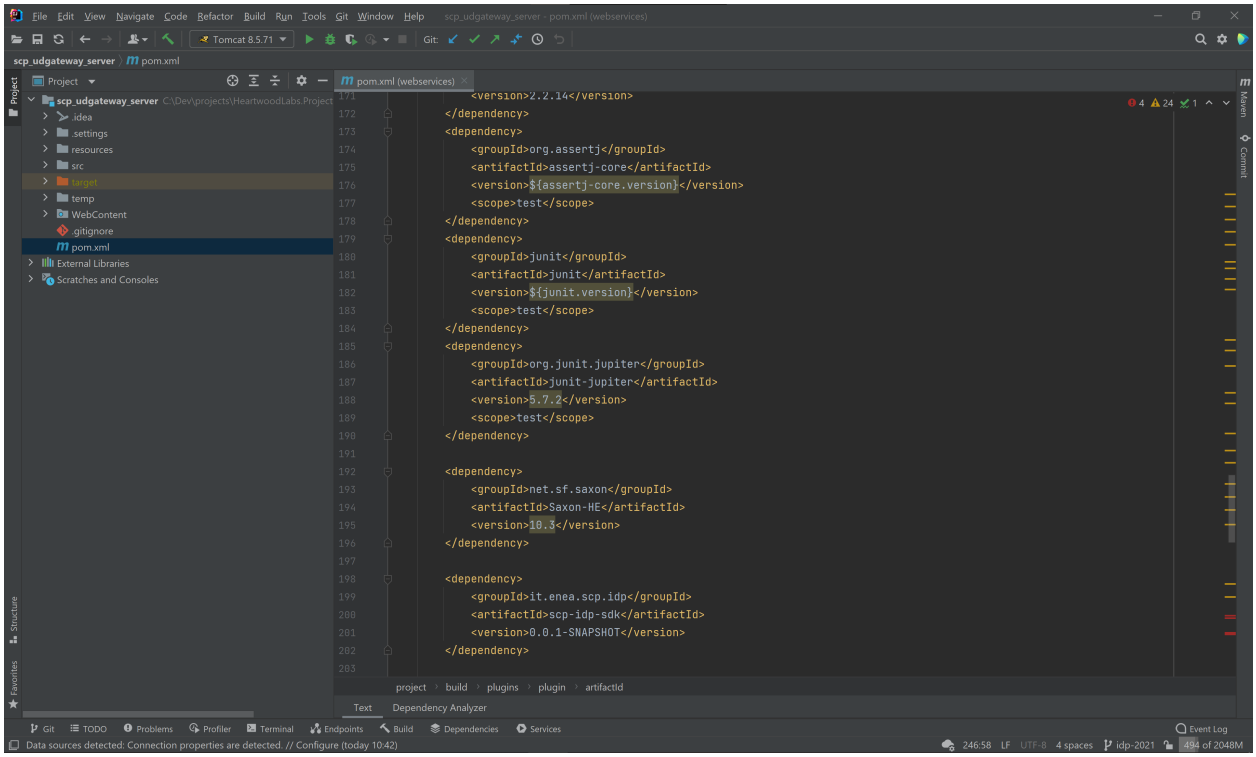

Figura 13 - Esempio integrazione

L'UD Gateway è adesso censito nell'IDP e viene associato agli attributi clientId e clientSecret, generati dall'IDP al momento dell'inserimento e usati nei flussi di seguito descritti.

Gli attuali metodi di login e logout sono stati sostituiti da quelli esposti dall'SDK.

I sorgenti dell'UD Gateway sono accessibili sull'istanza GitLab presso ENEA: http://crossserv.bologna.enea.it/gitlab/cristiano.novelli/scp_udgateway_client_core

5 Task 3 – Modulo Data Fusion per gestione flessibilità energetica

Il modulo di Data Fusion fa uso dei servizi presenti all'interno della SCP. In particolare, è stato istanziato il progetto scp-job-template (Figura 14), in modo da implementare le funzionalità sfruttando il componente SCP-SCHEDULER.

Il modulo è stato implementato estendendo la classe it.enea.scp.scheduler.job.ScpJob esposta dallo SCHEDULER. In questo modo il modulo viene reso disponibile al componente SCHEDULER e può essere configurato e schedulato (invocato periodicamente). I metodi del ciclo di vita del modulo che sono stati implementati sono:

- void init(ScpJobConfiguration configuration): fornisce al modulo i dati di inizializzazione:
  - Elenco degli UD di input
  - Parametri di configurazione
- Optional<List<UD>> execute(): contiene la logica specifica del modulo di Data Fusion, di seguito descritta.
- void shutdown(): chiamato al termine del ciclo di esecuzione.

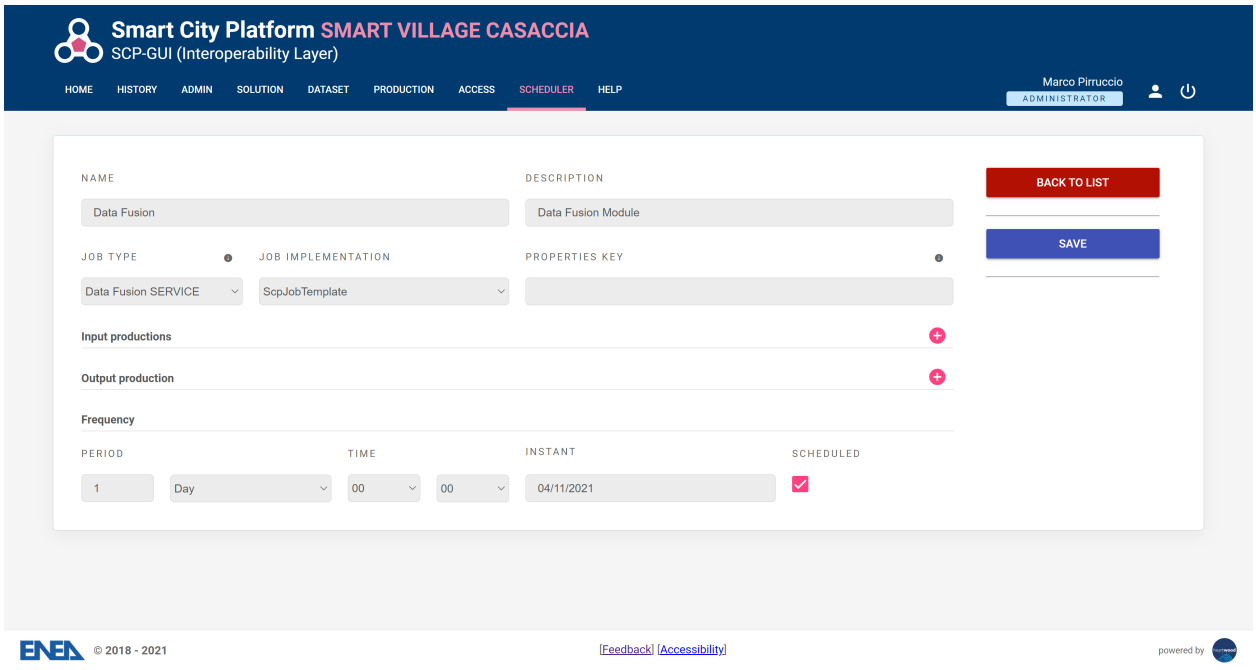

Figura 14 - Implementazione Job

L'interazione fra il componente SCHEDULER e il modulo consiste nelle seguenti fasi, dipendentemente dalla configurazione eseguita (Figura 15):

- Lo SCHEDULER effettua una lastRequest verso l'UD Gateway, chiedendo l'urban dataset LastFlexUD.
- Viene invocato il metodo execute del modulo, a cui vengono passati gli UD ricevuti al passo precedente.

- Il modulo calcola la flessibilità potenziale complessiva della comunità a cui gli UD ottenuti al passo precedente afferiscono. Ritorna il nuovo UD allo SCHEDULER.
- Lo SCHEDULER effettua una push verso l'UD Gateway in modo da pubblicare un nuovo UD contenente tali aggregazioni.

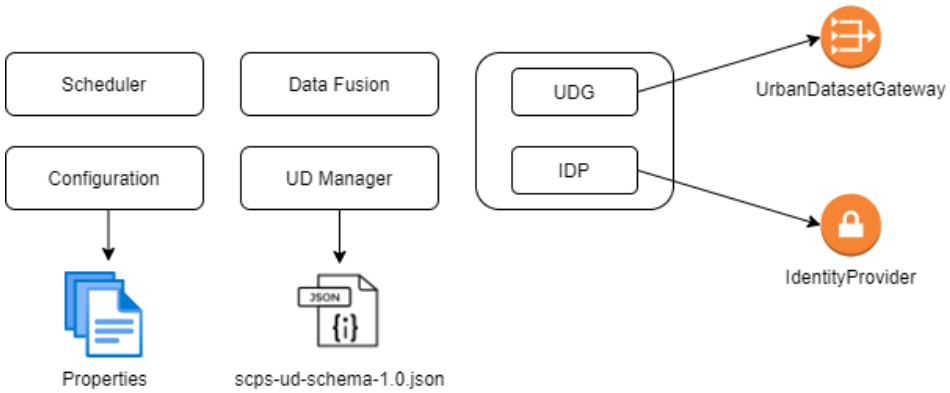

Figura 15 - Fasi dello Scheduler

6 Task 4 – Sviluppo KPI

Per finalizzare le strategie poste nel progetto che mirano a misurare e quantificare l'efficientamento energetico degli edifici, è stato implementato un KPI (Indicatore di prestazione chiave) accessibile da portale attraverso un servizio RESTful, rispettando le policy di sicurezza e profilazione previsti dalla piattaforma.

Il KPI è alimentato da un ´dataframe´ di un file strutturato che ha seguito una sequenza di pre-processing e cleaning dei dati, successivamente un ulteriore processo si occupa del calcolo del singolo KPI che viene storicizzato su un database relazionale MySQL ed un servizio RESTful rende disponibile la lettura del KPI dai sistemi GUI e di presentazione esterni.

Gli step di implementazione eseguiti sono i seguenti:

1. connessione al dataframe su piattaforma Hadoop
2. lettura ed aggregazione dei dati con PySpark
3. calcolo del KPI con PySpark
4. definizione della frequenza di analisi, streaming o batch, ed implementazione
5. storicizzazione dei dati del KPI su database relazionale MySQL
6. esposizione web service del KPI con framework Flask

6.1 Descrizione del KPI Premialità per autoconsumo (Caso PROSUMER)

Il KPI ha come obiettivo incentivare il consumo di energia autoprodotta dagli impianti Fotovoltaici dei complessi edilizi con il massimo consumo in loco dell'energia prodotta dal singolo utente o dalla comunità a cui questo appartiene. Si vuol far leva sul risparmio sia economico che ambientale in termini di mancate emissioni di CO2.

La struttura del dataframe elaborato è costituita dai seguenti valori, per le unità di misura vedasi la Tabella 1:

Consumo elettrico al tempo i: $E_{i\,load}$:[kWh] dell'utenza;

Produzione elettrica $E\,prod_i$= energia prodotta dal Fotovoltaico :[kWh] ;

Tabella 1 - Unità di misura

Consumo elettrico al tempo i	kWh	Eload
Produzione elettrica	kWh	Eprod
Identificativo edificio	int	ID
Data rilevamento lettura	Timestamp	TS

Tabella 2 - Algoritmo KPI

Algoritmo/formula per il calcolo
min (E $load_i$; E$prod_i$) con: E $load_i$= energia assorbita dai carichi nel quarto d'ora (*kWh*) E $prod_i$= energia prodotta dal Fotovoltaico nel quarto d'ora (kWh) i = 15 minuti

6.2 Calcolo del KPI

È stato implementato un batch in pySpark con esecuzione ogni 15 minuti che si occupa della lettura dei dati di tutti gli edifici con una connessione ad Hadoop e della storicizzazione del risultato del calcolo del KPI con i seguenti step (sono presenti solo stralci del codice completo).

Caricamento degli ID e lastUpdate da MySQL

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

df = sqlContext.read.format('jdbc').options(url='jdbc:mysql:dbEnergy',
dbtable='(select ID,lastUpdate from inventory) as inventory').load()
```

Caricamento delle librerie necessarie ed accesso ad Hadoop

```
from pyspark import SparkFiles

spark.sparkContext.addFile('hdfs:///test/kpi_1/data_file.json')
with open(SparkFiles.get('data_file.json'), 'rb') as handle:
    j = json.load(handle)
    or_do_whatever_with(handle)
```

Scorrimento dei dati

Per ogni ID rilevato nel dbtable viene fatto scorrere il file json e memorizzato ogni valore Eprod, Eload e Timestamp successivi a lastUpdate, nel loop per ogni ID vengono aggregati i valori di Eprod e Eload ad intervalli di 15 minuti.

Se i restanti valori provenienti dal Json non coprono l'intervallo di 15 minuti non vengono utilizzati e viene aggiornata la variabile di lastUpdate con l'ultimo valore utile dell'ultimo intervallo completo.

Calcolo e storicizzazione del KPI

Ottenuto il nuovo dataframe con i dati aggregati di Eprod e Eload ogni 15 minuti per ogni ID viene applicata la formula di calcolo min(Eload, Eprod) e storicizzata

```
df = sc.parallelize(data).toDF(['ID', 'kpi', 'TS' ])


df.write.format('jdbc').options(
      url='jdbc:mysql://localhost/dbEnergy',
      driver='com.mysql.jdbc.Driver',
      dbtable='kpiTable',
      user='*******',
      password='*******').mode('append').save()
```

Il processo può essere rappresentato dallo schema in Figura 16:

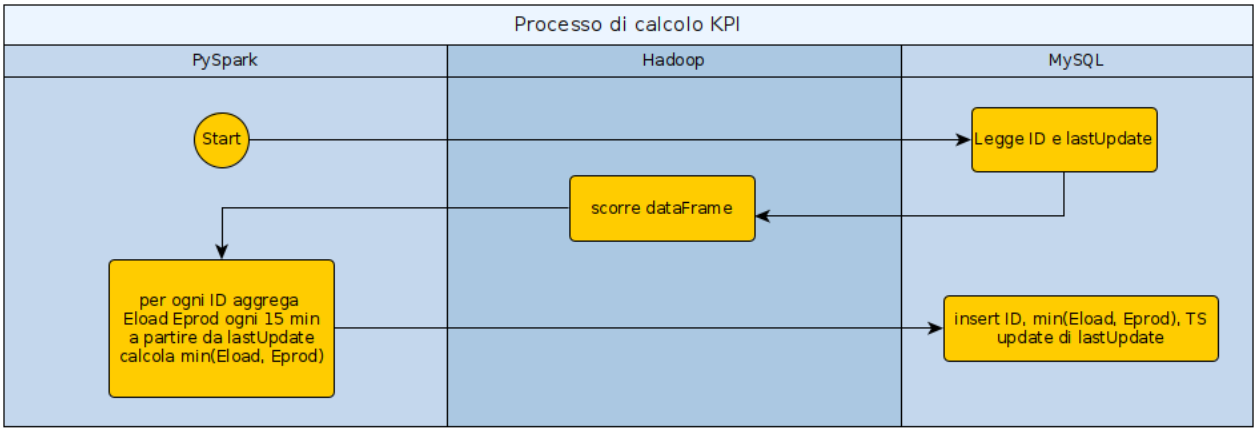

Figura 16 - Schema Processo KPI

6.3 Presentazione dei dati con servizio REST

È stato implementato un servizio di interrogazione dei KPI sfruttando il framework Python Flask che permette di realizzare servizi REST ed il framework SQLAlchemy per l'accesso alle funzioni CRUD sul database relazionale MySQL.

Per facilitare le operazioni di configurazione e debug si è scelto di suddividere le informazioni di configurazione dal resto del programma attraverso un file config.py

```
# config.py

class Config(object):
    """
    Common configurations
    """

class DevelopmentConfig(Config):
    """
    Development configurations
    """

    DEBUG = True
    SQLALCHEMY_ECHO = True

class ProductionConfig(Config):
    """
    Production configurations
    """

    DEBUG = False

app_config = {
    'development': DevelopmentConfig,
    'production': ProductionConfig
}

SECRET_KEY = '<^>YOUR_SECRET_KEY^>'
SQLALCHEMY_DATABASE_URI = 'mysql://******:******@localhost/dbEnergy'
```

L'init della applicazione si occupa di eseguire l'import delle librerie necessarie e di inizializzare l'app con la corretta configurazione

```
# app/__init__.py

# third-party imports
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

# local imports
from config import app_config

# db variable initialization
db = SQLAlchemy()

def create_app(config_name):
    app = Flask(__name__, instance_relative_config=True)
    app.config.from_object(app_config[config_name])
    app.config.from_pyfile('config.py')
    db.init_app(app)

    return app
```

Il core dell'API REST è rappresentato dalla funzione che interroga il database con il framework SQLAlchemy che evita ogni tentativo di attacco di SQL Injection, benchè si utilizzi il solo verbo GET è stata fatta una scelta di secure coding by design.

```
def read_one(id):
    """

    Questa funzione risponde alla request su /api/kpi/{id}
    :param id:   ID dell'edificio o complesso energetico
    :return:            ultimo KPI riferito ad ID
    """


    kpi = kpiTable.query \
        .filter(kpiTable.ID == id) \
        .one_or_none()

    # verifica l'esistenza del kpi
```

```
    if kpi is not None:
        # Serializza i dati
        kpi_schema = KPISchema()
        return kpi_schema.dump(kpi).data

    # Altrimenti se non corrisponde un kpi all'ID
    else:
        abort(404, 'KPI not found for ID: {id}'.format(id=id))
```

La risposta alla request /api/kpi/{id} è rappresentata dal seguente JSON:

```
{
      "id" = 12,
      "kpi" = 132,
      "ts" = 1642617376
}
```

Il servizio REST implementato non necessita di particolari sistemi di autenticazione poiché è un servizio machine to machine implementato nel medesimo segmento di rete delle web application che lo invocano e non risulta esposto.

7 Task 5 - Test, manutenzione, coaching-on-the-job

Sono state erogate le ore di formazione remota indicate nell'allegato tecnico ENEA, relativamente all'installazione e configurazione dei componenti SCP-IDP, SCP-GUI e SCP-SCHEDULER. Tali attività sono state opportunamente conteggiate dai referenti ENEA. In particolare:

- Sessione di demo di utilizzo dell'IDP per utenti Developer
- Configurazione delle istanze di SCP al fine di adottare l'IDP
- Formazione verso fornitori ENEA al fine di integrare l'IDP tramite tecnologie eterogenee
- Migrazione di alcune istanze di SCP-GUI
- Sono state erogate le ore di formazione remota indicate nell'allegato tecnico ENEA, relativamente all'installazione e configurazione dei componenti DUMP-DB-MYSQL, BATCH-PYSPARK, REST-SERVICE.

8 Conclusioni

Il lavoro di implementazione svolto nell'ambito di questo contratto è propedeutico alle successive fasi di sviluppo dei servizi che saranno implementati nel triennio successivo per la finalizzazione del portale LEC.

9 Riferimenti bibliografici

I riferimenti bibliografici devono essere richiamati nel testo con numeri progressivi tra parentesi quadre e riportati a fine testo con il seguente formato:

1. A. Autore, B. Autore, "Titolo dell'articolo", Rivista o Libro, n. volume (anno), pp. iniziale-finale. Nel caso di libri seguono Editore, Luogo di pubblicazione.

Nel caso di contributi a convegni:

1. A. Autore, "Titolo del contributo", titolo degli atti, data e luogo del convegno, editore, anno di pubblicazione, pp. iniziale-finale.

10 Abbreviazioni ed acronimi

Acronimo	*Significato*	*Breve Descrizione*
IDP	IDentity Provider	Sistema integrato per la gestione delle identità digitali
SDK	Software Development Kit	Insieme di librerie e programmi che permetto l'interazione con una soluzione software
GUI	Graphic User Interface	Interfaccia grafica che permette agli utenti di interagire con la soluzione software
KPI	Key Perfonmance Indicators	indicatori che riflettono i fattori critici di successo per un'organizzazione
M2M	Machine To Machine	Comunicazione tra soluzioni software attraverso la condivisione di un protocollo
CRUD	Create Read Update Delete	Funzioni fondamentali di interazione con un database o sistema applicativo
API	Application Programming Interface	librerie di funzioni che permettono al programmatore di interagire con un programma o una piattaforma software

HTML	HyperText Markup Language	Linguaggio ipertestuale per il trasferimento di contenuti via rete Internet
HTTP	Hypertext Transfer Protocol	è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web
REST	REpresentational State Transfer	interfaccia di programmazione che usa HTTP per gestire dati remoti
SQL	Structured Query Language	Linguaggio di interrogazione dei Database
SCP	Smart City Platform	Progetto ENEA con la finalità di mettere a disposizione dei cittadini, delle municipalità e dei diversi stakeholder un valido strumento in grado di raccogliere i dati dalla città e armonizzarli attraverso un linguaggio comune,