



Ricerca di Sistema elettrico

Sviluppo SCP-Scheduler per l'esecuzione di Service nella SCP

Marco Pirruccio (Heartwood Labs SRLS)

SVILUPPO SCP-SCHEDULER PER L'ESECUZIONE DI SERVICE NELLA SCP

Marco Pirruccio (Heartwood Labs SRLS)

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero della 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: 1.21 Sperimentazione del Framework per la Governance dei Dati Urbani Energetici

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 *"Servizio per lo sviluppo di un componente software denominato SCP-SCHEDULER, per la gestione ed esecuzione di processi nella Smart City Platform"*

Responsabile Unico del Procedimento ENEA: Stefano Sylos Labini

Responsabile del Contratto per il Contraente: Marco Pirruccio

Indice

SOMMARIO	4
INTRODUZIONE	5
1 DESCRIZIONE DELLE ATTIVITÀ SVOLTE E RISULTATI	6
1.1 TASK "ANALISI E PROGETTAZIONE"	6
1.1.1 <i>Analisi SERVICE 1: "Exporter"</i>	7
1.1.2 <i>Analisi SERVICE 2: "Data Fusion"</i>	8
1.1.3 <i>Analisi SERVICE 3: "Get Remote UD"</i>	9
1.1.4 <i>Analisi SERVICE 4: "Push Remote UD"</i>	10
1.1.5 <i>Dall'analisi al risultato</i>	11
1.2 TASK "IL COMPONENTE SCP-SCHEDULER"	12
1.3 TASK "SERVICE TEMPLATE"	16
1.4 TASK "MODULI SERVICE"	18
1.5 TASK "SCP-DASHBOARD"	20
1.6 TASK "SCP-GUI"	21
1.7 TASK "FORMAZIONE"	22
1.8 TECNOLOGIE E ACCESSIBILITÀ	22
2 CONCLUSIONI	23
3 RIFERIMENTI BIBLIOGRAFICI	24
4 ABBREVIAZIONI ED ACRONIMI	24

Sommario

L'attività di consulenza richiesta si focalizza principalmente nella progettazione e sviluppo di un componente software denominato "SCP-SCHEDULER", modulo che verrà inserito nel prototipo di piattaforma SCP di ENEA per permettere:

- lo scheduling di processi interni denominati SERVICE o JOB (p.es. data fusion);
- l'invio e il recupero di UrbanDataset (UD) da altre piattaforme.

Nel report in oggetto si farà direttamente riferimento all'Allegato Tecnico "SCP-Scheduler" definito da ENEA per questo contratto (tale Allegato Tecnico è riportato nel report LA21, riferimento RdS/PTR(2021)/013 [1]).

Riprendiamo e riportiamo un frammento di tale Allegato Tecnico in cui si descrivono i requisiti fondamentali individuati dall'analisi effettuata da ENEA:

1. *l'esigenza di uno Scheduler (backend) che permetta di configurare e gestire i moduli SERVICE, consentendone il discovery e l'esecuzione periodica;*
2. *la necessità di un'interfaccia utente web per lo Scheduler (frontend) per permettere la gestione dello scheduler e quindi la configurazione dei moduli SERVICE;*
3. *un modulo "SERVICE Template" che contiene la logica del caso generale. Tale "SERVICE Template" è predisposto per fare tutte o alcune di queste azioni:*
 - a. *recuperare UrbanDataset localmente o da remoto;*
 - b. *eseguire operazioni di calcolo o di datafusion;*
 - c. *esportare UrbanDataset e inviarli a un UrbanDatasetGateway locale o remoto;*
4. *la necessità di implementare gli algoritmi di moduli SERVICE specifici.*

Oltre a questi aspetti, l'attività di sviluppo ha previsto anche interventi relativi i componenti SCP-GUI e SCP-Dashboard che, proprio grazie allo Scheduler, si arricchiscono di nuove funzionalità.

L'obiettivo principale dello SCP-Scheduler è l'abilitazione di comunicazione tra diverse Smart City Platform (SCP) ma permette, inoltre, la preparazione dei dati che diventano input della SCP-Dashboard.

In questa direzione, si è risposto puntualmente ai requisiti e alle funzionalità richieste.

Seguono, nei prossimi capitoli, le attività di sviluppo, suddivise nei seguenti task:

1. Task "Il componente SCP-SCHEDULER"
2. Task "Service Template"
3. Task "Moduli SERVICE"
4. Task "SCP-DASHBOARD"
5. Task "SCP-GUI"
6. Task "Formazione" .

1 Introduzione

Questo è il report delle attività svolte, nell’ambito del Contratto di Appalto avente per oggetto “Servizio per lo sviluppo di un componente software denominato SCP-SCHEDULER, per la gestione ed esecuzione di processi nella Smart City Platform” nell’ambito della Linea di Attività 1.21 “Sperimentazione del Framework per la Governance dei Dati Urbani Energetici”.

In particolare, il componente “SCP-Scheduler” realizzato e qui descritto è elemento indispensabile per la realizzazione del caso studio pilota (figura seguente), che definisce il flusso dati che parte da una SCP agente su scala urbana (SCP-Casaccia) per giungere a una SCP agente su scala nazionale (inter-SCP), descritto nel dettaglio nella Linea di Attività 20 (LA20, riferimento report Report RdS/PTR2020/013 [2]) e la cui implementazione e relativa sperimentazione sono oggetto della Linea di Attività 21 (LA21, riferimento report RdS/PTR(2021)/013 [1]).

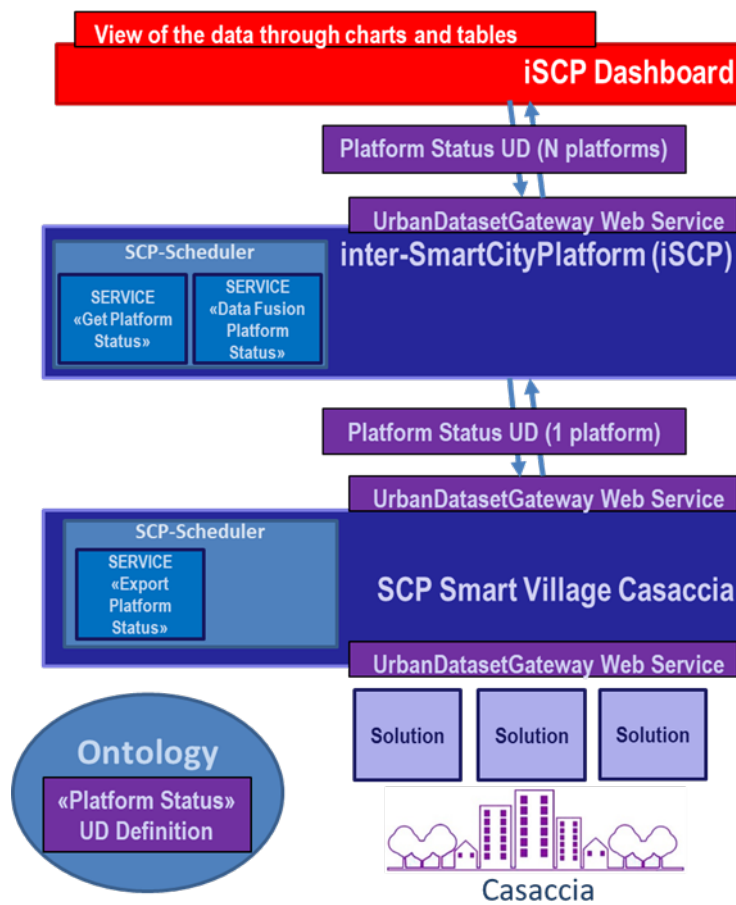


Figura 1. Caso studio "iSCP - SCPCasaccia"

Il report in oggetto ha sviluppato tutte le funzionalità richieste e descritte nel dettaglio nell’Allegato Tecnico “SCP-Scheduler” definito da ENEA per questo contratto.

Tale Allegato Tecnico è riportato integralmente nel report LA21, RdS/PTR(2021)/013 [1]; nei prossimi capitoli si farà riferimento a questi requisiti e funzionalità che, sotto la supervisione del personale ENEA, sono state sviluppate puntualmente.

2 Descrizione delle attività svolte e risultati

2.1 Task “Analisi e progettazione”

Le prime attività che sono state affrontate per la realizzazione dello SCP-Scheduler sono state quelle di analisi e progettazione del nuovo componente procedendo, in prima istanza, all’individuazione puntuale e non ambigua delle tipologie di SERVICE/JOB e delle loro caratteristiche peculiari, che il componente avrebbe dovuto supportare, permettendone la configurazione tramite l’interfaccia utente.

L’analisi di ogni tipologia di SERVICE/JOB, infatti, ha permesso di capire quali analogie e differenze vi fossero, in modo tale da progettare un’interfaccia utente per lo SCP-Scheduler che fosse efficace e che si adattasse intuitivamente alla configurazione del SERVICE in oggetto, recuperando automaticamente le informazioni quando possibile.

Tre SERVICE/JOB sono stati individuati nel caso studio “iSCP - SCPCasaccia” definito da ENEA che, per sua natura, cerca di stressare il più possibile l’utilizzo dello Scheduler, richiamandone tutte le funzionalità necessarie per implementare un flusso di dati completo.

I tre SERVICE/JOB sono:

- “Export Platform Status”,
- “Get Platform Status”,
- “DataFusion Platform Status”.

Partendo da questi requisiti sono stati generalizzati altrettanti SERVICE/JOB astratti:

- “Exporter”,
- “Get Remote UD”,
- “DataFusion UD”,

A questi SERVICE/JOB astratti ne è stato aggiunto un quarto:

- “Push Remote UD”.

Ognuno di questi SERVICE/JOB utilizza la configurazione dei permessi in produzione e accesso agli UD per poter recuperare o inviare UrbanDataset dal web service UrbanDatasetGateway della SCP locale o remota in pieno rispetto delle specifiche SCPS definite da ENEA per la rappresentazione dei dati in formato JSON e per la trasmissione degli stessi tramite interfaccia condivisa.

2.1.1 Analisi SERVICE 1: “Exporter”

La prima tipologia di SERVICE/JOB è denominata “Exporter” e permette di esportare periodicamente un set di informazioni utilizzando in output un UrbanDataset, che poi sarà salvato tramite l’UrbanDatasetGateway nel sistema della SCP locale.

Quali siano queste informazioni e dove/come vengono recuperate è un problema completamente demandato all’ [ALGORITMO INTERNO] ovvero il JOB che dovrà essere implementato ed esportato come libreria .jar da associare nella configurazione del SERVICE.

Esempio: nel caso studio pilota definito da ENEA, il JOB Exporter è stato configurato per eseguire l’esportazione delle informazioni che sono state poi inserire nel “Platform Status” in output.

SERVICE TIPO 1: EXPORTER

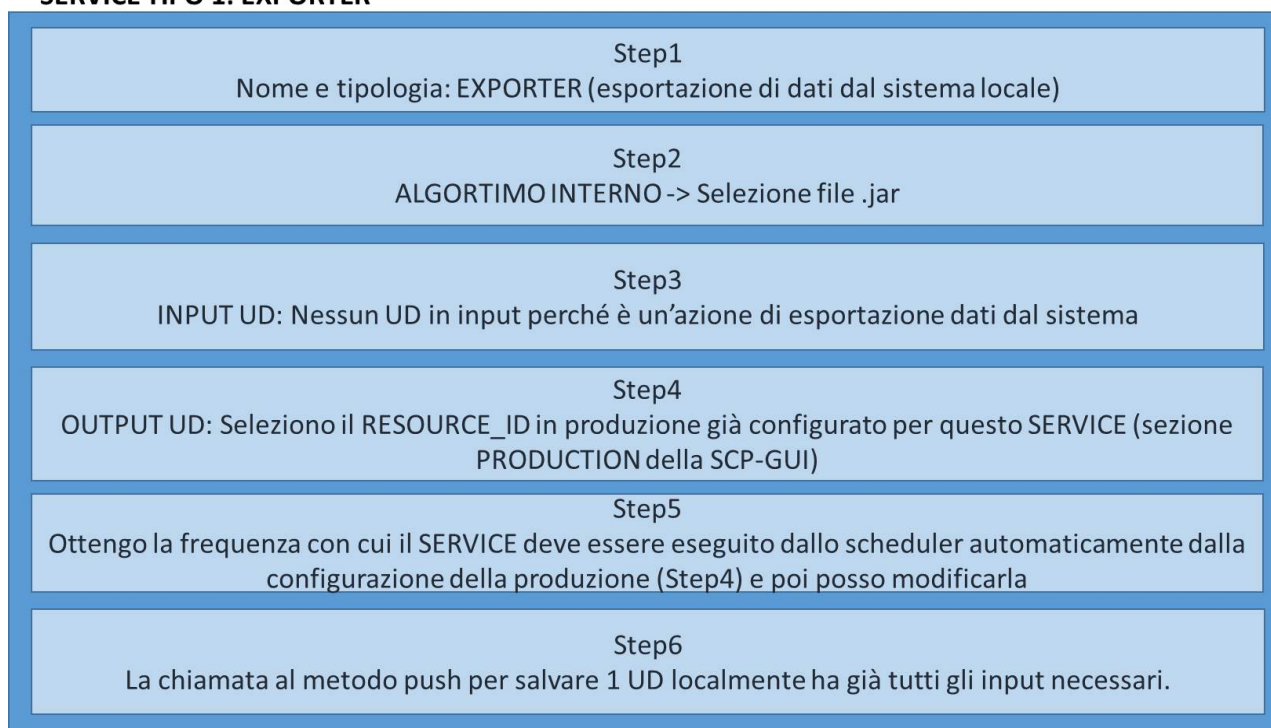


Figura 2. Analisi Service/JOB di tipo “Exporter”

Come si può leggere nell’immagine schematica del SERVICE “Exporter”, sono stati individuati gli elementi della configurazione che contraddistinguono ogni JOB astratto, organizzati in sei passi:

- Step1: Nome e Tipologia;
- Step2: Algoritmo Interno: l’implementazione dell’esportazione;
- Step3: UrbanDataset in input: non presenti in questo SERVICE;
- Step4: UrbanDataset in output: con cui rappresentare le informazioni esportate; in questa fase è necessario selezionare la produzione precedentemente definita come permesso per salvare questo UD nel sistema locale;
- Step5: configurazione della frequenza che, inizialmente, viene impostata con quella espressa nella produzione che è stata selezionata, poi può essere modificata;
- Step6: salvataggio dell’output con push verso l’UrbanDatasetGateway locale.

2.1.2 Analisi SERVICE 2: “Data Fusion”

La tipologia di SERVICE/JOB denominata “Data Fusion” permette di prendere in input, dal sistema della SCP locale, una serie di UrbanDataset, anche di tipo diverso, e applicare un algoritmo di data fusion che generi in output un UrbanDataset che poi sarà salvato tramite l’UrbanDatasetGateway nel sistema della SCP locale. Come avvenga la Data Fusion, su quali dati e con quale algoritmo, è un problema completamente demandato all’ [ALGORITMO INTERNO] ovvero il JOB che dovrà essere implementato ed esportato come libreria .jar da associare nella configurazione del SERVICE.

Esempio: nel caso studio pilota definito da ENEA, il JOB “DataFusion” è stato configurato per effettuare la DataFusion di N UD “Platform Status”, relativi ad altrettante SCP, generando un singolo UD “Platform Status” in output che sarà recuperato dalla SCP-Dashboard della inter-SCP.

SERVICE TIPO 2: DATA FUSION

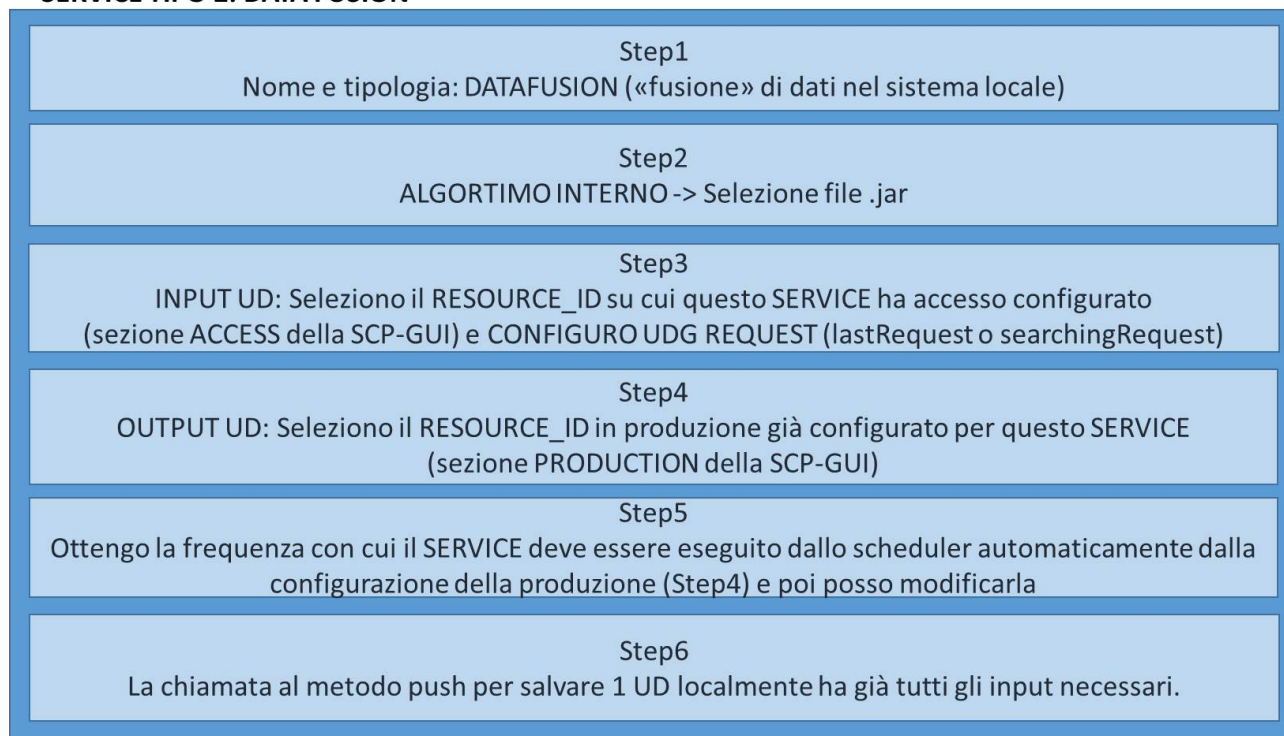


Figura 3. Analisi Service/JOB di tipo “Data Fusion”

Come si può leggere nell’immagine schematica del SERVICE “Data Fusion”, sono stati individuati gli elementi della configurazione che contraddistinguono ogni JOB astratto, organizzati in sei passi:

- Step1: Nome e Tipologia;
- Step2: Algoritmo Interno: l’implementazione della datafusion;
- Step3: UrbanDataset in input: ovvero gli UD di un certo tipo, su cui è stato configurato il permesso di accesso nella SCP locale, recuperati tramite chiamata all’UrbanDatasetGateway utilizzando i metodi, configurabili, lastRequest o searchingRequest, per ricercare rispettivamente l’ultimo UD o gli UD in una data finestra spazio-temporale;
- Step4: UrbanDataset in output: con cui rappresentare il risultato della datafusion; in questa fase è necessario selezionare la produzione precedentemente definita come permesso per salvare questo UD nel sistema locale;
- Step5: configurazione della frequenza che, inizialmente, viene impostata con quella espressa nella produzione che è stata selezionata, poi può essere modificata;
- Step6: salvataggio dell’output con push verso l’UrbanDatasetGateway locale.

2.1.3 Analisi SERVICE 3: “Get Remote UD”

La tipologia di SERVICE/JOB denominata “Get Remote UD” permette di recuperare, dal sistema di una SCP remota (o sistema remoto SCPS-compliant equivalente), uno o più UrbanDataset per poi salvarli tramite l’UrbanDatasetGateway nel sistema della SCP locale.

Esempio: nel caso studio pilota definito da ENEA, il JOB “Get Remote UD” è stato utilizzato N volte per recuperare gli N UD “Platform Status” dalle relative N SCP remote, per poi salvare tali UD nella SCP locale, ovvero la inter-SCP.

SERVICE TIPO 3: GET REMOTE

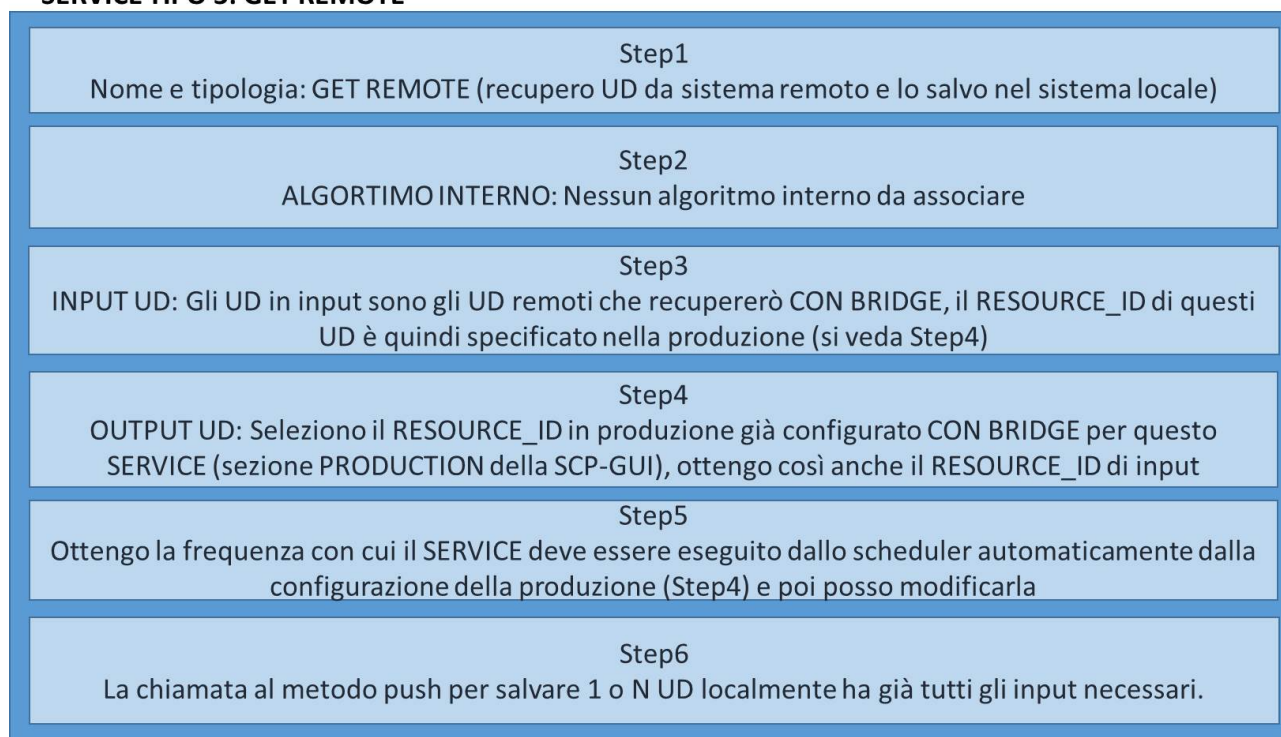


Figura 4. Analisi Service/JOB di tipo “Get Remote UD”

Come si può leggere nell’immagine schematica del SERVICE “Get Remote UD”, sono stati individuati gli elementi della configurazione che contraddistinguono ogni JOB astratto, organizzati in sei passi:

- Step1: Nome e Tipologia;
- Step2: Algoritmo Interno: nessun algoritmo interno necessario;
- Step3: UrbanDataset in input: ovvero gli UD di un certo tipo, presenti sul sistema remoto su cui è stato configurato il permesso di accesso a tali UD, recuperati tramite chiamata all’UrbanDatasetGateway remoto utilizzando i metodi, configurabili, lastRequest o searchingRequest, per ricercare rispettivamente l’ultimo UD o gli UD in una data finestra spazio-temporale;
- Step4: UrbanDataset in output: con cui salvare nella SCP locale gli UD recuperati dal sistema remoto; in questa fase è necessario selezionare la produzione precedentemente definita come permesso per salvare questi UD nel sistema locale; si noti che questa è una produzione che fa uso del protocollo di comunicazione con BRIDGE;
- Step5: configurazione della frequenza che, inizialmente, viene impostata con quella espressa nella produzione che è stata selezionata, poi può essere modificata;
- Step6: salvataggio dell’output con push verso l’UrbanDatasetGateway locale.

2.1.4 Analisi SERVICE 4: “Push Remote UD”

Inizialmente non previsto (poiché non utilizzato né descritto nel caso studio pilota) la tipologia di SERVICE/JOB denominata “Push Remote UD” permette di inviare, dal sistema della SCP locale, uno o più UrbanDataset a un UrbanDatasetGateway di una SCP remota (o sistema remoto SCPS-compliant equivalente). È stato individuato, analizzato e implementato per supportare la comunicazione della SCP in modo tale che da solo “passiva” (in attesa di UD) diventasse anche “attiva” (capace di inviare UD).

Esempio: nel caso studio pilota definito da ENEA, il JOB “Get Remote UD” potrebbe essere sostituito dal JOB “Push Remote UD”, in questo modo non sarebbe la inter-SCP a recuperare gli N UD dalle relative N SCP remote, ma sarebbero le N SCP remote a inviare gli UD “Platform Status” alla inter-SCP.

SERVICE TIPO 4: PUSH REMOTE

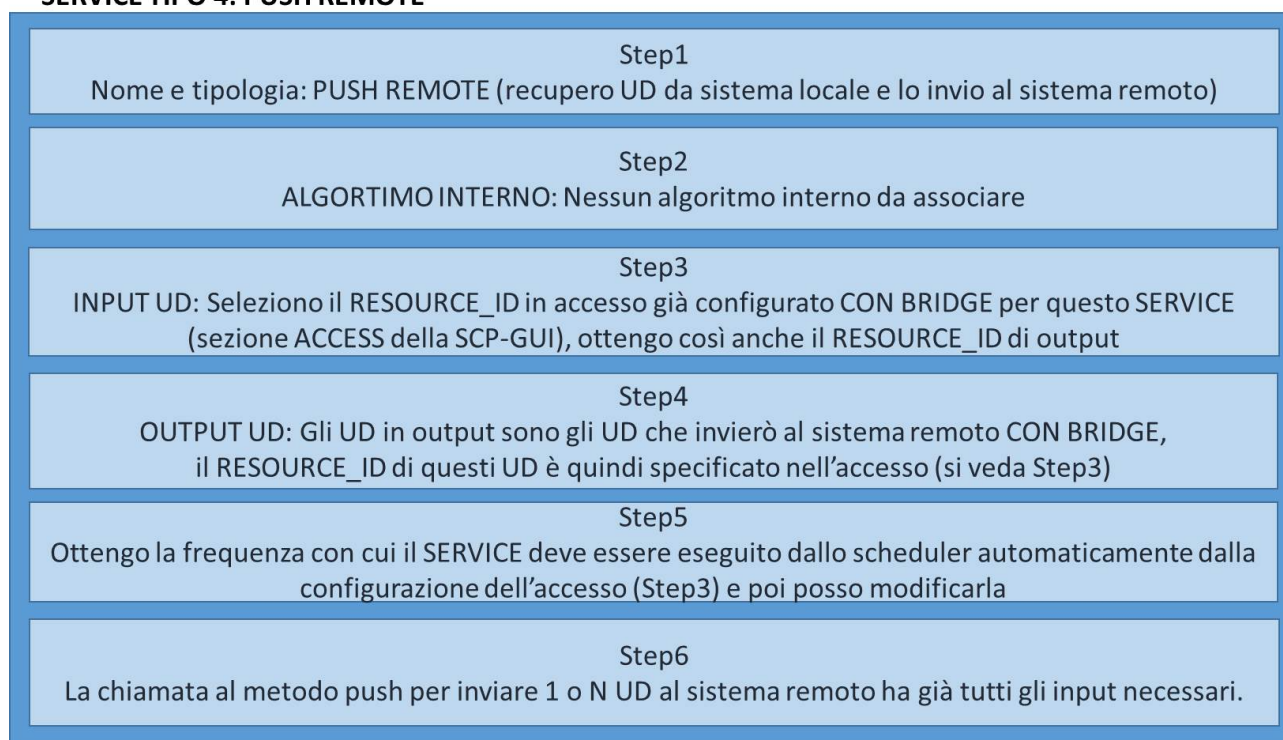


Figura 5. Analisi Service/JOB di tipo “Push Remote UD”

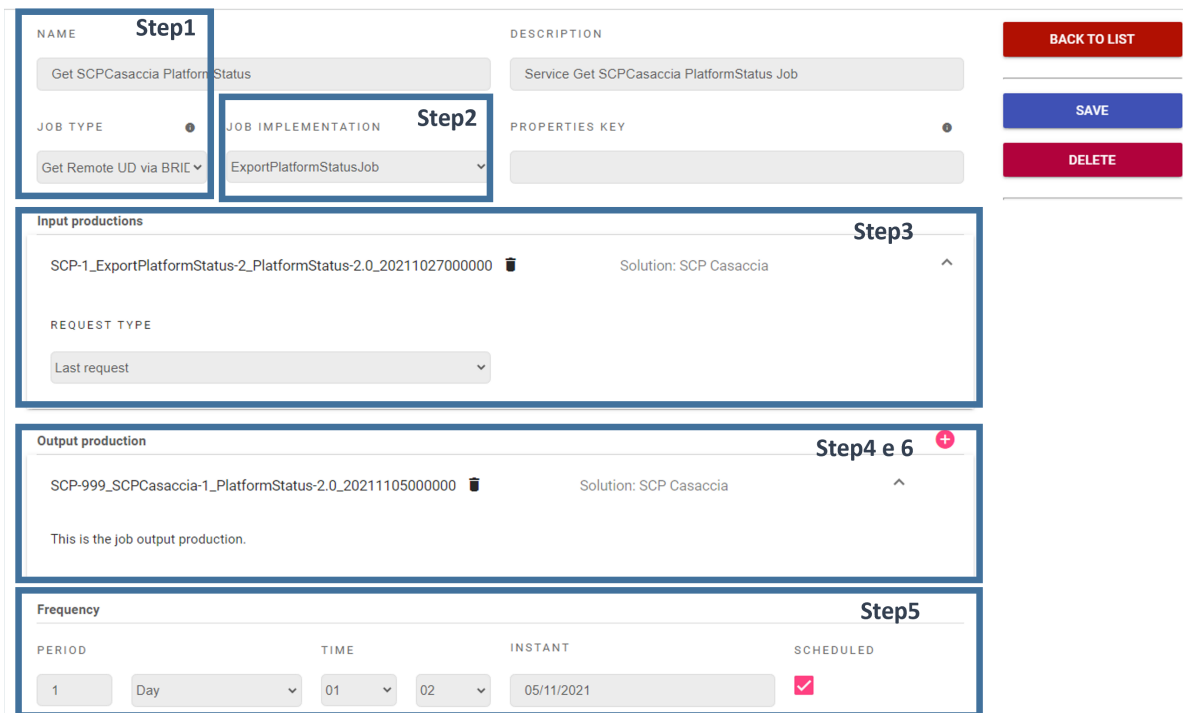
Come si può leggere nell’immagine schematica del SERVICE “Get Push UD”, sono stati individuati gli elementi della configurazione che contraddistinguono ogni JOB astratto, organizzati in sei passi:

- Step1: Nome e Tipologia;
- Step2: Algoritmo Interno: nessun algoritmo interno necessario;
- Step3: UrbanDataset in input: ovvero gli UD di un certo tipo, presenti nel sistema locale su cui è stato configurato il permesso di accesso a tali UD, recuperati tramite chiamata all’UrbanDatasetGateway locale utilizzando i metodi, configurabili, lastRequest o searchingRequest, per ricercare rispettivamente l’ultimo UD o gli UD in una data finestra spazio-temporale; si noti che questo è un accesso che fa uso del protocollo di comunicazione con BRIDGE;
- Step4: UrbanDataset in output: con cui inviare nella SCP remota gli UD recuperati dal sistema locale; in questa fase è necessario configurare sul sistema remoto la produzione come permesso per salvare questi UD;
- Step5: configurazione della frequenza;

- Step6: salvataggio dell'output con push verso l'UrbanDatasetGateway remoto.

2.1.5 Dall'analisi al risultato

Viene mostrato in questo paragrafo uno screenshot dell'interfaccia utente che permette la configurazione dei JOB/SERVICE nello SCP-Scheduler, evidenziando gli step comuni che sono stati schematicamente individuati e descritti nei precedenti paragrafi e qui trovano concreto risultato.



The screenshot displays a configuration form for a job in the SCP Scheduler. The form is organized into several sections, each labeled with a step number:

- Step 1:** NAME (Get SCPCasaccia Platform Status) and DESCRIPTION (Service Get SCPCasaccia PlatformStatus Job).
- Step 2:** JOB IMPLEMENTATION (ExportPlatformStatusJob).
- Step 3:** Input productions (SCP-1_ExportPlatformStatus-2_PlatformStatus-2.0_20211027000000). Solution: SCP Casaccia.
- Step 4 e 6:** Output production (SCP-999_SCPCasaccia-1_PlatformStatus-2.0_20211105000000). Solution: SCP Casaccia. This is the job output production.
- Step 5:** Frequency configuration (Period: 1, Day, Time: 01, 02, Instant: 05/11/2021, SCHEDULED: checked).

On the right side of the form, there are three buttons: BACK TO LIST (red), SAVE (blue), and DELETE (purple).

Figura 6. Gli step di creazione del JOB

Si noti come ogni step coincida con una scelta utente a cui, successivamente, consegue la presentazione grafica degli step successivi, coerentemente con la scelta fatta. Nel paragrafo 1.2 sono indicati i vari passi che permettono la definizione dei JOB/SERVICE.

2.2 Task “Il componente SCP-SCHEDULER”

Questo task ha comportato la realizzazione del componente SCP-SCHEDULER, integrato nella SCP-GUI della piattaforma SCP di ENEA; lo scopo dello SCP-Scheduler consiste nell’orchestrare l’esecuzione di SERVICE (chiamati anche JOB) configurati nella SCP.

Un SERVICE/JOB ha la caratteristica di essere schedulato nel sistema, tramite trigger che scattano periodicamente, secondo la frequenza indicata. Il JOB, inoltre, possiede le informazioni relative all’accesso ad *UrbanDataset* (UD) prodotti da una SCP, locale o remota, così come quelle relative alla produzione di uno o più UD, localmente o remotamente.

Si noti che lo SCP-Scheduler ha implementato le specifiche SCPS¹ di ENEA per la rappresentazione degli *UrbanDataset* e la comunicazione in lettura e scrittura con il web service *UrbanDatasetGateway*.

In particolare, lo SCP-SCHEDULER fornisce le seguenti funzionalità:

- Servizio di discovery dei JOB disponibili
- Configurazione dei JOB
- Azioni CRUD per i JOB
- Schedulazione, avvio e stop dei JOB configurati

I sotto-componenti dello SCP-Scheduler sono mostrati schematicamente nella seguente figura.

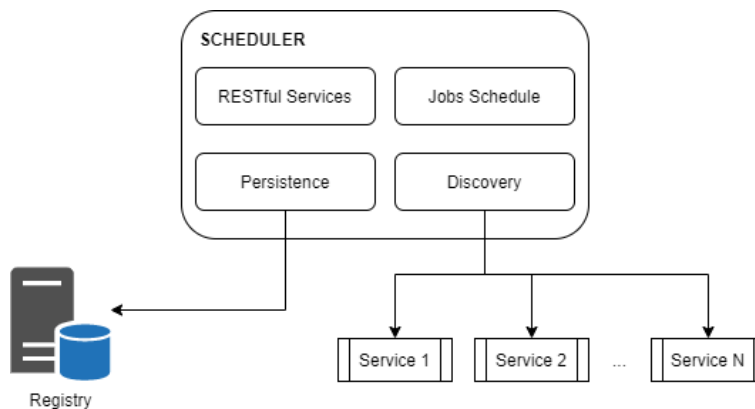


Figura 7. Sotto-componenti dello SCP-Scheduler

Discovery è il componente che, sfruttando il meccanismo offerto dal class loader dell’application server, identifica i JOB presenti nella webapp e permette di interagire con essi.

Persistence si occupa della persistenza delle configurazioni dei JOB. In particolare, si propone l’introduzione di una nuova tabella JOB all’interno del database Registry, così fatta:

- Id: identificativo del job, PK
- Name, description: nome e descrizione testuale del job.
- Type: il tipo permette di automatizzare alcuni comportamenti della lettura e scrittura degli UD, come in seguito descritto.

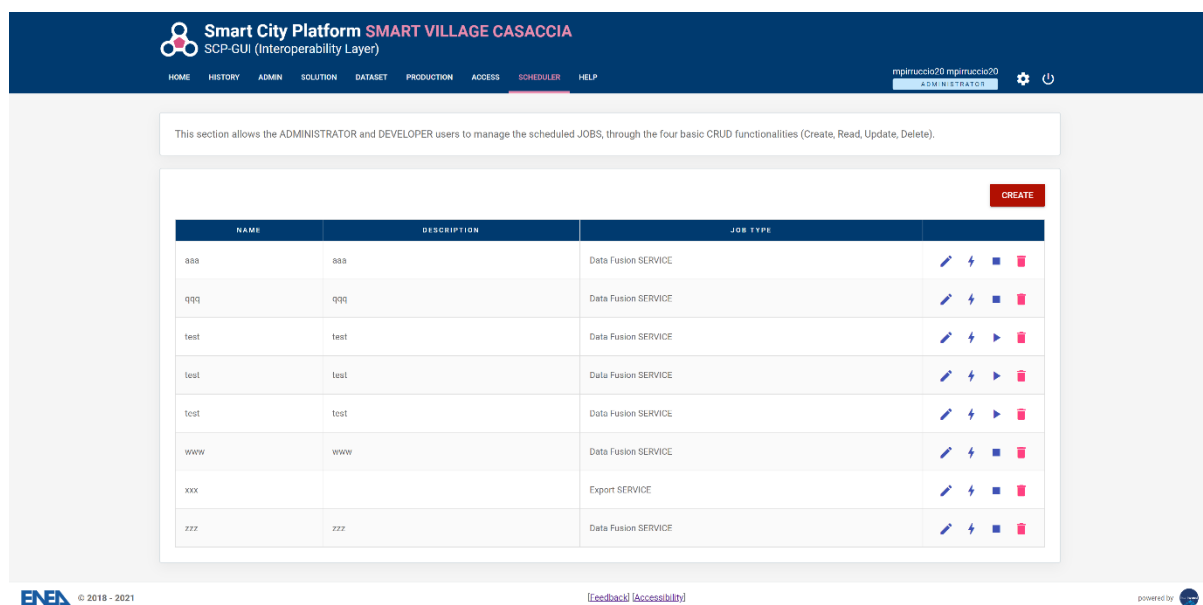
¹ Smart City Platform Specification for interoperability layer (SCPS), <https://smartcityplatform.enea.it/#/it/specification/>

- ImplementationClass: implementazione java del SERVICE associato al JOB.
- Frequency: informazioni di frequenza di esecuzione del job, composta da period, time e instant.
- Input resources: informazioni di accesso all'UDG secondo i pattern push e request/response.
- Output resource: informazioni di generazione degli UD del SERVICE.

Il Job Schedule è il componente che gestisce il ciclo di vita dei JOB. Utilizza la libreria open source Quartz. I servizi RESTful permettono l'accesso alle funzionalità precedentemente descritte. Sono utilizzati dalle interfacce di amministrazione.

La componente client è composta da una sezione profilata, accessibile dal menu principale di navigazione di SCP-GUI.

La prima pagina mostra l'elenco dei JOB configurati e, per ognuno di essi, fornisce le azioni di avvio, stop, modifica e cancellazione.



The screenshot shows the 'Smart City Platform SMART VILLAGE CASACCIA' interface. The top navigation bar includes 'HOME', 'HISTORY', 'ADMIN', 'SOLUTION', 'DATASET', 'PRODUCTION', 'ACCESS', 'SCHEDULER', and 'HELP'. The 'SCHEDULER' tab is active. Below the navigation bar, there is a text box stating: 'This section allows the ADMINISTRATOR and DEVELOPER users to manage the scheduled JOBS, through the four basic CRUD functionalities (Create, Read, Update, Delete)'. A 'CREATE' button is visible in the top right corner of the table area.

NAME	DESCRIPTION	JOB TYPE	
aaa	aaa	Data Fusion SERVICE	edit start stop delete
qqq	qqq	Data Fusion SERVICE	edit start stop delete
test	test	Data Fusion SERVICE	edit start stop delete
test	test	Data Fusion SERVICE	edit start stop delete
test	test	Data Fusion SERVICE	edit start stop delete
www	www	Data Fusion SERVICE	edit start stop delete
xxx		Export SERVICE	edit start stop delete
zzz	zzz	Data Fusion SERVICE	edit start stop delete

At the bottom of the page, there is a footer with 'ENEA © 2018 - 2021', a '[Feedback] (accessibility)' link, and a 'powered by' logo.

Figura 8. Lista Jobs nella GUI dello SCP-Scheduler

La pagina di dettaglio JOB permette la loro creazione e modifica.

In particolare:

- L'elenco dei JOB disponibili è fornito dal componente di discovery.
- Il componente della persistenza permette di accedere alle production utilizzate dal SERVICE.
- La frequenza può essere impostata automaticamente, dipendentemente dalla production di output selezionata, o definita manualmente.

Il flusso di cancellazione del JOB, similmente a quanto già presente nella SCP GUI, mostra prima un popup in cui viene chiesta una conferma alla cancellazione. In caso affermativo:

- Viene eliminato il record corrispondente dalla tabella JOB.
- Il JOB configurato nel componente Job Schedule viene rimosso.

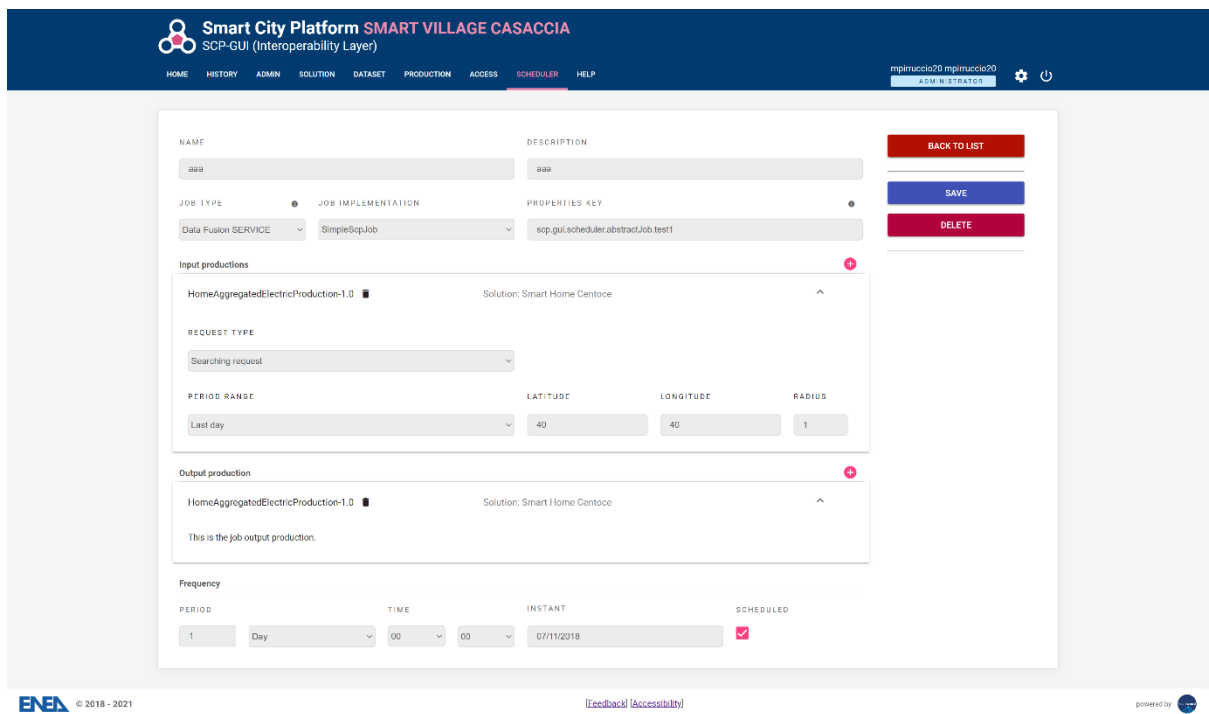


Figura 9. Creazione ed edizione Job nello SCP-Scheduler

In funzione del tipo di job selezionato in fase di creazione, il sistema permette di ricercare le produzioni di input e di output compatibili con la tipologia indicata.

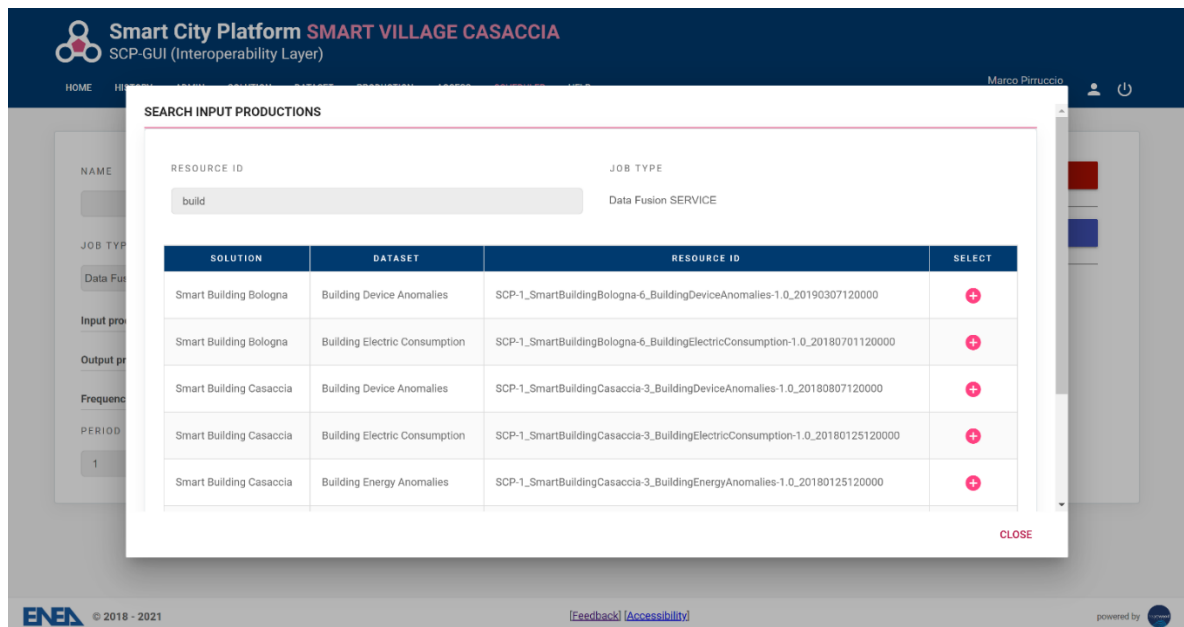


Figura 10. Selezione delle produzioni di input

Dopo aver selezionato le produzioni di input, per ognuna di esse è possibile specificare il tipo di request da utilizzare per la lettura degli UD, come indicato nelle SCPS Communication 2.0²:

- Last request
- Searching request: in questo caso le interfacce mostrano degli ulteriori campi in cui specificare i parametri di ricerca.



Input productions +

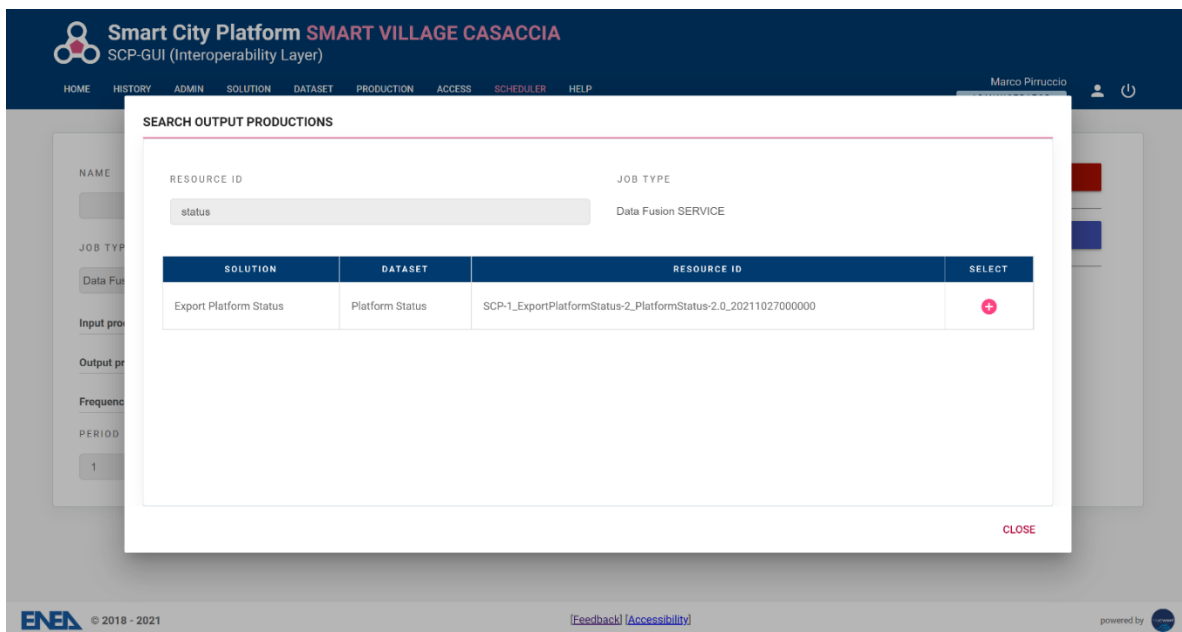
SCP-1_DecisionSupportSystem-11_WeatherCondition-1.0_20181001120000 Solution: Decision Support System

REQUEST TYPE

Searching request

PERIOD RANGE LATITUDE LONGITUDE RADIUS

Figura 11. I parametri della searching request



Smart City Platform SMART VILLAGE CASACCIA
SCP-GUI (Interoperability Layer)

HOME HISTORY ADMIN SOLUTION DATASET PRODUCTION ACCESS SCHEDULER HELP

Marco Pirruccio

SEARCH OUTPUT PRODUCTIONS

RESOURCE ID JOB TYPE

status Data Fusion SERVICE

SOLUTION	DATASET	RESOURCE ID	SELECT
Export Platform Status	Platform Status	SCP-1_ExportPlatformStatus-2_PlatformStatus-2.0_20211027000000	+

CLOSE

ENEA © 2018 - 2021 [Feedback] [Accessibility] powered by

Figura 12. Selezione delle produzioni di output

La selezione dell'UrbanDatasetGateway da cui leggere gli UD e a cui inviare gli UD prodotti dipende dalla ciò che è stato indicato come produzioni di input e di output. Nel caso in cui si sia selezionata una produzione associata a uno specifico endpoint, verranno usate queste informazioni; in caso contrario verrà usato l'UrbanDatasetGateway di default della SCP-GUI.

Dal punto di vista implementativo, le funzionalità dello scheduler è stata inclusa all'interno della SCP-GUI. Anche lo scheduler, pertanto, in modo omogeneo alla SCP-GUI, è composto da due componenti che forniscono funzionalità diverse: la componente server, che si occupa di gestire il ciclo di vita dei JOB; la

² <https://smartcityplatform.enea.it/#/it/specification/communication/2.0/index.html>

componente client, che include le interfacce utente utilizzate dagli amministratori della piattaforma per la configurazione.

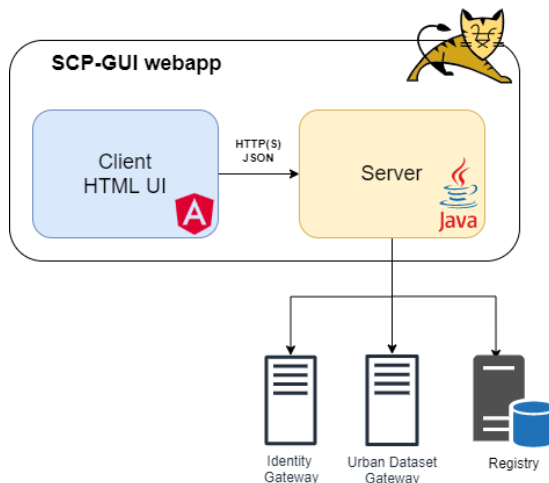


Figura 13. Architettura di deployment

Il processo di build e di deploy della SCP-GUI è stato modificato in modo da includere anche questo componente nei flussi già in essere.

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

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

2.3 Task “Service Template”

Il SERVICE Template è il progetto da utilizzare per sviluppare nuovi SERVICE/JOB che possono essere gestiti dal componente SCP-SCHEDULER.

Un JOB implementa l'interfaccia *it.enea.scp.scheduler.job.ScJob* o estende la classe *it.enea.scp.scheduler.job.SimpleScJob*.

I metodi implementabili dal JOB sono chiamati dall'SCP-SCHEDULER durante la sua esecuzione temporizzata:

- *void init(ScJobConfiguration configuration)*: chiamato all'avvio, fornisce al JOB i dati di inizializzazione: l'elenco degli UD di input e i parametri di configurazione
- *Optional<List<UD>> execute()*: contiene la logica specifica del JOB. Tipicamente utilizzerà gli UD di input per produrre uno o più UD di output. Questi UD, se presenti, verranno inviati dall'SCP-SCHEDULER all'UD Gateway associato alla produzione di output del JOB.
- *void shutdown()*: chiamato al termine del ciclo di esecuzione del JOB.

I parametri di configurazione del JOB vanno indicati nel file *scp-scheduler.properties*, presente nel server della SCP-GUI, insieme al file *scp-gui.properties*.


Ogni JOB può esprimere i parametri a cui è interessato specificando, nella scheda di editazione, tramite l'attributo *propertiesKey*, il prefisso di tali parametri.

Ad esempio, specificando *scp.myJob*, potrà leggere tutte le property la cui chiave è:

scp.myJob.myProperty1=myValue1

scp.myJob.myProperty2=myValue2

...

Viene qui mostrato il diagramma dei componenti che compongono il Service Template e, di conseguenza, anche un SERVICE. Le funzionalità peculiari del Service Template sono evidenziate tramite l'icona .

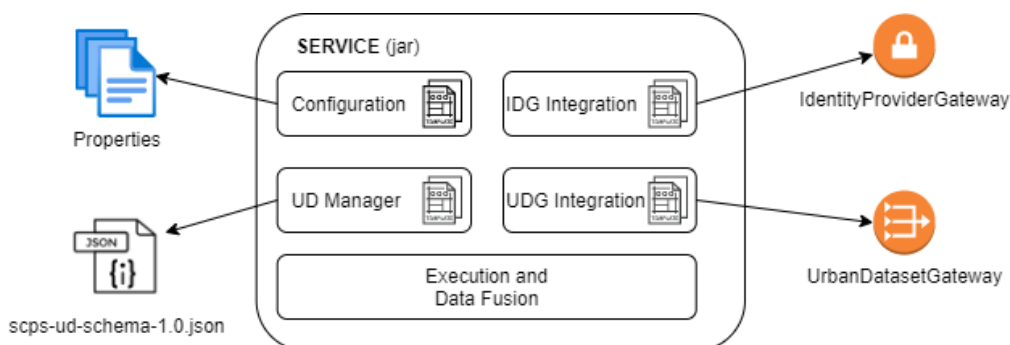


Figura 14. Architettura software del service template

Il componente Configuration fornisce il servizio di configurazione del SERVICE, leggendo il file properties precedentemente descritto.

Il componente UD Manager ha lo scopo di validare gli UD tramite lo schema JSON delle specifiche "SCPS Information". Comprende inoltre le funzionalità di salvataggio, lettura e cancellazione degli UD presenti localmente, sul file system del server.

Il componente UDG Integration si occupa di interfacciarsi all'UDGateway e consuma le sue API. In particolare integra i servizi:

- LastRequest: lettura UD
- SearchingRequest: lettura UD
- Push: pubblicazione UD

Questo componente abilita la comunicazione secondo i pattern push e request/response e implementa pertanto anche le funzionalità indicate come SCP-BRIDGE nell'allegato tecnico.

È inoltre stato realizzato il componente IDP Integration, che si occupa di interfacciarsi all'IdentityProvider al fine di effettuare il login ed ottenere un token con cui firmare le chiamate all'UDGateway.

Questo progetto è pubblicato nel repository Maven di GitLab. È pertanto utilizzabile includendo la dipendenza nel *pom.xml* del JOB implementato.

Ad esempio:

```
<repositories>
  <repository>
    <id>gitlab-maven</id>
    <url>https://gitlab.com/api/v4/packages/maven</url>
  </repository>
</repositories>
```

```
<dependencies>
  <dependency>
    <groupId>it.enea.scp.scheduler</groupId>
    <artifactId>job-template</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
</dependencies>
```

Nel caso in cui si stia configurando l'accesso al repository Maven per la prima volta, va effettuata una ulteriore modifica al file di configurazione di Maven. È necessario che nel file *settings.xml* di Maven sia indicato un personal access token generato per il proprio profilo da https://gitlab.com/-/profile/personal_access_tokens.

Va selezionato lo scope api in fase di creazione.

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
  https://maven.apache.org/xsd/settings-1.0.0.xsd">
<servers>
  <server>
    <id>gitlab-maven</id>
    <configuration>
      <httpHeaders>
        <property>
          <name>Private-Token</name>
          <value>*****</value>
        </property>
      </httpHeaders>
    </configuration>
  </server>
</servers>
</settings>
```

I sorgenti del SERVICE Template sono in:

- <https://gitlab.com/marcopirruccio/scp-scheduler-job-template>.

2.4 Task "Moduli SERVICE"

Questo task è strettamente relativo al Caso Studio "Da inter-SCP a SCP-Casaccia" descritto in Appendice B dell'Allegato Tecnico definito da ENEA (riportato nel report della LA1.20, riferimento RdS/PTR2020/013 [2]). In tale Caso Studio sono stati individuati 3 moduli SERVICE che, secondo la richiesta, "dovranno essere implementati, installati, configurati ed eseguiti dallo SCP-Scheduler".

E' necessario prima fare una precisazione riguardo la terminologia, consolidata e raffinata, di concerto con il personale ENEA:

- un JOB è un modulo software che implementa un qualsiasi algoritmo;
- un SERVICE è un caso specifico di JOB che legge e/o scrive UrbanDataset localmente ad una SCP;
- un BRIDGE è un particolare caso di SERVICE che leggere e/o scrive UrbanDataset che si trovano su una SCP/Solution remota.

JOB, SERVICE e BRIDGE sono moduli software configurabili ed eseguibili dallo SCP-Scheduler con una frequenza temporale predefinita.

I tre moduli software sviluppati sono i seguenti:

- SERVICE "Export Platform Status": effettua un'azione di esportazione, secondo specifiche date, dell'UrbanDataset "Platform Status", presso la SCP-Casaccia e ne effettua la push nell'UrbanDatasetGateway locale secondo specifiche SCPS Communication;
- BRIDGE "Get Platform Status": recupera l'UD "Platform Status", salvato (nel punto 1) nella SCP-Casaccia, e ne effettua la push nell'UrbanDatasetGateway locale presso la inter-SCP secondo specifiche SCPS Communication;
- SERVICE "Data Fusion Platform Status": presso la inter-SCP, effettua l'azione di Data Fusion di più "Platform Status" (singoli) recuperati da più SCP (al momento solo da SCP-Casaccia) in un singolo "Platform Status" (multiplo) che sarà poi recuperato e utilizzato dall iSCP-Dashboard.

Per effettuare la configurazione è stato necessario installare lo SCP-Scheduler nelle due SCP coinvolte (SCP-Casaccia e inter-SCP) nonché configurare le collaborazioni secondo le indicazioni del personale ENEA.

I tre moduli software sono quindi stati implementati, configurati e testati negli SCP-Scheduler relativi e sono oggetto della fase sperimentale del caso studio pilota (si veda report LA1.21, riferimento Report RdS/PTR(2021)/013 [1]).

Si rimane a disposizione per fornire tutto il supporto necessario per giungere alla conclusione della sperimentazione e ottenere il risultato pianificato.

2.5 Task “SCP-DASHBOARD”

Il task in oggetto ha comportato l’inserimento di un nuovo tipo di grafico che fa uso della tecnologia Google Maps nel componente SCP-DASH (SCP-Dashboard), componente che permette di configurare Dashboard da associare alle SCP per la visualizzazione dei dati tramite tabelle e grafici (charts).

In particolare, per questo nuovo grafico, sono state utilizzate le API di Google Maps, esposte tramite la libreria *@angular/google-maps*.

È quindi utilizzabile il nuovo tipo di chart “gmap”, che viene visualizzato come mostrato nella seguente figura.



Figura 15. Nuovo grafico con tecnologia Google Maps

Il formato dell’UrbanDataset (UD) utilizzato per alimentare questo tipo di grafico è analogo a quello utilizzato per il grafico descritto nel caso studio “Map Info” (si veda report LA1.20, riferimento RdS/PTR2020/013 [2]).

La configurazione del report fa quindi uso di un chart di tipo “gmap”, il cui mapping è, ad esempio:

```

"mapping": {
  "line": [
    {
      "lineId": 1,
      "labels": [ "Latitude", "Longitude", "N. Solutions", "SCP" ],
      "values": [
        "exp.gMapValues(ud.UrbanDataset.values.line[0].coordinates,
          ud.UrbanDataset.values.line[0].property,      {'ManagedSolutions',
            'SCPName'})" ],
      "description": "",
      "unit": "adimensional"
    },
    {...}
  ]
}

```

Si noti come anche per questo tipo di grafico è stata utilizzata la libreria OGNL per poter esprimere delle espressioni con cui accedere ai dati dell’UD. In particolare l’espressione *exp.gMapValues* usa i parametri:

- *ud.UrbanDataset.values.line[0].coordinates*: permettere di leggere le coordinate della linea 0 dell’UD letto

- `ud.UrbanDataset.values.line[0].property`: permette di leggere le property della linea 0
- `{'ManagedSolutions', 'SCPName'}`: specifica quali property restituire.

I sorgenti del nuovo tipo di grafico sono inclusi nel progetto SCP-DASHBOARD:

- <https://gitlab.com/marcopirruccio/scp-dash>
- <https://gitlab.com/marcopirruccio/scp-dash-server>.

2.6 Task “SCP-GUI”

In questo task è stato sviluppato un nuovo grafico mostrato nella sezione HOME della SCP-GUI, per i soli utenti ADMINISTRATOR, READER, DEVELOPER.

Questo grafico è analogo a quello già presente per le SOLUTION e mostra gli indicatori di interoperabilità relativi ai SERVICE. In particolare, tramite una vista ad istogramma, indica quanto ogni service è stato aderente alla configurazione della produzione UrbanDataset definita.

Grazie a questo grafico è possibile monitorare in tempo reale la percentuale di UrbanDataset prodotti da ogni SERVICE rispetto alla frequenza configurata (se, per esempio, un SERVICE deve esportare un UD al giorno e si sta osservando l’ultima settimana, se sono stati prodotti 7 UD viene mostrata una percentuale pari al 100%).

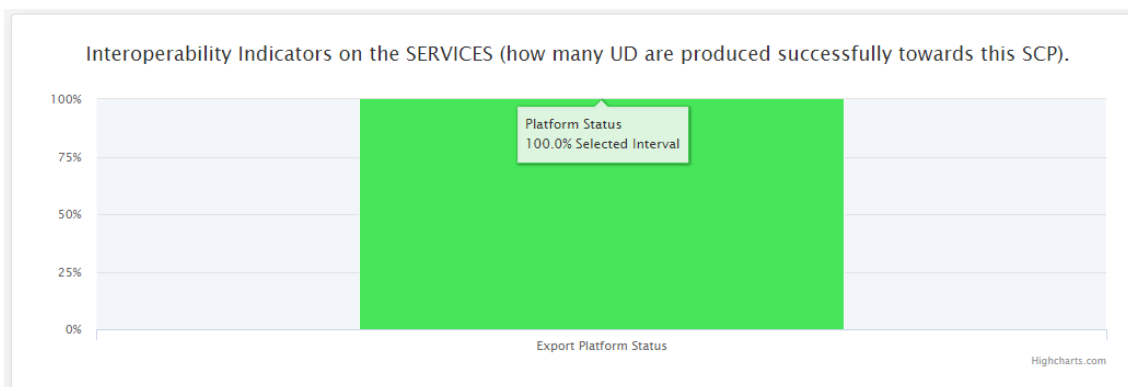


Figura 16. Grafico indicatori di interoperabilità per SERVICE

Si noti come questa nuova funzionalità nella SCP-GUI permetta di monitorare i SERVICE che sono stati configurati e vengono eseguiti dallo SCP-Scheduler.

I dati mostrati in questo nuovo grafico possono essere filtrati:

- solo OPENDATA
- range di date.

I sorgenti del componente che implementa questo grafico sono inclusi nei progetti della SCP-GUI e sono accessibili su GitLab:

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

2.7 Task “Formazione”

Heartwood Labs ha già erogato diverse ore di formazione remota relativamente all’installazione e configurazione dei componenti SCP-SCHEDULER, SERVICE Template e SCP-DASHBOARD.

Tali attività sono state recepite dai referenti ENEA e opportunamente conteggiate.

La società resta a disposizione per ulteriore supporto per la modifica e configurazione delle applicazioni web e dei componenti qui indicati tramite ulteriori sessioni di formazione, attualmente differite da ENEA a causa emergenza Covid19.

2.8 Tecnologie e accessibilità

Dal punto di vista tecnologico, i componenti qui descritti sono realizzati 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.

Le interfacce rispettano i requisiti di accessibilità livello AA. La verifica di tali requisiti è stata effettuata con i validatori:

- Vamolà³
- WAVE Web Accessibility Evaluation Tool⁴.

³ Validatore per l’accessibilità “Vamolà”, http://www.validatore.it/vamola_validator/checker/index.php

⁴ Validatore per l’accessibilità “WAVE”, <https://wave.webaim.org/>

3 Conclusioni

In conclusione, sono stati sviluppati correttamente i seguenti componenti software:

1. Il componente “SCP-SCHEDULER” integrato nella “SCP-GUI”;
2. Il componente “Service Template” che permette di creare nuovi SERVICE;
3. I tre "Moduli SERVICE" richiesti:
 - SERVICE "Export Platform Status";
 - BRIDGE "Get Platform Status";
 - SERVICE “Data Fusion Platform Status”;
4. La modifica al componente “SCP-DASHBOARD” per supportare il servizio Google Maps;
5. La modifica del componente SCP-GUI per ottenere un grafico di indicatori per l’interoperabilità per i SERVICE, simile a quello già presente per le Solution.

Le attività di sviluppo non hanno comportato criticità e sono state svolte nei tempi anche grazie alle funzionalità richieste dettagliate e descritte dal personale ENEA con cui si è lavorato di concerto.

Tutte le attività di sviluppo suddette sono già state recepite dal personale ENEA che utilizzerà i relativi risultati per proseguire l’obiettivo primario dell’attività LA21, relativa l’implementazione del caso studio pilota e la relativa sperimentazione (in particolare relativamente la comunicazione tra inter-SCP agente su scala nazionale e almeno una SCP agente su scala urbana).

Heartwood Labs rimane a disposizione per fornire completo supporto al personale ENEA per la customizzazione e utilizzo dei componenti sviluppati al fine di completare il caso studio pilota.

4 Riferimenti bibliografici

[1] C. Novelli, A. Frascella, A. Brutti, N. Gessa, S. Pizzuti, F. Moretti, G. Santomauro, M. Chinnici, G. Ponti, "Sperimentazione piattaforma Smart City su scala nazionale"
Riferimento Rapporto Tecnico "RdS/PTR(2021)/013"

[2] C. Novelli, G. Santomauro, A. Frascella, A. Brutti, C. Petrovich, F. Moretti, M. Chinnici, G. Ponti, S. Pizzuti, "Sviluppo Piattaforma per la Governance dei Dati Urbani Energetici",
Riferimento Rapporto Tecnico "RdS/PTR2020/013"

5 Abbreviazioni ed acronimi

Tutte le abbreviazioni ed acronimi utilizzati sono stati già precedentemente individuati e formalizzati da ENEA nelle specifiche Smart City Platform Specification for interoperability layer (SCPS) e, in particolare, tali definizioni si trovano nel Glossario che si trova sul portale dello SCP Project a questo URL:

<https://smartcityplatform.enea.it/#/it/specification/glossary.html>

6 Curriculum scientifico

curriculum scientifico di Marco Pirruccio.

6.1 *Articoli, pubblicazioni, docenze e presentazioni*

- Docenza su tecnologie multiplayer per lo sviluppo di videogiochi per il corso PA 2020-15444/RER "Game Producer".
- Docenza per iMasterArt su Java EE, servizi REST, unit test, build tramite Maven.
- Docenze su C# e Unity per IPID, Italian Party of Indie Developers.
- Docenza per il corso IFTS 2017-7563/RER "Tecnico per la progettazione e la prototipazione di dispositivi "Internet delle cose" per il monitoraggio dei dati ambientali con tecnologie Arduino e Raspberry Pi".
- Workshop sulle tecnologie dei dispositivi wearable, tenuto presso il Politecnico di Milano, Scuola del Design, corso di Laurea in Design della Moda, A.A. 2016-2017.
- Docenza per il corso IFTS 2016-5696/RER "Progettista e programmatore di dispositivi e applicazioni per l'IoT".
- Docenza per il corso IFTS 2015-4256/RER "Tecnico per lo sviluppo di applicazioni informatiche per l'Internet delle cose con linguaggio Java".
- Docenza per il corso 2015-4110/RER/1 "Analista programmatore per lo sviluppo di applicazioni tramite linguaggio Java".
- Partecipazione alla "Rimini Beach Mini Maker Faire" del 2015. Presentazione del prototipo di una stazione meteo sviluppata su Raspberry Pi in tecnologia Java. Legge i seguenti sensori: ML8511 (intensità UV), BMP180 (temperatura, pressione, altitudine), TGS2600 (contaminanti dell'aria), DHT22 (temperatura, umidità). I dati raccolti sono spediti su un cloud utilizzando il protocollo MQTT. È stato inoltre presentato un client web che tramite WebSocket legge in tempo reale questi dati e li mostra sotto forma di grafici interattivi.
- Partecipazione all'"Enterprise European Business Game" del 2015 al fine di supportare lo sviluppo di un'idea di business presso l'Istituto Manfredi Tanari.
- Ciclo di lezioni presso l'Istituto Manfredi Tanari durante l'anno scolastico 2014-2015 per il corso: "Sviluppatore di applicazioni web per dispositivi mobili".
- Docenza per il corso IFTS 2014-2950/RER: "Progettista e sviluppatore di applicazioni cross-platform tramite tecnologia web per la pubblicazione di contenuti per smartphone e tablet".
- Presentazione per il "Festival della Cultura Tecnica" del 2014 durante l'evento "INNETworking". Partecipazione al tavolo di lavoro "Creatività e ICT" relativamente alle nuove tecnologie web, mobile e IoT.
- Docenza per il corso IFTS 2013-2289/RER: "Sviluppatore di applicazioni per mobile con tecnologie web responsive per smartphone e tablet".
- Docenza per il corso "Java Enterprise Edition" presso E-One Group.
- Docenza per il corso 2012-1476/RER/1: "Analista programmatore specializzato nello sviluppo di applicazioni web ottimizzate per dispositivi mobili".
- Docenza per il corso 2011-1259/RER/3: "Analista programmatore orientato alle tecnologie web Enterprise 2.0".
- Incontri sulle tecnologie web per l'IIS Mattei di San Lazzaro (Bologna) per il "Progetto Alternanza Scuola/Lavoro" nel 2010.
- Formazione su Hibernate e Struts per Hars di Modena nel 2008.
- Formazione su Hibernate per D'vel di Bologna nel 2007.
- Ciclo di lezioni su tecnologie web, Java EE e mobile presso Gecod e Kettydo+ dal 2006 ad oggi.
- Ciclo di lezioni su Struts e Web Services per il Centro ENEA di Bologna nel 2005.
- Scrittura nel 2005 dell'articolo "Installazione di Java su Debian GNU/Linux" per il JUG di Bologna.
- Formazione per Quadrante su design pattern, JEE e Struts nel 2004.

- Partecipazione al “Semantic Web Best Practices and Deployment Working Group” del W3C con l’Università di Bologna, Ontopia e Mondeca nel 2004.
- Pubblicazione nel 2003 dell’articolo “Metadata on the Web: on the integration of RDF and Topic Maps” su “Proceedings of the Extreme Markup Languages 2003 Conference Montreal”.
- Presentazione dell’articolo “Metadata on the Web: on the integration of RDF and Topic Maps” alla conferenza “WWW2003” (a Budapest, Ungheria) e alla conferenza “Extreme Markup Languages 2003” (a Montreal, Canada).
- Ciclo di lezioni su RDF e Topic Maps durante il corso “Laboratorio di Tecnologie Web” del corso di laurea in Informatica dell’Università di Bologna nel 2002.

6.2 Conoscenze informatiche

- Tecnologie e linguaggi conosciuti: Java (JSE e JEE), C, C++, C#, Perl, Python, SQL, UML, XML, XSL, DTD, XML Schema, JSON, REST, Swagger (Open API), RDF, TCP/IP, UDP, HTTP, SOAP, WebSocket, MQTT, LoRaWAN, Bluetooth.
- Tecnologie Java adottate: EJB, JPA, JDBC, JTA, JNDI, JMS, JMX, JSP, JSTL, JSF, JNLP, JAI, JAX-WS, JAX-RS, JAXB.
- Tecnologie di frontend web: JavaScript, TypeScript, jQuery, Angular, Bootstrap, HTML, CSS.
- Tecnologie mobile: Flutter, React Native.
- Game engine: Unity, Unreal.
- IoT: Raspberry Pi, Arduino AVR e SAMD, ESP8266, ESP32, LilyPad, etc.
- Voice: skill e azioni custom per Amazon Alexa e Google Assistant.
- Ambienti di sviluppo: IntelliJ IDEA, Eclipse, Microsoft Visual Studio.
- Application server: Tomcat, JBoss, WebSphere.
- RDBMS: Oracle, SQL Server, DB2, PostgreSQL, MySQL.
- Database NoSQL: MongoDB, HBase, OrientDB, Neo4j.
- Cloud: Amazon EC2, Amazon S3, Google App Engine.
- Source Configuration Management: CVS, Subversion, Git.
- Principali framework e librerie Java utilizzate:
 - Web: Struts, Axis, RestEasy, HttpClient, Jackson, WurfI, JSF (PrimeFaces).
 - Integrazione di sistemi: JTOpen (iSeries e AS/400), Novell JLDAP (eDirectory e LDAP).
 - Caching distribuito: EHcache.
 - Messaggistica multicast: JGroups, Terracotta Toolkit.
 - Scheduling: Quartz.
 - Search: Elasticsearch (certificazioni: “Core Elasticsearch: developer” e “Advanced Elasticsearch: data modeling”), Lucene, Compass, Solr.
 - Persistenza: JPA, Hibernate, MyBatis.
 - Build: Ant, Ivy, Maven.
 - Logging: Log4J, Commons Logging, SLF4J.
 - Test: JUnit, MockEJB, Mockito.
 - Reportistica: iText, JasperReports, Cewolf, JfreeChart, JfreeReport, Apache POI.