ECODIGIT

Ecosistema Digitale per la Fruizione e la Valorizzazione
dei Beni e delle Attività Culturali della Regione Lazio

D2.2 Descrizione del middleware (documento architetturale)

Acronimo Progetto:	**EcoDigit**
Titolo Progetto:	**Ecosistema digitale per la fruizione e la valorizzazione dei beni e delle attività culturali della regione Lazio**

D2.2

Work Package:	WP2 T2.2	
Deliverable Dovuto il:		2 Ottobre 2019
Inizio Progetto:		2 Ottobre 2018
Durata Progetto:		15 mesi
Reponsabile Deliverable:		Massimo Mecella
Versione:		0.7
Stato:		Bozza
Autore:	Miguel Ceriani RM1 Massimo Mecella RM1 Mauro Saccone RM3 Marco Puccini ENEA Chiara Chiarelli CNR	
Altri contribuenti al lavoro riportato nel deliverable:	Valentina Presutti ISTC-CNR Marialuisa Mongelli ENEA Antonio Budano INFN Maria Prezioso RM2 Marco Canciani RM3 Giovanni Fiorentino UNITUS	
Reviewer:	Ludovica Marinucci ISTC-CNR Luigi Asprino ISTC-CNR	

Per citare questo documento si prega di utilizzare il seguente record bibliografico

Miguel Ceriani, Massimo Mecella, Mauro Saccone, Marco Puccini, Luigi Asprino e Chiara Chiarelli. *D2.2 Descrizione del Middleware (documento architetturale)*. Deliverable Progetto EcoDigit. 2019

Revisioni			
Versione	**Data**	**Modificata da**	**Commento**
v 0.1	15/09/2019	M. Ceriani, M. Mecella	Scheletro Documento
v 0.2	20/09/2019	M. Saccone	
v 0.3	20/09/2019	M. Puccini	
v 0.4	14/10/2019	M. Ceriani, M. Mecella	Descrizione del Front End Modulare
v 0.5	04/11/2019	C. Chiarelli	
v 0.6	26/11/2019	M. Ceriani, M. Mecella	Executive Summary e Conclusioni
v 0.7	13/01/2020	M. Ceriani, M. Mecella	Revisione a seguito di feedback da Luigi Asprino

Executive Summary

Il presente deliverable D2.2 “Descrizione del middleware” presenta l’architettura del sistema di gestione di dati e servizi per il progetto ECODIGIT, dal loro acquisizione nel sistema fino alla loro visualizzazione. L’architettura proposta intende estendere la piattaforma Digital Library sviluppata per il progetto Anagrafe con nuove funzionalità. La Digital Library è una piattaforma per l’acquisizione, gestione, interrogazione e visualizzazione di diverse tipologie di contenuti di interesse per la ricerca. Nell’ambito di ECODIGIT la Digital Library viene arricchita ed estesa per gestire una maggiore diversità di contenuti, modelli e formati, in modo da abilitare le applicazioni specifiche previste per ECODIGIT, in particolare la gestione di informazioni su beni culturali con finalità di ricerca e didattica. Per gestire l’acquisizione di molteplici tipologie di contenuti, modelli e formati, viene proposto un insieme di moduli adattatori, configurabili a seconda delle esigenze e delle fonti specifiche. A livello di gestione di questi contenuti, la Digital Library viene arricchita attraverso la memorizzazione dei metadati e le relazioni tra i contenuti in un knowledge graph (mantenuto in un triple store). Infine, viene proposta una nuova struttura per il front end, progettato per essere gestito in maniera modulare e configurabile, associando in maniera dichiarativa le diverse tipologie di contenuti con specifiche componenti di visualizzazione che possono includere risorse e servizi aggiuntivi, locali o remoti. Come componenti front end, vengono progettate tre casi di particolare interesse per i fini di ECODIGIT: interfaccia di ricerca, visualizzatore modelli 3D, visualizzatore GIS.

Indice

Elenco delle figure

1 Introduzione

1.1 Obiettivi del Work Package

L'Obiettivo del Work Package 2 di EcoDigit è definire l'architettura di un componente middleware che estende il sistema Anagrafe. Questo componente middleware deve:

- Garantire l'integrazione, nel sistema Anagrafe, di servizi avanzati per la fruizione e la valorizzazione del patrimonio culturale.
- Facilitare la pubblicazione dei servizi.
- Permettere il riuso composizionale e modulare dei servizi.

Questo Work Package dedicato a garantire la progettazione dei servizi in modo che possano essere usati modularmente e composizionalmente e facilmente integrati nel sistema Anagrafe. A tal fine sarà progettata un'architettura modulare organizzata per servizi e le relative interfacce necessarie a realizzare l'interconnessione tra di essi. Con questo approccio si intende rendere tutti i servizi in grado di comunicare tra loro e di essere usati in modo componibile allo scopo di progettare e implementare altri servizi più complessi e mirati a specifici obiettivi. Ad esempio la definizione di itinerari turistici intelligenti (di potenziale interesse per le imprese), la generazione di percorsi narrativi (utili per la formazione e per la ricerca), la ricerca semantica di contenuti. Sempre all'interno di questo Work Package, in particolare nel task T2.3, sarà sviluppato un prototipo dimostrativo dell'uso composizionale dei servizi.

1.2 Obiettivo del deliverable

Il presente deliverable documenta l'attività del task T2.2, ovvero la progettazione dell'architettura software di EcoDigit, con particolare riferimento a la parte che funge il ruolo di middleware. EcoDigit ed il progetto parallelo Anagrafe contribuiscono a comporre un'architettura software integrata complessiva che risponde ai requisiti dei due progetti e al più generale contesto del Distretto Tecnologico Beni a Attività della Regione Lazio. Anagrafe propone un'architettura basata sulla Digital Library, piattaforma per acquisizione, integrazione, gestione, e visualizzazione di molteplici contenuti, sviluppata dal CNR. L'architettura di Ecodigit include la Digital Library, integrandola con moduli software che permettono di rispondere agli obiettivi specifici di questo progetto, in particolare accrescendone le capacità di acquisizione di fonti molteplici e le modalità di visualizzazione dei dati. Questo deliverable descrive dunque brevemente la Digital Library e poi più in dettaglio come questa è stata estesa nell'ambito di EcoDigit.

1.3 Relazione con le altre attività del progetto

Per i motivi già citati, questo deliverable è relazionato ad attività del progetto Anagrafe. E' inoltre in relazione con altre attività all'interno di EcoDigit. In particolare, il lavoro qui descritto si basa sui risultati dei task T3.1, ovvero l'individuazione delle potenziali sorgenti dati, e T2.1, ovvero i requisiti generali dell'architettura e il rilevamento delle tecnologie utilizzabili. E' stato eseguito in parallelo al task T3.2 che comporta la definizione del modello di integrazione di alcune sorgenti e lo supporta. Il task qui descritto è stato realizzato in coordinazione anche con i task T4.2, T4.3 che riguardano la progettazione di specifici moduli software che l'architettura deve supportare. L'architettura qui descritta serve infine da base per i vari task che portano allo sviluppo del prototipo software finale, che ha l'obiettivo di mostrare fattibilità ed efficacia delle soluzioni proposte (T2.3, T3.3 e T4.4).

1.4 Outline documento

Nella Sezione 2 *Digital Library Management System* si descrive l'architettura della piattaforma della Digital Library, mentre nella Sezione 3 *Estensione Digital Library per acquisizione di dati eterogenei* si descrivono le estensioni a questa piattaforma per consentire l'importazione di dati da fonti esterne verso la Digital Library, per ECODIGIT. La Sezione 4 *Front end modulare* si descrive l'architettura del front end per ECODIGIT, basato parzialmente sul front end della Digital Library. Nella Sezione 5 *Conclusioni*, infine, si tirano le conclusioni, anche in funzione delle fasi successive di progetto.

2 Digital Library Management System

La figura 1 descrive l'architettura della Digital Library alla base del sistema Anagrafe, mettendo in luce il flusso dei dati dai content provider fino ai diversi portali/client che permettono agli utenti ed ai sistemi esterni di accedere ai contenuti. In questa illustrazione i componenti sono raggruppati in modo da rappresentare in modo più esplicito il flusso dei dati, dall'acquisizione alla fruizione.

Scorrendo l'illustrazione dall'alto si può vedere che i contenuti ed i metadati di varia natura, forniti dai diversi provider e codificati secondo diversi standard, sono raccolti dal sistema di harvesting/crawling il cui nucleo centrale è costituito da Apache Manifold[1]. Gli Information Object sono passati al Gestore di code multithreaded (Message Broker) basato su ActiveMQ[2], che li accoda in attesa che vengano processati per l'ingestion. Il motore workflow basato su Activiti[3] applica gli opportuni flussi per la pre-elaborazione di contenuti e metadati utilizzando diversi componenti di servizio (Service Components). Contenuti e metadati vengono poi depositati nei diversi store, in base alla loro tipologia, e fruiti da parte degli utenti attraverso il portale dell'Anagrafe ed altri client.

Per maggiori informazioni riguardo all'Architettura del Digital Library Management System si rimanda al Deliverable 4.1 del progetto Anagrafe.

[1] https://manifoldcf.apache.org/
[2] https://activemq.apache.org/
[3] https://www.activiti.org/

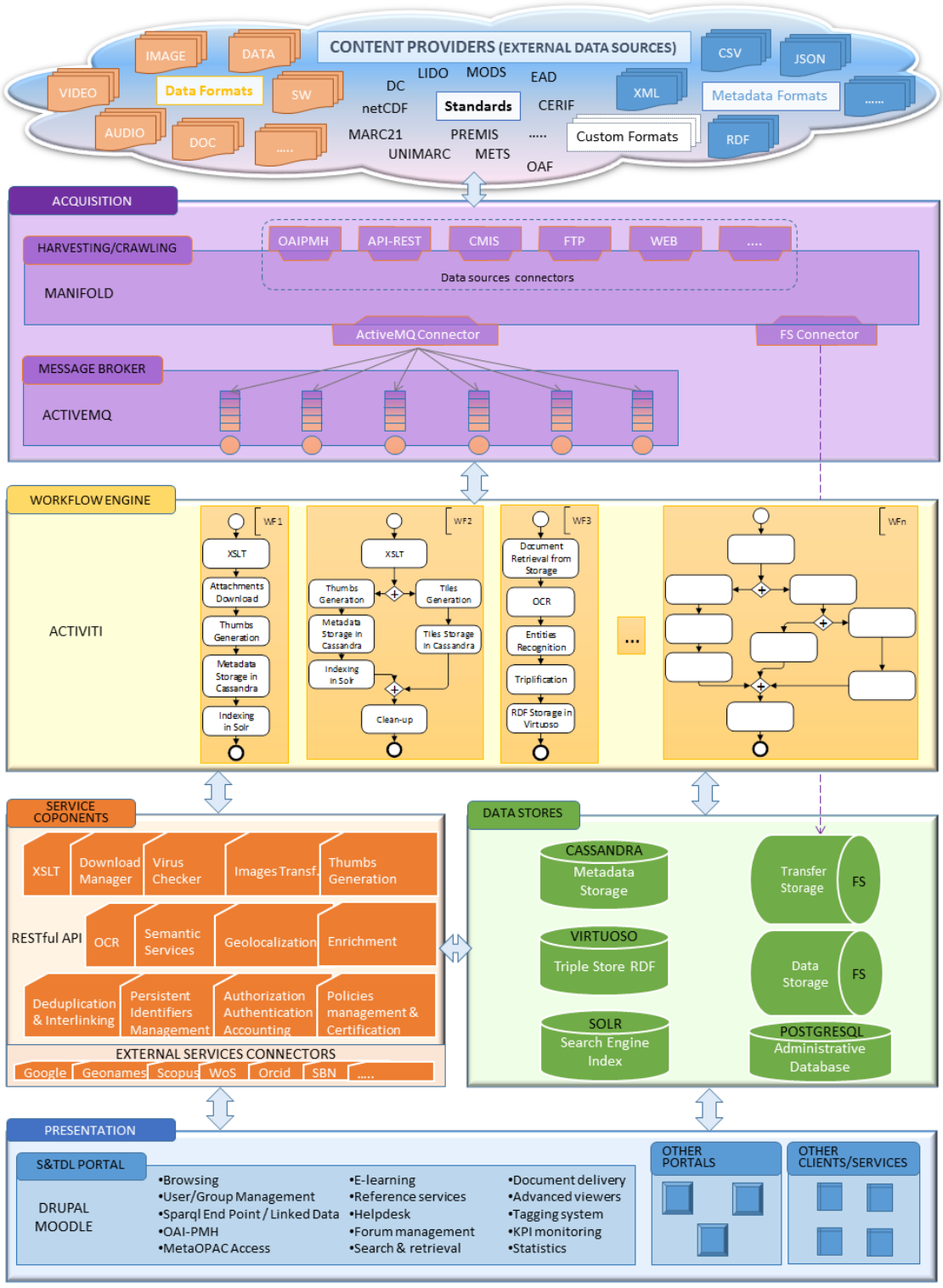

Figura 1: Digital Library: componenti e flussi di dati

3 Estensione Digital Library per acquisizione di dati eterogenei

La Figura 2 mostra il flusso di dati nel sistema, in particolare come possono essere acquisiti nella Digital Library un insieme di dati di tipologie eterogenee, provenienti da molteplici fonti, eterogenee anch'esse. La Figura 2 mostra l'architettura attraverso un diagramma di componenti UML. Le diverse tipologie e fonti considerate sono state individuate a partire dai risultati del questionario analizzato nel deliverable D3.1 ed il confronto con i partner di progetto. Essendo un'estensione, la presente architettura considera tipologie di fonti che non possano essere già acquisite direttamente dalla Digital Library.

Le nuove tipologie di fonti che diventano integrabili grazie a questa architettura sono le seguenti:

- Questionari di Google Forms
- Dati tabulari rappresentati come CSV (Comma Separated Values)
- Grafi di conoscenza RDF esposti attraverso uno SPARQL Endpoint
- Database relazionali accessibili tramite SQL
- Qualsiasi altra sorgente in grado di esportare i dati secondo lo schema concettuale definito nel deliverable D3.2

L'architettura proposta offre gli strumenti affinché una sorgente, originariamente non compatibile con il modello di ingresso del sistema, possa diventarlo. Il deliverable D3.3 descrive nel dettaglio tutti gli strumenti e i metodi che una sorgente può utilizzare per rendere i propri dati conformi al modello di ingresso. Le componenti mostrate nella parte alta della Figura 2 realizzano l'integrazione. Queste componenti, che saranno descritte nelle successive sezioni, realizzano questa integrazione applicando le trasformazioni necessarie.

La Digital Library, attraverso Apache Manifold, accetta file XML che vengono acquisiti ed eventualmente convertiti in modo da rispettare uno di un insieme di schemi XML corrispondenti alle diverse tipologie di dato gestite. Nell'architettura qui proposta i dati, integrati in formato RDF, vengono forniti ad Apache Manifold come RDF/XML, ovvero la serializzazione di RDF sotto forma di XML. Gli stessi dati, inoltre, vengono inseriti in un triple store, ovvero un database che gestisce nativamente dati RDF. Questo permette di che i dati integrati dal sistema possano essere disponibili per eventuali interrogazione attraverso da uno SPARQL endpoint[4] oppure disponibili per la navigazione tramite il componente di dereferenziazione delle IRI LodView.

[4]disponibile alle URL `http://150.146.207.67/yasgui/` (GUI) e `http://150.146.207.67/sparql/ds` (interaccia SPARQL)

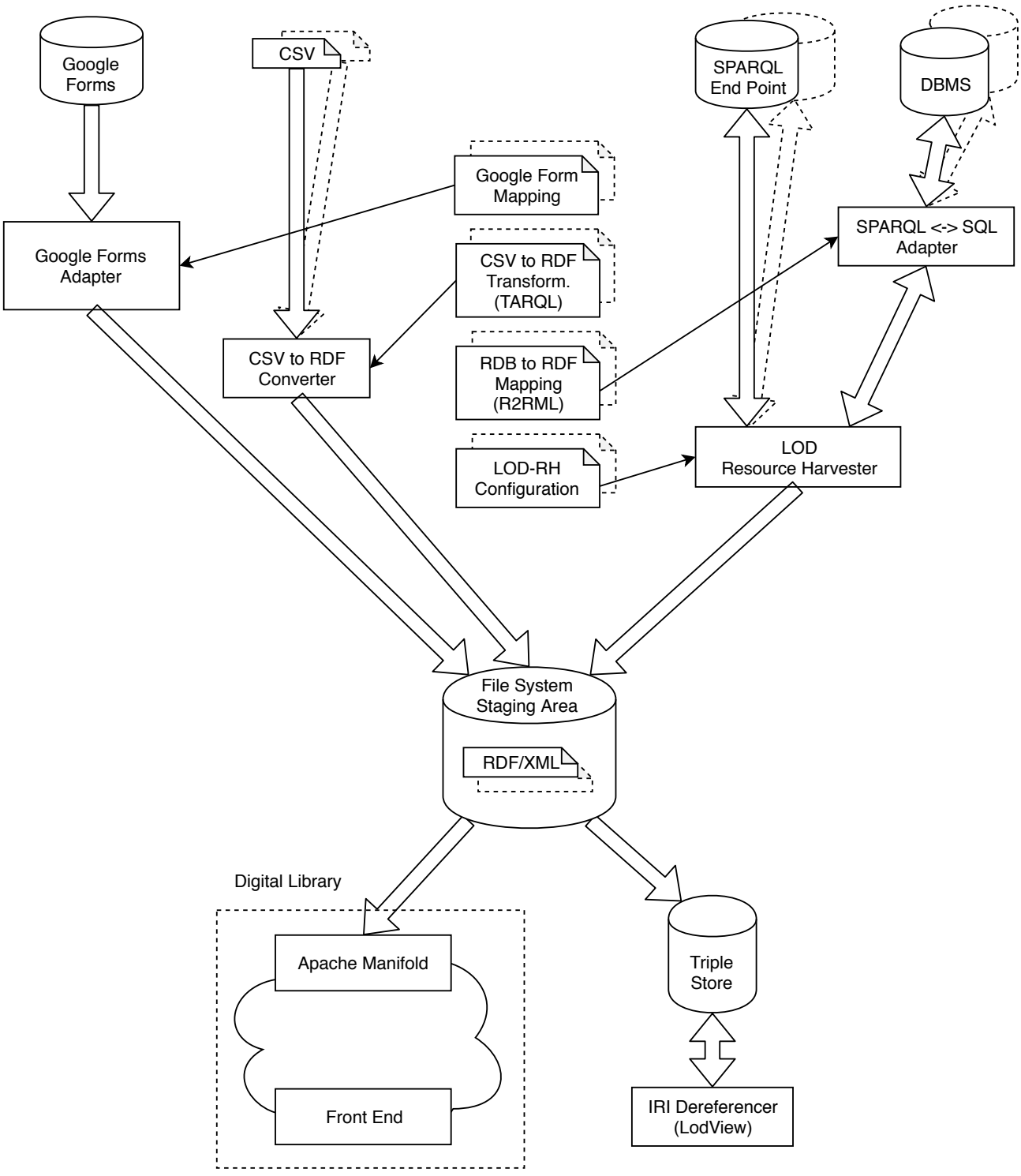

Figura 2: Flusso di acquisizione di dati eterogenei

3.1 Descrizione adattatori

Descriviamo nel seguito la funzione dei singoli componenti utilizzati per adattare ed integrare le diverse tipologie di sorgenti. Alcuni di questi componenti sono preesistenti, mentre alcuni sono stati sviluppati specificamente per ECODIGIT.

3.1.1 Google Forms Adapter

Questo componente si connette alla web API di Google Forms[5] per scaricare i risultati di uno o più questionari online. In questo modo è possibile utilizzare questionari online di Google Forms per inserire (manualmente) contenuti nel sistema. Il questionario sarà stato progettato in funzione dei requisiti specifici di una determinata comunità, predisponendo i campi utili al caso. I contenuti inseriti dagli utenti del questionario vengono convertiti dall'adattatore in RDF: una nuova risorsa viene creata per ogni inserimento ed i contenuti dei campi vengono memorizzati come valori di proprietà RDF. L'output del tool è un file RDF/XML per ogni questionario connesso, che viene aggiornato ogni volta che vengono inseriti nuovi contenuti.

3.1.2 CSV to RDF

Dati tabulari esistenti possono essere convertiti in RDF in modo simile, associando le colonne a proprietà RDF. Diversi standard sono stati sviluppati per convertire i CSV in RDF:

- **CSV2RDF**[6], lo standard del World Wide Web Consortium (W3C) che offre sia una rappresentazione RDF di default utilizzabile per qualunque file CSV, sia la possibilità di definire il mapping a RDF in maniera dichiarativa [5];
- **SPARQL-Generate**[7], un'estensione del linguaggio SPARQL per permettere la generazione di RDF a partire da documenti CSV, XML, JSON e altri formati [4];
- **TARQL**[8], un'altra estensione di SPARQL (ma questa solo semantica, senza modificarne la sintassi) progettata specificamente per manipolare CSV con SPARQL, in particolare permettendone la conversione a RDF attraverso query di tipo CONSTRUCT.

Librerie già esistenti effettuano conversioni con ciascuno di questi standard. In particolare TARQL sembra particolamente adatto agli scopi di questo progetto, perché è specificamente progettato per l'accesso a dati tabulari e non richiede la conoscenza di una nuova sintassi (usa quella di SPARQL).

[5] https://developers.google.com/apps-script/reference/forms
[6] https://www.w3.org/TR/csv2rdf/
[7] http://w3id.org/sparql-generate
[8] http://tarql.github.io/

3.1.3 SPARQL/SQL Adapter

Alcuni contenuti che è interessante importare si trovano in database relazionali strutturati in molteplici modi. Mappare questi dati esistenti su RDF è un problema molto importante nell'ambito del web semantico e c'è stato molto lavoro di ricerca e sviluppo in quest'area. Il W3C ha definito due standard: **Direct Mapping** e **R2RML**. Analogamente al caso dei CSV, il primo di questi standard offre una rappresentazione RDF di default applicabile a qualunque database relazionale, mentre il secondo offre la possibilità di descrivere (in maniera dettagliata e con alto livello di espressività) la mappatura di uno specifico database su RDF [1, 3]. R2RML è più adatto ai nostri scopi in quanto permette di realizzare già la fase di integrazione, in quanto è possbilie rappresentare un database direttamente con i modelli e vocabolari comuni scelti per la relativa tipologia di dati. Diversi tool supportano R2RML, offrendo l'accesso tramite SPARQL endpoint a database relazionali esistenti, mappati mediante R2RML. In questo caso i dati continuano a risiedere nel database, le query SPARQL vengono trasformate in query SQL, eseguite sul database ed infine i risultati riconvertiti dall'output SQL al formato dell'output SPARQL.

3.1.4 Linked (Open) Data Resource Harvester

L'inserimento di contenuti sulla digital library avviene per singoli contenuti in forma "resource-centric": i metadati di ogni contenuto inserito nel sistema sono rappresentati in un file RDF/XML la cui risorsa "principale" rappresenta il contenuto e i singoli metadati sono rappresentati mediante triple RDF che associano la risorsa a valori letterali (data di creazione, numero di pagine, ecc.) o ad altre risorse (autore, contenuto relazionato, ecc.). Alcune fonti integrabili offrono accesso a dataset più complessi, composti da molteplici contenuti relazionati tra loro. Questo vale per i dataset RDF esposti attraverso uno SPARQL endpoint e i database relazionali mappati su RDF utilizzando R2RML o simili. Entrambi questi casi possono essere trattati come dataset RDF (virtuali nel secondo caso).

Per poter integrare queste tipologie di fonti, nell'ambito del progetto è stato elaborato il componente Linked (Open) Data Resource Harvester (LOD-RH). Questo componente permette di selezionare insiemi di risorse di interesse in un dataset RDF, filtrando per tipologia di risorsa o attraverso query più complesse. Per ogni risorsa così individuata, viene estratto un insieme di triple RDF ad essa relazionate (anche questo configurabile, ad esempio prendendo tutte le triple in cui la risorsa selezionata e soggetto o oggetto). Questo insieme di triple viene serializzato in un file RDF/XML incentrato su quella specifica risorsa/contenuto. I file RDF/XML generati vengono inviati alla digital library per l'acquisizione.

4 Front end modulare

Lato front end, il progetto Ecodigit include l'arricchimento della piattaforma Digital Library con un sistema che permette la fruizione di variegate tipologie di contenuto e con diverse modalità, in funzione dei meta dati associati e del contesto di fruizione. Per far ciò viene predisposta un'architettura aperta e modulare per cui si possono connettere al sistema molteplici ed eterogenei moduli di accesso ai contenuti, che sono poi associati a tipologie di dati, meta dati e contesto per mezzo di meccanismi dichiarativi che mantengono la separazione e indipendenza tra dati e moduli software.

Nella Sezione 4.2 viene descritta l'architettura del front-end. Successivamente, le Sezioni 4.1 e 4.2 descrivono rispettivamente l'attuale architettura di front end della Digital Library e come viene estesa per implementare la nuova architettura modulare. Come parte di questa nuova architettura e per soddisfare lo specifico ambito d'uso della piattaforma in Ecodigit, verranno sviluppate in particolare tre specifiche componenti front end, un nuova interfaccia generale di ricerca, un visualizzatore di modelli 3D ed un visualizzatore di dati GIS. Nelle Sezioni 4.3, 4.4 e 4.5 viene descritta l'architettura di queste componenti, in particolare in relazione a come dialogano con il resto del front end.

4.1 Architettura di servizi modulare

L'architettura è stata progettata in modo da permettere la fruizione di diverse tipologie di contenuto e la costruzione dinamica delle pagine in base alla tipologia del contenuto stesso.

L'architettura è basata sul concetto di *vista*. Una vista è quell'insieme di configurazioni e componenti software che permettono di visualizzare un certo tipo di contenuto (es., un visualizzatore di modelli 3D o un visualizzatore di mappe GIS). Nell'architettura proposta, una vista è composta dai seguenti elementi:

- un componente di front end che realizza l'interazione (eventualmente composto da altri sotto-componenti) in base ad una configurazione;
- un'operazione di mapping (definito in forma dichiarativa) dalla risorsa da visualizzare ed i suoi metadati alla configurazione del componente;
- opzionalmente un insieme di servizi indipendenti di supporto al componente, che vengono eseguiti in container separati con un arbitrario stack di tecnologie software.

L'associazione tra tipologie di contenuto e viste viene realizzata tramite un sistema di regole che può essere configurato indipendentemente dalle definizioni interne delle viste.

La figure 4 e 5 mostrano rispettivamente il flusso di dati e l'architettura (come diagramma componenti UML) del front end che implementa queste caratteristiche. Il modulo principale di dell'architettura è il *Dynamic Management of Front End and Services* che acquisisce il contenuto da visualizzare dal *Content Repository* ed i corrispondenti metadati dal *Metadata Store*

e individua la vista da utilizzare in funzione del mapping definito attraverso il componente *Content vs View Mapping Rules* La vista corrispondente viene caricata dal componente *Data View Definitions* ed eseguita, se necessario importando il componente UI necessario (*UI Component x*), che può a sua volta utilizzare altri componenti al suo interno, e lanciando i corrispondenti servizi (*Service y*, *Service z*, ...). Opzionalmente, è possibile utilizzare servizi esterni (*External Services*), sia attraverso il front end interno (nel caso di web API), sia direttamente come parte dell'interfaccia utente (nel caso di applicazioni web con interfaccia utente). Questo permette tra l'altro di accedere a dati (*External Data*) che per diversi motivi si preferisce mantenere all'esterno (ad esempio per motivi legati ai diritti di accesso).

4.2 Tecnologie

E' stato identificato un insieme di tecnologie che possono essere utilizzati per realizzare l'architettura descritta in 4.1.

Complessivamente, il front end è realizzato in maniera da poter ospitare molteplici servizi tra loro indipendenti, che possano essere aggiunti ed eseguiti in maniera dinamica. Per far ciò, si è scelto di utilizzare tecnologie di virtualizzazione che permettano questa separazione. Le tecnologie di virtualizzazione al livello del sistema operativo (OS-level virtualization) permettono di avere un alto livello di indipendenza tra i servizi, pur richiedendo meno risorse rispetto alle soluzioni di virtualizzazione completa (in cui tutto lo stack applicativo, a partire dal sistema operativo, deve essere replicato).

La piattaforma tecnologica che viene proposta per la OS-level virtualization è *Docker*[9], che è in questo momento lo standard de-facto in quest'ambito. In Docker, i moduli che eseguono questi servizi indipendenti sono detti *container* e la loro installazione ed esecuzione e gestita dal *Docker Engine*. La versione statica dei container, ovvero quello che sono prima di essere eseguiti, sono le *immagini*. Le immagini possono caricate e condivise sul repository publico *Docker Hub*.

Quindi il front end sarà costituito da un'istanza del Docker Engine che gestirà tutti container necessari, potendoli all'occorrenza anche andare a scaricare da Docker Hub. Ci sarà almeno un container sempre attivo in ascolto delle connessioni web dall'esterno; questo costituirà il servizio principale del front end, l'entry point del servizio verso l'esterno. L'obiettivo di questo servizio è costruire l'interfaccia utente a partire dai contenuti richiesti e le viste definite, gestendo i componenti UI ed i servizi necessari.

Esistono diverse tecnologie front end per lo sviluppo modulare di applicazioni. Ad oggi una delle più diffuse e supportate è *React*[10], che dispone inoltre di un grande numero di componenti sviluppati e liberamente disponibili. Le componenti UI che utilizzeremo per il prototipo saranno integrate con React, o perchè già nativamente supportanti questo framework o perchè saranno adattate. Il servizio principale sarà quindi un'applicazione React,

[9] https://www.docker.com/
[10] https://reactjs.org/

basata su *Node.js*[11] e *webpack*[12]. L'installazione e gestione dei componenti UI avverrà tramite *npm*[13], il gestore di pacchetti di Node.js. Anche per npm esiste un repository pubblico di pacchetti, *npm registry*, di modo che anche i componenti UI possono essere scaricati ed installati dinamicamente.

4.3 Interfaccia utente di ricerca

Il task 4.4 del progetto ECODIGIT prevede lo sviluppo di un prototipo che mostri le principali funzionalità sviluppate all'interno del progetto. Il prototipo consisterà in un tool che permetterà di effettuare una ricerca avanzata sui contenuti all'interno della Digital Library con la possibilità di visualizzare i risultati anche su una mappa. L'interfaccia grafica sarà composta da una pagina web che contiene un campo di testo per effettuare la ricerca e di un'ulteriore area dove sarà possibile filtrare ulteriormente i contenuti in base a dei criteri aggiuntivi. È in fase di studio l'architettura di una libreria che faccia da interfaccia tra la pagina di ricerca e la digital library. Il prototipo sarà inoltre progettato per permettere una visualizzazione avanzata (rispetto a quella già disponibile nella Digital Library) di contenuti, quali ad esempio oggetti 3D, e la visualizzazione di contenuti non attualmente gestiti dalla Digital Library, come ad esempio mappe GIS.

4.4 Visualizzatore modelli 3D

Il visualizzatore 3DHOP[14] è la collezione di funzioni che utilizzano librerie Open Source javascript per la visualizzazione di modelli tridimensionali. Questo strumento è stato ri-progettato per essere integrato nell'architettura modulare del front-end di Ecodigit. Di default 3DHOP prevede la costruzione di una pagina html relativa ad ogni modello, scrivendo nel sorgente della pagina le funzioni che vogliono essere richiamate, con relativi valori che eventualmente queste richiedono. In questa modalità, seppur offrendo una grande elasticità di utilizzo, questo strumento costringe a scrivere manualmente una pagina per ogni modello. L'obiettivo è quello di rendere invece lo strumento adatto ad ogni modello. A questo scopo è stata pensata un'architettura costituita dai seguenti elementi:

- *gatherer* component: componente che si occupa di raccogliere i metadati utili a descrivere il modello 3D al momento del suo upload nella DL. Tali metadati vengono registrati nella DL da un set di metadati definito describer, relativo al singolo documento (modello 3D).

[11] `https://nodejs.org/`
[12] `https://webpack.js.org/`
[13] `https://www.npmjs.com/`
[14] 3DHOP: 3D Heritage Online Presenter- Marco Potenziani, Marco Callieri, Matteo Dellepiane, Massimiliano Corsini, Federico Ponchio, Roberto Scopigno - Computers & Graphics, Volume 52, November 2015, Pages 129-141, ISSN 0097-8493

- *assembler* component: questo componente si occupa, una volta che l'utente ha selezionato il modello da visualizzare, di raccogliere le informazioni dal suo describer, associarvi le relative funzioni da richiamare, passando a queste eventuali valori.
- 3dhop components: sono le funzioni e le librerie di 3DHOP, riorganizzate in componenti React, che vengono richiamati dall'*assembler*. Queste vengono dunque passate all'engine del costruttore della pagina che li organizza secondo un template predefinito.

Sostanzialmente questi componenti si frappongono tra l'interfaccia utente ed il layer di storage e database, consentendo l'automazione delle operazioni.

Metadati del descrittore Uno dei primi risultati raggiunti è stata la definizione dei campi che costituiscono il set di metadati utili a descrivere i modelli 3D, partendo dalla mappatura delle caratteristiche distintive che questi ultimi rispetto alle funzionalità implementabili nel visualizzatore. Per ognuno dei modelli, al momento del loro upload nell'ecosistema, si dovranno fornire informazioni riguardo (tra parentesi ci sono i valori possibili):

- Identificatore (URI modello)
- Orientazione modello lungo Z (si/no)
- Vista iniziale (Top/Front/Perspective)
- Modello scalato (si/no)
- Unità di misura (m/cm/mm)
- Distanza iniziale (0-n)
- Modello singolo (si/no)
- Per ogni eventuale modello collegato (attivo se precedente: si):
  - Identificatore (URI modello collegato)
  - Collegato (si/no)
  - Riferimento (si/no)

Il costruttore prenderà questi metadati per costruire la pagina attraverso le varie funzioni 3DHOP che saranno opportunamente richiamate e alle quali passerà le variabili. In particolare, per la costruzione di modelli multipli, leggerà tutti i modelli collegati, leggerà quale tra questi deve essere il riferimento, passando queste informazioni alle funzioni 3DHOP che servono a caricare più modelli. Il form per l'upload dei modelli deve dunque prevedere la possibilità di inserire in modo dinamico queste informazioni. Nel caso non si volesse o potesse fornire qualcuna o tutte queste specifiche, sono previsti dei valori di default che consentono in ogni caso, l'assemblaggio della pagina di visualizzazione.

4.5 Visualizzatore dati GIS

Il visualizzatore dati GIS è stato realizzato utilizzando le librerie messe a disposizione del progetto Open Source Openlayers[15]. Oltre alla sua natura Open, esso permette di visualizzare una serie di tipologie di file diversi, anche combinandoli tra loro, accogliendo i dati prodotti dai diversi partner del progetto Ecodigit. Openlayers è una piattaforma scritta in php e javascript, in maniera analoga al visualizzatore 3D, per cui si integra facilmente in pagine web dinamiche o statiche, con la possibilità di richiamare i dati da un database comune.

Le tipologie di dati caricabili sono vettoriali e raster. Concentrandosi sulla prima tipologia, si è scelto per optare sul formato di file KML, perché nativamente conserva le caratteristiche "estetiche" della simbolizzazione, così come definita all'interno dei software GIS dai soggetti produttori. Un altro formato di file utilizzato in eventuale sostituzione del KML è il Json, che - con uno script più articolato - permette di conservare le caratteristiche di simbolizzazione, ma con un peso del file maggiore. Il visualizzatore GIS è stato, inoltre, pensato per due livelli di utilizzo. Un primo livello è quello di visualizzazione spaziale dei risultati di una possibile ricerca, localizzando spazialmente, appunto, i risultati emersi da una query nel database. Cliccando sulla feature scelta, sarà successivamente possibile accedere alla risorsa richiesta. Questo sistema è stato pensato per favorire una lettura spaziale del risultato della ricerca, come valore aggiunto all'elenco dei risultati, ma anche come elemento di valorizzazione dei "siti minori" inseriti del DB. Il secondo livello del visualizzatore GIS permette di visualizzare la geometria della feature scelta. Questo, allegato alla scheda descrittiva, visualizza le caratteristiche del sito oggetto di intervento, ma permette anche di approfondire le informazioni che possono essere lette direttamente attraverso il visualizzatore webGIS. Cliccando, infatti, sulla feature, un pop up mostrerà le informazioni messe a disposizione, come la possibilità, di accedere ad ulteriori risorse, sia quelle presenti nella stessa piattaforma, sia quelle disponibili altrove in rete.

Il visualizzatore all'interno della scheda può ospitare più di una feature contemporaneamente, indipendentemente che esse siano di tipologia diversa (punto, linea, poligono) o formato del file diverso (KML, Json,...), mantenendo le caratteristiche simboliche indicate dai produttori del dato. La scheda, invece, conterrà le informazioni relative all'oggetto visualizzato, nonché al dato stesso, attraverso la pubblicazione dei metadati.

5 Conclusioni

L'architettura software di ECODIGIT, descritta in questo deliverable, è pensata per rispondere ai requisiti di ECODIGIT andando a complementare ed estendere con nuove funzionalità la piattaforma del progetto "Anagrafe". Perciò la Digital Library, piattaforma al centro del-

[15] https://openlayers.org/

la soluzione software per Anagrafe, è stata incorporata, arricchendo ed estendendola alla bisogna. Per ridurre i rischi nello sviluppo e per rendere il sistema più stabile nel tempo, la soluzione proposta è basata su tecnologie e strumenti software esistenti e consolidate. I moduli software di carattere più sperimentale o comunque necessari a dimostrare la piattaforma (obiettivo finale del progetto) sono già stati sviluppati o in fase di sviluppo. In particolare, i task T2.3, T3.3 e T4.4, che riguardano lo sviluppo di specifiche componenti software per il dimostratore, sono già in corso in maniera compatibile con l'architettura proposta grazie al coordinamento realizzato tra quei task ed il presente.

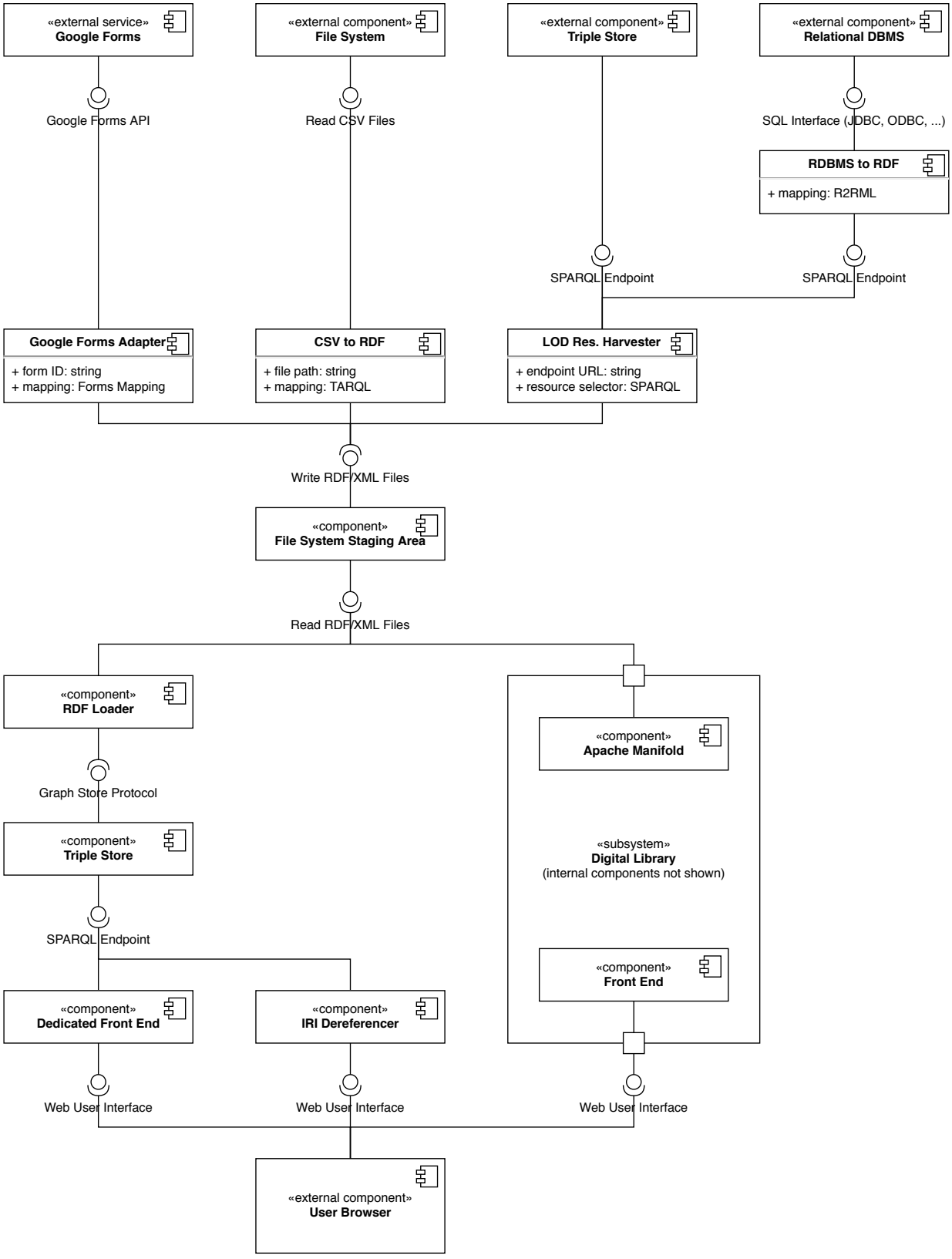

Figura 3: Architettura per l'acquisizione di dati eterogenei

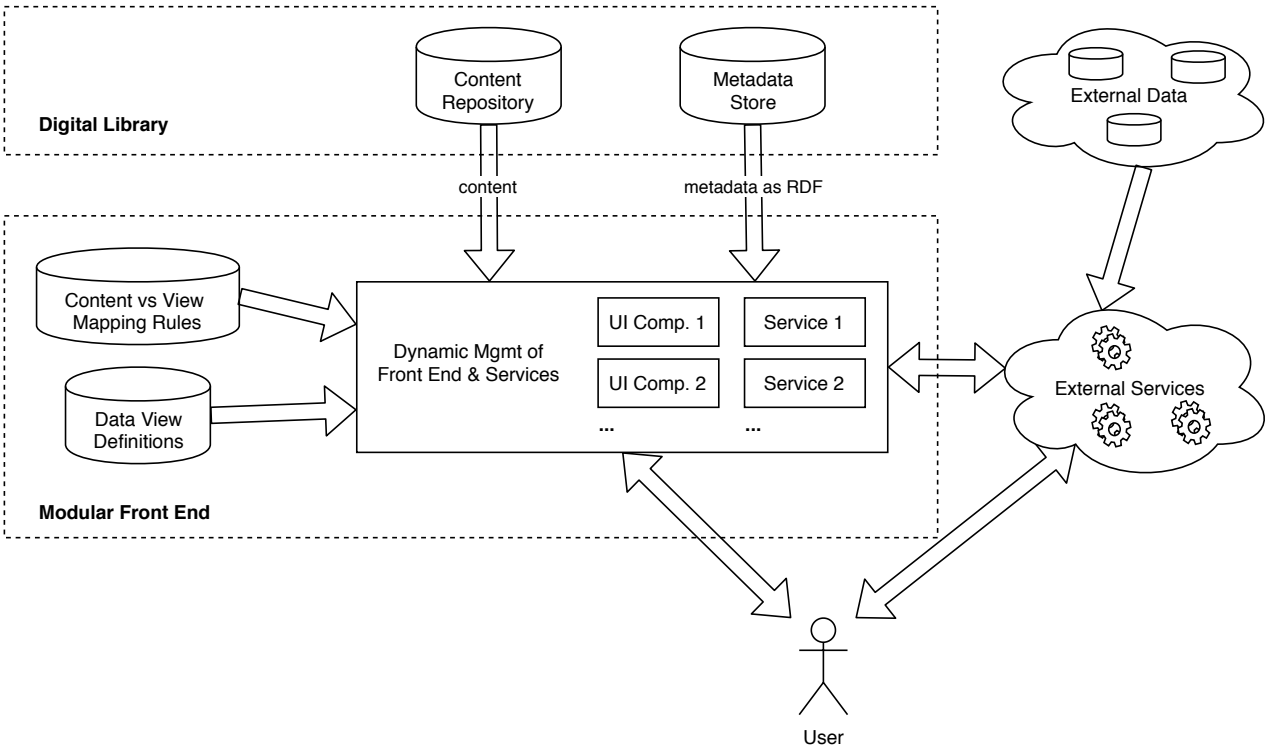

Figura 4: Flusso di dati del front end modulare

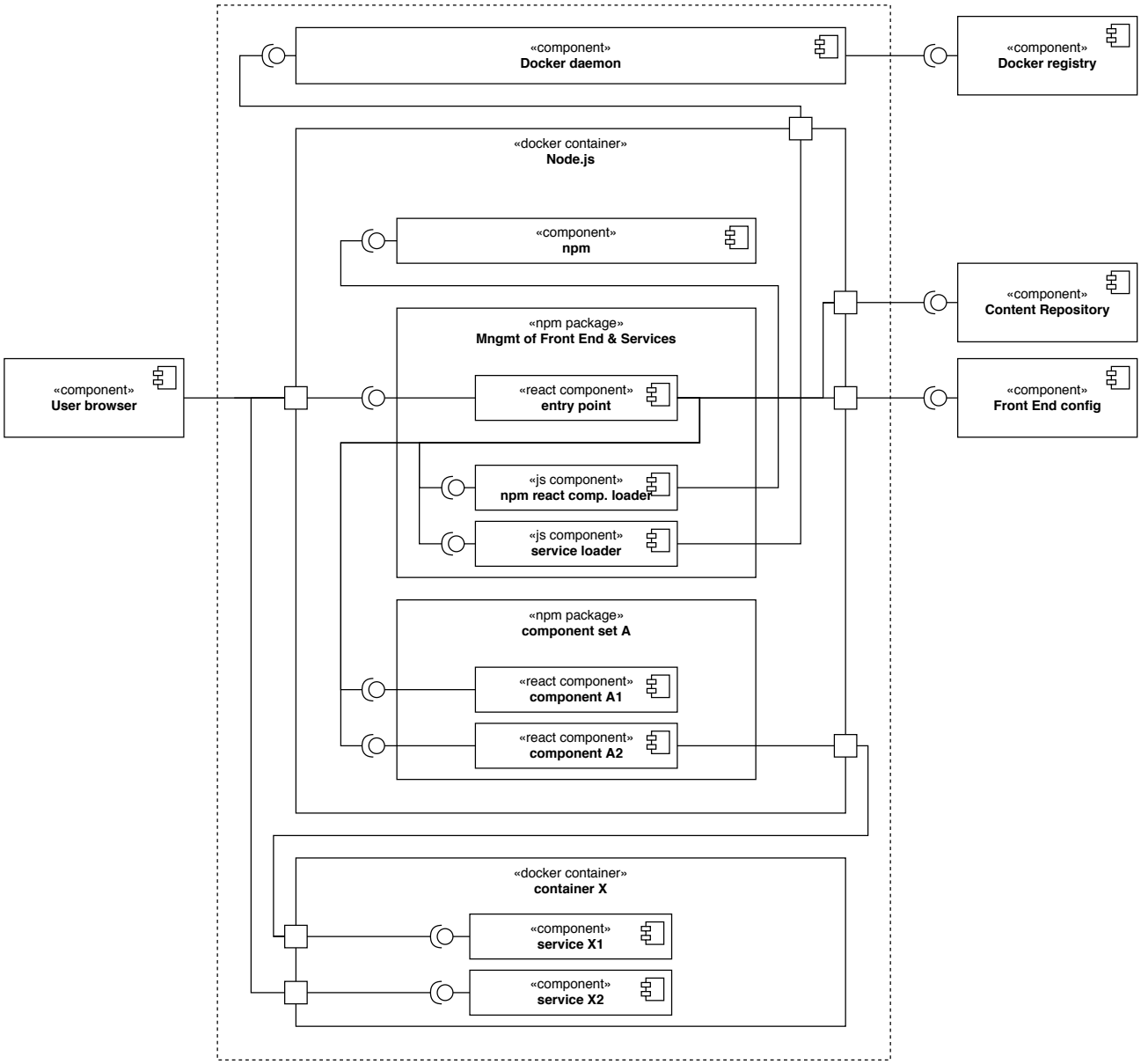

Figura 5: Diagramma componenti UML del front end modulare