ECODIGIT

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

D2.1 Documento di analisi dei requisiti e Survey delle tecnologie

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

D2.1

Work Package:	WP2 - Task 2.1	
Deliverable Dovuto il:		2 Gennaio 2019
Inizio Progetto:		2 Ottobre 2018
Durata Progetto:		15 Mesi
Reponsabile Deliverable:		Massimo Mecella (RM1)
Versione:		1.0
Stato:		Finale
Autore:	Luigi Asprino	ISTC-CNR
	Valentina Anita Carriero	ISTC-CNR
	Ludovica Marinucci	ISTC-CNR
	Andrea Giovanni Nuzzolese	ISTC-CNR
	Valentina Presutti	ISTC-CNR
	Massimo Mecella	RM1
	Miguel Ceriani	RM1
Altri contribuenti al lavoro riportato nel deliverable:	Marialuisa Mongelli	ENEA
	Andrea Quintiliani	ENEA
	Marco Puccini	ENEA
	Francesco Iannone	ENEA
Reviewer:	Alessandro Russo	ISTC-CNR

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

Luigi Asprino, Valentina Anita Carriero, Ludovica Marinucci, Andrea Giovanni Nuzzolese, Massimo Mecella, Valentina Presutti e Miguel Ceriani. *D2.1 Documento di analisi dei requisiti e Survey delle tecnologie*. Deliverable Progetto EcoDigit. 2019

Revisioni			
Versione	**Data**	**Modificata da**	**Commento**
v 0.1	20/11/2018	Luigi Asprino	Creazione Documento
v 0.2	27/11/2018	Andrea Giovanni Nuzzolese e Luigi Asprino	Metodologia
v 0.3	5/12/2018	Valentina Presutti e Luigi Asprino	Analisi Requisiti Funzionali
v 0.4	11/12/2018	Valentina Anita Carriero e Ludovica Marinucci	Survey Modelli Concettuali
v 0.5	18/12/2018	Andrea Giovanni Nuzzolese e Luigi Asprino	Analisi Requisiti Non Funzionali
v 0.6	17/12/2018	Massimo Mecella e Miguel Ceriani	Survey delle Tecnologie Middleware
v 0.7	18/12/2018	Massimo Mecella e Valentina Presutti	Revisione Contenuti e Executive Summary
v 0.8	18/12/2018	Alessandro Russo	Review
v 1.0	20/12/2018	Massimo Mecella e Valentina Presutti	Versione Finale

Executive Summary

Le cinque università statali del Lazio in rete con CNR, ENEA e INFN si candidano a costituire il Centro di Eccellenza del Distretto Tecnologico per i beni e le attività Culturali (DTC) del Lazio. La mission del Centro è costituire un centro di aggregazione ed integrazione di competenze nel settore delle tecnologie per i beni e le attività culturali. In questo contesto, il progetto **EcoDigit-Ecosistema digitale per la fruizione e la valorizzazione dei beni e delle attività culturali del Lazio** ha l'obiettivo di arricchire il sistema Anagrafe delle Competenze del DTC con una piattaforma middleware che faciliti l'integrazione di nuove sorgenti di dati e consenta la pubblicazione e il riuso di servizi per la valorizzazione e la fruizione del patrimonio culturale del Lazio.

Il presente documento, dal titolo "D1.1 Documento di analisi dei requisiti e Survey delle tecnologie", è il risultato delle attività previste dal **Task 2.1**. Ha l'obiettivo di definire la gestione dei processi di: *(i)* definire una metodologia per la raccolta dei requisiti del middleware EcoDigit; *(ii)* l'analisi dei requisiti elicitati; *(iii)* fornire una panoramica delle tecnologie per la realizzazione del sistema middleware.

Il deliverable è composto da una sezione introduttiva che descrive gli obiettivi principali delle attività del Task 2.1 e la sua connessione con le attività degli altri task del progetto. Segue una sezione relativa all'analisi dei requisiti in cui: *(i)* verrà delineata la metodologia di analisi dei requisiti e definiti gli input e output di questa fase; *(ii)* verranno descritte le principali classi di attori del sistema; *(iii)* e verranno elicitati i requisiti funzionali e non funzionali del sistema. Essendo le ontologie uno strumento che favorisce l'integrazione di dati e l'interoperabilità semantica queste giocheranno un ruolo di primo piano nella realizzazione del sistema. Vista l'importanza di questo tipo di tecnologie, verrà fornita nella sezione 3 una panoramica delle principali ontologie nel dominio dei beni culturali. La sezione 4 fornirà un survey delle principali tecnologie middleware per la realizzazione del middleware

Indice

Elenco delle figure

Elenco delle tabelle

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 progettato un componente orchestratore e manager a cui i servizi dovranno essere connessi e le relative interfacce necessarie a realizzare tale connessione. 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

Questo Deliverable ha l'obiettivo di: *(i)* definire una metodologia per la raccolta dei requisiti del middleware EcoDigit; *(ii)* l'analisi dei requisiti elicitati; *(iii)* fornire una panoramica delle tecnologie per la realizzazione del sistema middleware. In particolare questa analisi si interesserà due tipologie di tecnologie che favoriscono l'integrazione di sistemi e l'interoperabilità semantica dei dati: Ontologie adottate nel Semantic Web per il dominio dei beni culturali (cf. Sezione 3) e Tecnologie Middleware (cf. Sezione 4).

1.3 Relazione con le altre attività del progetto

Sarà definita una metodologia per la raccolta dei requisiti che interesserà tutti i partner e tutti i Work Package in modo trasversale. Inoltre i requisiti identificati in questo documento interesseranno tutte le attività di progetto.

2 Analisi dei Requisiti

2.1 Metodologia per l'Analisi dei Requisiti

L'analisi dei requisiti è un'attività preliminare allo sviluppo di un sistema software che ha lo scopo di definire le funzionalità del sistema e i vincoli non funzionali che devono essere soddisfatti dal sistema sviluppato. L'analisi dei requisiti è una fase presente in tutti i modelli di ciclo di vita del software, ma con diverse enfasi e diverse connotazioni. Al fine di contestualizzare questa attività, nei prossimi paragrafi introdurremo brevemente le due principali categorie di modelli di ciclo di vita del software.

Modelli a cascata. Nei modelli di sviluppo detti "a cascata" (Figura 1), l'analisi dei requisiti è la prima fase che deve concludersi con la stesura di una specifica che viene usata come guida nelle successive fasi di sviluppo del sistema. Questa specifica prevede (nelle metodologie più comuni e ampiamente accettate) un diagramma di specifica di casi d'uso in linguaggio UML eventualmente da ulteriore documentazione che descrive le entità del diagramma e descrive eventuali ulteriori vincoli non funzionali che non possono essere catturati dal diagramma dei casi d'uso. Le fasi successive all'analisi di requisiti sono chiamate: progetto, sviluppo, collaudo e manutenzione. La fase di progetto ha lo scopo di determinare come il sistema farà quanto emerso dall'attività di analisi. La fase di sviluppo si occupa di implementare nel software i risultati delle fasi di analisi e di progetto. Nella fase di collaudo viene verificata la correttezza del software sviluppato così come altri vincoli non funzionali stabiliti in fase di analisi. Nella fase di manutenzione il software viene rilasciato al cliente e vengono effettuate tutte le attività volte a migliorare, estendere e correggere il sistema nel tempo.

Modelli iterativi e incrementali. Nella categoria dei modelli iterativi e incrementali (Figura 2) rientrano tutti i processi di sviluppo che ritornano ripetutamente su una fase affrontata in precedenza al fine di migliorare i risultati della iterazione precedente. Questa categoria di modelli sono stati introdotti per ridurre il rischio di fallimento intrinseco dei modelli a cascata. Infatti, l'impossibilità di migliorare i risultati di una fase con delle successive iterazioni aumenta il rischio di fallimento del progetto. Nei modelli iterativi e incrementali i risultati delle prime iterazioni sono degli artefatti documentali e prototipi di software che verranno raffinati nelle successive iterazioni. Ad ogni iterazione i risultati sono migliorati e viene aggiunto valore al prodotto.

In maniera simile nei modelli di sviluppo "iterativi e incrementali" , l'analisi dei requisiti è la prima fase di ogni ciclo di sviluppo e fornisce una specifica che viene usata come guida nelle fasi successive di progettazione, sviluppo, collaudo, rilascio, manutenzione e valutazione. A differenza dei modelli a cascata, nei modelli iterativi ed incrementali i risultati della fase di manutenzione e valutazione diventano degli input per la fase di analisi dei requisiti.

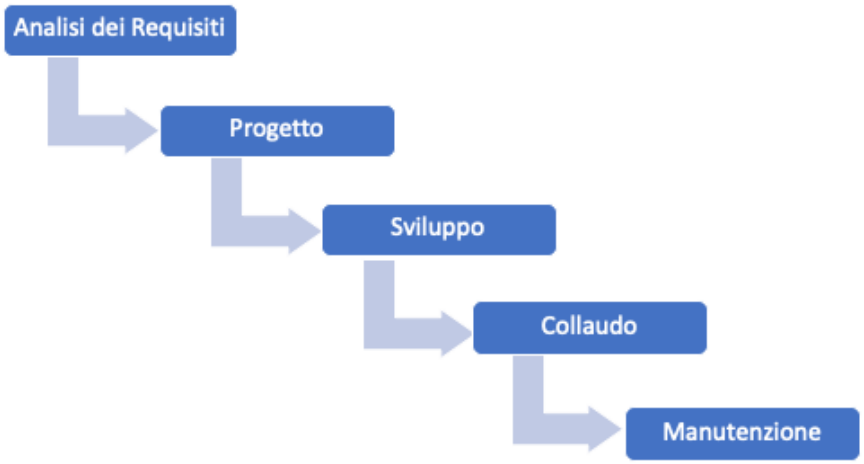

Figura 1: Modello di sviluppo a cascata.

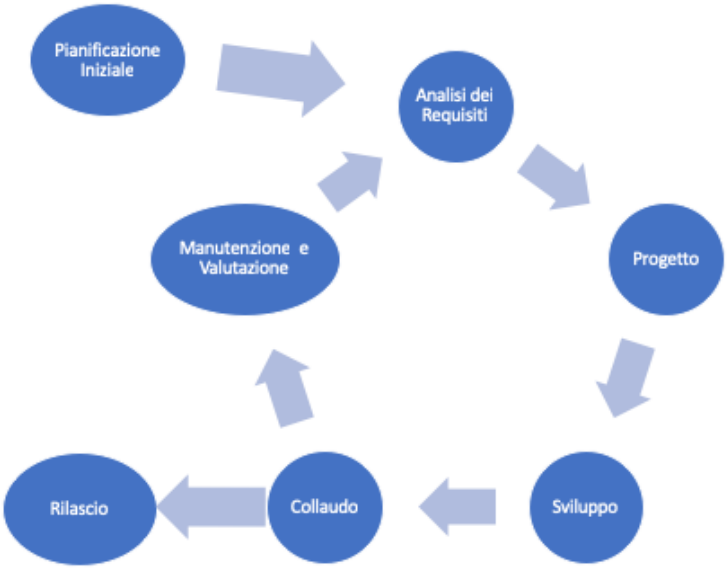

Figura 2: Modello di sviluppo iterativo e incrementale.

Input della fase di analisi. Questa analisi viene effettuata normalmente analizzando il dominio applicativo del sistema che si deve sviluppare attraverso due attività principali:

1. l'analisi della documentazione pre-esistente (es. una descrizione del sistema fornita dal committente);
2. l'intervista degli stakeholder (utenti o altro tipo di attori che dovranno interagire con il sistema) del sistema;
3. l'intervista a esperti di dominio;
4. la documentazione disponibile su sistemi simili o correlati (es. sistemi che realizzano funzionalità simili, compeditor di mercato o sistemi con cui il sistema in oggetto deve interagire) al sistema in oggetto;
5. survey della letteratura scientifica sul tema;
6. e, nel caso in cui il sistema software sia stato già rilasciato, i risultati della fase valutazione o riscontri degli utenti del sistema.

Output della fase di analisi. L'output della fase di analisi sono uno o artefatti documentali che *(i)* descrivano le funzionalità che il sistema deve realizzare; *(ii)* definiscano i requisiti non funzionali del sistema; *(iii)* descrivano gli attori del sistema e funzionalità che il sistema gli fornisce. Tali documenti devono essere redatti in una forma facilmente comprensibile, il più possibile precisa, completa, coerente e non ambigua. Nei processi iterativi e incrementali è importante che sia possibile apportare facilmente modifiche ai prodotti della fase di analisi in modo da permettere una evoluzione dei documenti nelle varie fasi di vita del software. Negli ultimi anni tecniche e dei linguaggi sono stati proposti per migliorare la comunicazione dei risultati della fase di analisi. UML (Unified Modeling Language)[1] è uno standard largamente accettato per (tra le altre cose) la descrizione delle funzionalità di un sistema e di come queste vengono percepite dagli attori di un sistema software. In linguaggio UML il diagramma fondamentale che definisce i requisiti funzionali di un sistema è lo "Use Case Diagram". In alcune metodologie di sviluppo software "agili" questo diagramma viene spesso accompagnato da delle "user story". Una user story è una storia scritta in linguaggio naturale che descrive una o più caratteristiche o funzionalità che il sistema deve implementare. Queste vengono definite usando dei template standard, ad esempio:

```
Come <utente>, vorrei <funzionalità>, così che <beneficio>
```

Queste storie vengono collezionate con attraverso lo studio degli input della fase di analisi. Per alcuni aspetti il diagramma UML dei casi d'uso e le user story si sovrappongono (e.g. entrambi possono essere usati per descrivere le funzionalità di un'applicazione). Oltre a fornire un linguaggio visuale (quindi più intuitivo) invece che descrittivo, le user story non

[1] http://www.uml.org/

permettono di definire relazioni tra attori del sistema (e.g. tassonomia di attori), relazioni tra casi d'uso (e.g. estensione, inclusione tra casi d'uso del sistema). Diversamente da UML, le user story permettono di esprimere vincoli non funzionali e descrivere dei requisiti che potrebbero essere troppo specifici per un linguaggio visuale come UML (e.g. sequenza dei casi d'uso).

Metodologia di raccolta analisi proposta per EcoDigit. Il middleware EcoDigit, e in generale tutta la piattaforma DTC Lazio, non possono prescindere da un modello di sviluppo software iterativo e incrementale ispirato a metodologie "agili" per due motivi principali. *(i)* Lo sviluppo della piattaforma è previsto in due fasi incrementali: una prima fase che si occupa di realizzare un sistema in forma prototipale, una seconda che si occuperà di mettere il sistema in produzione. *(ii)* La piattaforma deve essere pronta ad accogliere nuovi stakeholder che vogliano partecipare, contribuire, beneficiare del sistema middleware. Per garantire questo la metodologia di progettazione deve seguire la filosofia delle metodologie di sviluppo agili di "abbracciare il cambiamento", cioè seguire una metodologia di sviluppo che permetta di accogliere sempre nuovi requisiti.

Gli input per la fase di analisi per EcoDigit sono stati:

1. *Analisi Documentazione Pre-esistente*. Come documentazione pre-esistente abbiamo analizzato la descrizione del lavoro del progetto del Centro di Eccellenza - DTC Lazio. Questo documento definisce gli obiettivi del Centro di Eccellenza e quindi fornendo delle indicazioni sui requisiti funzionali e non che il middleware EcoDigit deve soddisfare per raggiungere tali obiettivi.

2. *Intervista degli stakeholder*. Due tipi di attività sono state avviate in questo senso. Una prima attività di censimento è stata avviata sin dall'inizio del progetto al fine di censire le sorgenti potenziali da aggregare attraverso EcoDigit. Questa attività svolta nell'ambito del Task 3.1 ha prodotto un questionario che è stato diffuso sul territorio. Una prima analisi delle risposte al questionario è presentata nel Deliverable D3.1 [1]. Quest'analisi ha fatto emergere un'ampia eterogeneità di sistemi che potrebbero interagire con il middleware EcoDigit. Le caratteristiche di questi sistemi hanno quindi influito sui requisiti del sistema EcoDigit. Una seconda attività di intervista ha interessato il gruppo di sviluppo del progetto Anagrafe. Da specifiche di progetto EcoDigit si interfaccerà con il progetto Anagrafe per offrirgli funzionalità di acquisizione e arricchimento della propria base dati. Di conseguenza, essendo il sistema Anagrafe di fatto uno stakeholder di EcoDigit, i requisiti di EcoDigit dovranno tenere in considerazione della struttura e dei vincoli tecnologici imposti da Anagrafe.

3. *Intervista a esperti di dominio.* Oltre all'attività di censimento che è di fatto un'intervista ad esperti di dominio, abbiamo ricevuto input da parte dei partner di progetto i quali vantano una lunga esperienza nel dominio dei beni culturali. L'analisi presentata in questo documento è frutto del contributo delle esperienze dei partner.

4. *Anche della documentazione di sistemi simili o correlati.* Anche se non si può considerare propriamente documentazione, il sistema esistente del centro di eccellenza ci ha fornito delle indicazioni sui requisiti che il nuovo sistema deve rispettare.

5. *Survey della letteratura scientifica sul tema.* L'obiettivo del sistema middleware è quello di dare uno strumento che favorisca l'integrazione di dati e software. Per quanto riguarda i dati, gli strumenti privilegiati per l'integrazione sono gli standard di fatto definiti per il Semantic Web e tra questi le ontologie come strumento per la definizioni di modelli dati condivisi. Per quanto riguarda il software le tecnologie middleware permettono l'integrazione di artefatti software in maniera del tutto neutrale rispetto la tecnologia con il quale sono stati sviluppati. Nelle sezioni 3 e 4 viene fornita una panoramica delle tecnologie che favoriscono l'integrazione di dati e software.

6. *Risultati della valutazione del sistema effettuata in iterazione precedenti*. Dato che il sistema non è stato rilasciato non sono disponibili valutazioni. In interazioni successive però la valutazione degli utenti del sistema costituirà un input per la definizione dei requisiti.

Nelle sezioni successive verranno presentati gli output della fase di analisi in termini di Requisiti funzionali (Sezione 2.2) e Requisiti non funzionali (Sezione 2.3).

2.2 Requisiti Funzionali

La descrizione dei requisiti funzionali è composta da un'analisi degli attori del sistema e un'analisi dei casi d'uso.

Analisi Attori. Questo tipo di analisi mira ad identificare quali sono le classi di attori che interagiranno con il sistema. Sono state identificate quattro classi di attori che sono state opportunamente specializzate dove necessario:

1. *Fornitore.* Il fornitore di contenuti si occupa di fornire dati, metadati, servizi avanzati o funzionalità al sistema EcoDigit. La figura del *Fornitore* viene specializzata in *fornitore di dati* e *fornitore di servizi*. La classe dei fornitori di dati racchiude tutte le entità interessate a fornire dati al sistema, siano essi dati veri e propri (e.g. un prodotto della ricerca) o metadati (e.g. descrizione del contenuto di un dataset che il fornitore detiene). La classe dei fornitori di servizi racchiude tutte le entità interessate a condividere i propri servizi avanzati o funzionalità attraverso il middleware EcoDigit.

2. *Fruitore.* Il fruitore di contenuti si occupa di usare dati, metadati, servizi avanzati o funzionalità offerte dal sistema EcoDigit per i propri scopi. La figura del *Fruitore* viene specializzata in *fruitore di dati* e *fruitore di servizi*. La classe dei fruitori di dati racchiude tutte le entità interessate a interrogare, usare i dati e metadati offerti dal sistema EcoDigit. La classe dei fruitori di servizi racchiude tutte le entità interessate a riutilizzare, per i propri interessi, servizi avanzati e funzionalità offerte dal middleware EcoDigit.

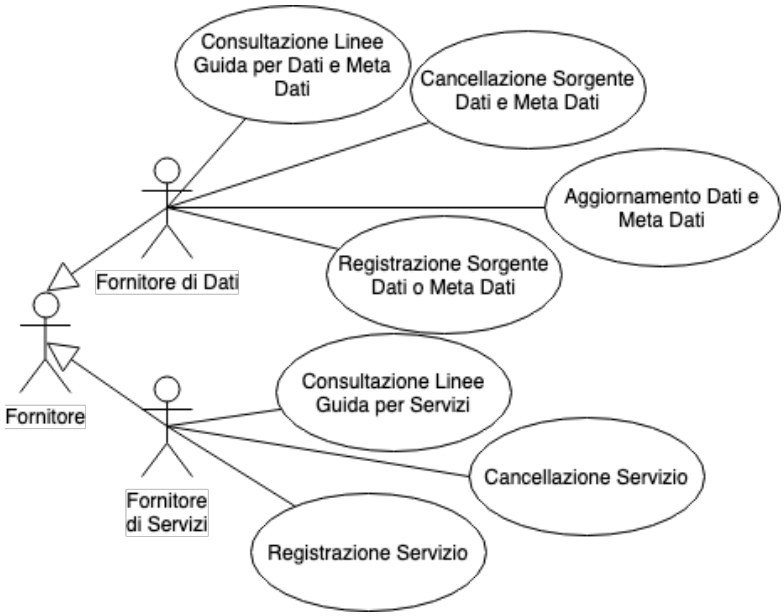

Figura 3: Il diagramma dei casi d'uso relativi alla classe di attori denominata *Fornitore*.

3. *Gestore di Linee Guida.* Il gestore di Linee Guida si occupa di definire le linee guida che le sorgenti (sia dati che software) devono soddisfare per essere integrate all'interno del middleware. Anche questo caso la figura del gestore di linee guida è specializzata in Gestore di Linee Guida di dati (i.e. si occupa di definire gli schemi che i dati e metadati devono rispettare per essere integrati in EcoDigit) e di servizi (i.e. si occupa di definire gli schemi che servizi avanzati e funzionalità devono rispettare per essere integrati in EcoDigit). Queste figure non necessitano dei privilegi di amministratore di sistema.

4. *Amministratore.* La figura dell'amministratore di sistema ha controllo su tutta la piattaforma e può agire su ogni suo componente. Il suo ruolo è quello di gestire la piattaforma garantendo che tutte le sue componenti funzionino in modo corretto.

Analisi Casi d'Uso. In questa sezione verranno presentati i casi d'uso del sistema. I casi d'uso sono descritti attraverso diagrammi UML e accompagnati da una descrizione testuale. Per facilitarne la lettura verranno raggruppati a seconda della classe di attore al quale si riferiscono.

La Figura 3 mostra i casi d'uso relativi alla classe di attori denominata *Fornitore*. Un *Fornitore di Dati* può *Consultare le Linee Guida per Dati e Meta Dati* per strutturare i dati o i meta-dati che intende fornire in un formato compatibile con il middleware EcoDigit. Se la sorgente è strutturata secondo le Linee Guida, può *Registrare la Sorgente di Dati e Meta Dati* così da permettere al middleware l'acquisizione. Se il fornitore di dati ha già registrato una sorgente ha la capacità di *Aggiornare i dati o meta dati forniti* da essa o eventualmente *cancellare la sorgente* dal middleware. Analogamente un *Fornitore di Servizi* può *consultare le linee guida per i servizi* per verificare se il servizio a sua disposizione è compatibile con il

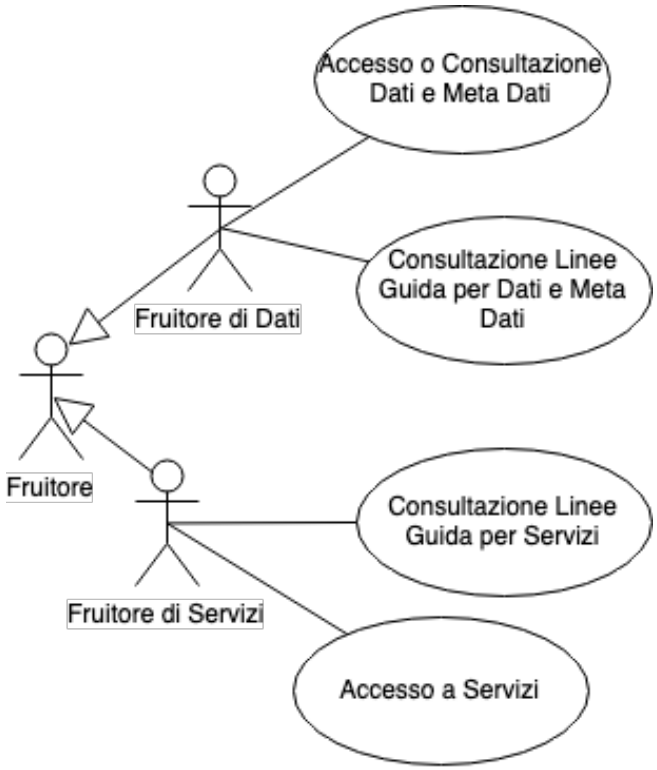

Figura 4: Il diagramma dei casi d'uso relativi alla classe di Attori denominata *Fruitore*.

middleware EcoDigit e nel caso può modificarlo per garantirne la compatibilità. Se il servizio rispetta le linee guida il fornitore di servizi potrà registralo presso il middleware EcoDigit che acquisirà i metadati (descrizione delle funzionalità offerte dal servizio). Una volta registrato, un fornitore di servizi potrà eventualmente cancellare il servizio da EcoDigit.

La Figura 4 mostra i casi d'uso relativi alla classe di attori denominata *Fruitore*. La classe dei *Fruitori* racchiude tutte le entità che sono interessate ad accedere e riutilizzare dati e servizi integrati e forniti dal middleware EcoDigit. In particolare, i *Frutitori di Dati* potranno *accedere e consultare dati e meta-dati* raccolti e forniti dal middleware. Inoltre i Fruitori di Dati potranno *consultare le linee guida per dati e meta-dati* le quali forniscono delle descrizioni sulla struttura delle sorgenti. Le linee guida definite in EcoDigit non saranno soltanto uno strumento di aggregazione delle sorgenti ma, definendo la struttura delle sorgenti, forniranno anche la documentazione necessaria per accedervi. Analogamente, i *Fruitori di Servizi* potranno *consultare le linee guida per i servizi* per apprendere le struttura dei servizi censiti da EcoDigit e ricevere le indicazioni sulle tecnologie con i quali sono stati realizzati. Infine, un *Fruitore di Servizi* può accedere al servizio offerto dal middelware.

La Figura 5 mostra i casi d'uso relativi alla classe di attori denominata *Gestore*. Il ruolo del *Gestore* all'interno del sistema EcoDigit viene associato a tutte quelle funzionalità relative alla definizione e aggiornamento di linee guida. In particolare, il *Gestore di Dati* si occuperà di gestire (cioè definire o aggiornare) le linee guida per i dati e meta-dati. Come emerso dal D3.1 Report sul censimento [1] sono presenti sorgenti afferenti ad una eterogeneità di

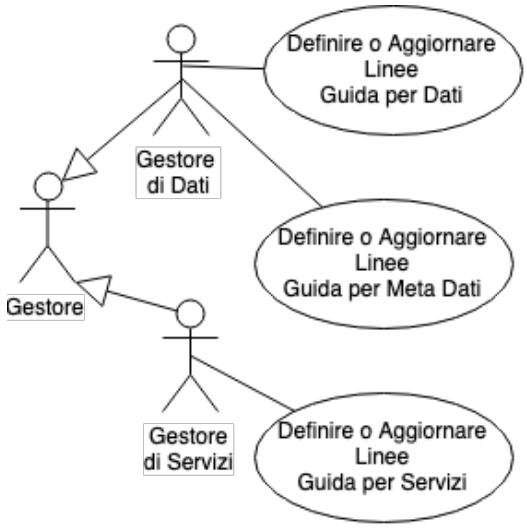

Figura 5: Il diagramma dei casi d'uso relativi alla classe di attori denominata *Gestore*.

domini di conoscenza. Per questo motivo, è opportuno che le linee guida vengano gestite per dominio, cioè, per ogni dominio che è emerso dal censimento (o che emergerà durante la vita del sistema), i gestori si dovranno occupare di dare delle linee guida su come strutturare quel dominio per essere compatibili con il middleware EcoDigit. Il *Gestore di Servizi*, in maniera analoga a quello dei dati, definisce e aggiorna le linee guida per i Servizi.

La Figura 6 mostra il diagramma dei casi d'uso per la classe di attori denominata *Amministratore*. L'amministratore ha potere su tutta la piattaforma e può agire su ogni componente. Di seguito ci concentreremo sulle funzionalità disponibili previste per l'amministratore. In particolare, come visto finora, le entità del dominio di EcoDigit, e quindi su cui potrà agire l'amministratore, sono: Servizi, Dati, Meta-Dati, Linee guida e utenze. Con gestione si deve intendere un'azione volta alla creazione, aggiornamento, cancellazione di una entità. In particolare, un amministratore può *gestire le utenze*: creare nuove utenze, aggiornare i privilegi di utenze esistenti, o cancellare utenze esistenti. Un amministratore può *gestire servizi*: aggiornare i meta-dati relativi ad un servizio censito dal middelware, inserire o cancellare i meta-dati di un servizio. Un amministratore può *gestire una sorgente di dati o meta-dati*: può includere una nuova sorgente, modificare i dati relativi ad una sorgente, cancellare una sorgente. Un amministratore può *gestire le linee guida*: modificare le linee guida, aggiungere nuove linee guida, cancellare linee guida esistenti.

2.3 Requisiti Non Funzionali

Vista la grande eterogeneità emersa dal D3.1 Report sul censimento [1], i requisiti non funzionali più critici per la piattaforma riguarderanno: l'Interoperabilità, l'Adattabilità, l'Integrabilità e l'Estensibilità. I requisiti sono riportati in forma testuale in Tabella 1.

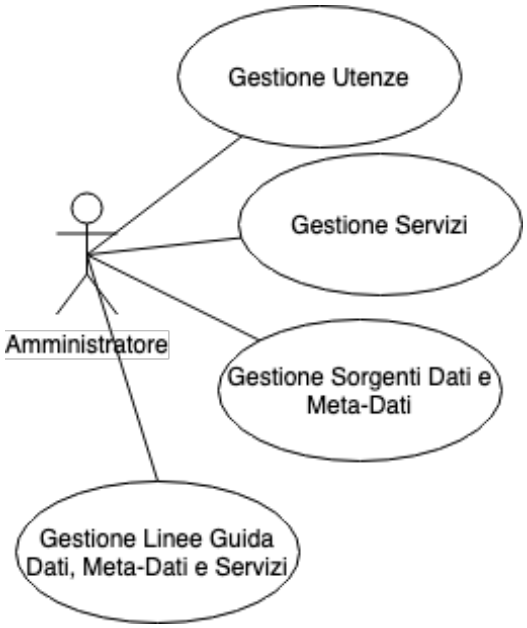

Figura 6: Il diagramma dei casi d'uso relativi alla classe di attori denominata *Amministratore*.

3 Survey dei Modelli Concettuali di Riferimento nel Dominio dei Beni Culturali

Nel dominio dei beni culturali sono state elaborate diverse ontologie e sistemi di organizzazione della conoscenza (knowledge organisation systems - KOS), frutto del sempre maggiore interesse verso le tecnologie semantiche e i Linked Open Data (LOD) da parte delle istituzioni volte alla conservazione e valorizzazione dei beni culturali, per favorire l'accesso al patrimonio culturale e il suo arricchimento.

In ambito italiano, l'ICCD (Istituto per il Catalogo e la Documentazione - MiBAC) ha avviato con l'ISTC-CNR un progetto, ArCo (Architettura della Conoscenza)[2], volto alla pubblicazione di una rete di ontologie per i beni culturali e dei dati sul patrimonio culturale italiano, raccolti nel SIGECweb e riversati nel Catalogo Generale dei Beni Culturali, in LOD.

La versione attuale della rete di ontologie e dei dati è la 0.4, ma è prevista la pubblicazione di una prima versione stabile entro il mese di Aprile 2019. Le ontologie rappresentano informazioni di vario tipo (dalla cronologia, alle misure, ai materiali, alla localizzazione, all'attribuzione di autore, alla bibliografia associata alla scheda di catalogo, etc.) relative a diverse tipologie di beni culturali (archeologici, storici e artistici, demoetnoantropologici, scientifici e tecnologici, numismatici, architettonici e paesaggistici, fotografici, musicali), in accordo con le Normative per la catalogazione emanate dall'ICCD.

Le ontologie di ArCo fanno un riuso diretto di Cultural-ON[3] [4], un'ontologia che modella

[2] http://wit.istc.cnr.it/arco
[3] http://dati.beniculturali.it/cis/

ID	Tipo	Requisito
R1	Adattabilità, Estensibilità	Il sistema deve permettere una evoluzione agile dei modelli concettuali indicati per i domini censiti e degli schemi di metadatazione.
R2	Adattabilità, Estensibilità	Il sistema deve poter accogliere dati definiti con diverse sintassi e strutturate secondo un qualsiasi schema concettuale.
R3	Adattabilità, Estensibilità	Il sistema deve permettere di poter manipolare i dati e metadati in ingresso in modo da trasformarli secondo lo schema concettuale e di metadatazione deciso per il sistema.
R4	Integrità dei dati	I dati acquisiti dal sistema devono essere riconducibili ad un unico schema di metadatazione che può essere opportunamente esteso secondo per le varie tipologie di dati.
R5	Disponibilità dei dati	Il sistema deve permettere di interrogare i dati e metadati dei dataset acquisiti.
R6	Disponibilità dei dati	Il sistema deve il indicizzare dati e metadati acquisiti.
R7	Estensibilità	Il sistema deve permettere l'aggiornamento, la modifica e l'aggiunta di dati e metadati.
R8	Estensibilità	Il sistema deve permettere lo sviluppo di nuovi servizi integrati per la profilazione, accresimento semantico, analisi semantica ecc. (cf. Sviluppo Incrementale e itrativo).

Tabella 1: Requisiti non funzionali.

istituti ed eventi culturali, anch'essa frutto della collaborazione tra ICCD e ISTC-CNR, e dei moduli di OntoPiA[4], un set di ontologie e di vocabolari controllati, elaborato per la Pubblica Amministrazione, che modella informazioni di alto livello e trasversali a diversi domini (luoghi e indirizzi, ruoli, coordinate temporali, servizi pubblici, strutture ricettive, etc.)

Classi e proprietà di ArCo verranno inoltre allineati ad altre ontologie del dominio, tra cui CIDOC-CRM[5], Europeana Data Model[6], BIBFRAME[7] e FRBRoo[8] (modelli per la descrizione bibliografica), FEntry[9] e OAEntry[10] (ontologie specifiche per le fotografie e le opere d'arte), RiC-O (Records in Context Ontology, ontologia sulla descrizione archivistica in via di definizione). Gli allineamenti riguarderanno anche i vocabolari controllati del Getty Research

[4] https://github.com/italia/daf-ontologie-vocabolari-controllati
[5] http://www.cidoc-crm.org/cidoc-crm/
[6] https://www.europeana.eu/schemas/edm
[7] http://id.loc.gov/ontologies/bibframe/
[8] https://www.ifla.org/files/assets/cataloguing/FRBRoo/frbroo_v_2.4.pdf
[9] http://www.essepuntato.it/2014/03/fentry
[10] http://purl.org/emmedi/oaentry

Institute[11]

Due modelli ampiamente adottati in progetti che vedono la pubblicazione di dati sui beni culturali sono: CIDOC-CRM e Europeana DM [2].

CIDOC - Conceptual Reference Model è stato sviluppato per i musei, ma è un'ontologia di riferimento per il settore dei beni culturali in generale. Questa ontologia ha un approccio evento-centrico: modella cioè il bene culturale in relazione agli eventi che lo coinvolgono (creazione, acquisizione, spostamento da un luogo a un altro, etc.).

L'Europeana Data Model è sviluppato per rappresentare i dati sul patrimonio culturale in modo che possano confluire nella piattaforma Europeana. Questo modello ha un approccio oggetto-centrico e riusa altri modelli, come Dublin Core, collocandosi a un livello top-level e rinunciando quindi alla rappresentazione di informazioni dettagliate sui beni culturali.

[11] http://www.getty.edu/research/tools/vocabularies/index.html

4 Survey delle Tecnologie Middleware

Con middleware si intende un componente software che funge da intermediario tra diverse applicazioni e componenti software. I middleare sono spesso utilizzati come supporto per sistemi distribuiti complessi con architetture multitier. L'integrazione dei processi e dei servizi, residenti su sistemi con tecnologie e architetture diverse, é un'altra funzione dei middleware.

Originariamente il middleware ha iniziato a guadagnare popolarità durante gli anni ottanta, come soluzione al problema di connettere le nuove applicazioni ai vecchi sistemi preesistenti (sistemi legacy), sebbene il termine sia stato usato fin dal 1968[12]. Il middleware inoltre ha facilitato il calcolo distribuito, cioè la connessione di molteplici applicazioni per formare un'applicazione più grande, solitamente in una rete.

Una definizione tecnica di middlware è quella di software di connessione che consiste in un insieme di servizi e/o di ambienti di sviluppo di applicazioni distribuite che permettono a più entità (processi, oggetti, ecc.), residenti su uno o più elaboratori, di interagire attraverso una rete di interconnessione a dispetto di differenze nei protocolli di comunicazione, architetture dei sistemi locali, sistemi operativi, ecc.

In particolare il middleware trova applicazione nelle moderne architetture multilivello, ovvero per middleware si intende il software che rende accessibile sul Web risorse hardware o software che prima erano disponibili solo localmente o su reti non Internet.

Il middleware, schematicamente, risulta composto da: *(i)* ambiente di sviluppo applicativo (per l'utilizzo da parte di uno sviluppatore software); *(ii)* servizi di configurazione ed amministrazione del sistema; *(iii)* servizi di supporto, come un directory service, un security service, un time service, servizio di supporto alle transazioni, ecc. (non sempre tutti presenti).

Durante la progettazione di un sistema informativo distribuito, la scelta del middleware è di fondamentale importanza, in quanto l'utilizzo di una particolare tecnologia di middleware influisce sulle scelte architetturali. Può persino accadere che l'adozione di un tipo di middleware renda più o meno facile (e persino possibile) la realizzazione di una soluzione basata su una specifica architettura.

I middleware possono essere classificati come:

- *basati su eventi*: queste tecnologie forniscono operazioni e servizi per gestire la pubblicazione e ricezione di eventi; offrono il dispatcher e i servizi di publish, subscribe, push e pull. Esempi sono JMS – Java Message Service, AMQP - Advanced Message Queuing Protocol, MQTT - MQ Telemetry Transport;
- *basati su RPC - Remote Procedure Call*, in cui il chiamante rimane in attesa della risposta (remota) da parte del servente.

[12] Cf. `https://ironick.typepad.com/ironick/2005/07/update_on_the_o.html`

4.1 RPC ed architetture orientate ai servizi

I middleware basati su RPC sono nel corso del tempo evoluti parallelamente all'evoluzione dei linguaggi ed ambienti di sviluppo, passando da quelli basati su linguaggi di 3a generazione (C - DCE) a quelli object oriented (C++ e Java - CORBA e Java RMI) a quelli orientati al Web (Web service). Al giorno d'oggi, queste tecnologie permettono di sviluppare le cosiddette *interfacce di servizio* e le *architetture orientate ai servizi*.

Una trattazione completa dei paradigmi per la progettazione e realizzazione delle interfacce di servizio esula dagli scopi del presente documento. Il lettore interessato ad approfondire gli aspetti metodologici può riferirsi a [5, 3]. Le interfacce di servizio possono essere progettate secondo due paradigmi:

- RPC-like. In questo paradigma, un'interfaccia di servizio espone una serie di operazioni (metodi) che permettono l'invocazione delle operazioni offerte dall'interfaccia. Il significato dell'operazione è informalmente espresso dal nome dell'operazione, dal numero e tipo dei suoi parametri, dal tipo del valore di ritorno (il tutto viene detto tecnicamente segnatura). Approcci formali prevedono che tale significato venga eventualmente anche descritto in opportuni documenti di accompagnamento e/o attraverso specifiche formali dell'interfaccia di servizio. Ogni operazione può quindi rappresentare sia semplici operazioni di computazione, che operazioni long-running, che operazioni di accesso a dati, ecc. Il paradigma è detto RPC-like in quanto le interfacce di servizio ricordano le librerie di chiamata a procedura (RPC sta per remote procedure call) e quindi l'interfaccia, metaforicamente, è di fatto una libreria di funzioni.
- Resource-oriented. In questo paradigma, l'interfaccia di servizio offre operazioni CRUD di accesso a risorse. CRUD - Create, Read, Update e Delete sono le 4 operazioni fondamentali con cui manipolare risorse. Una risorsa è un qualsiasi oggetto informativo che possiede uno stato. In questo paradigma, l'interfaccia di servizio, metaforicamente, è un accesso diretto ad una base informativa, e le uniche operazioni possibili sono appunto le modifiche a tali risorse.

Parallelamente, esistono delle tecnologie con cui poter naturalmente realizzare interfacce di servizio, che sono *(i)* SOAP ed il cosiddetto stack WS-*, e *(ii)* lo stile architetturale REST basato su HTTP. Entrambe sono indicate come modi per realizzare i cosiddetti Web service (servizi richiamabili sul Web).

Un Web service basato su SOAP espone un insieme di metodi richiamabili da remoto da parte di un client. SOAP definisce una struttura dati per lo scambio di messaggi tra applicazioni, codificata in XML; di fatto SOAP utilizza HTTP come protocollo di trasporto, ma non è limitato nè vincolato ad esso, dal momento che può benissimo usare altri protocolli di trasporto.

Un Web service RESTful adotta il modello basato su risorse secondo le seguenti caratteristiche: individuazione delle risorse mediante il formalismo delle URI; operazioni sulle

risorse effettuate sulla rappresentazione del loro stato; CRUD sulle risorse mediante HTTP utilizzando i metodi nella semantica prevista dal protocollo stesso.

La prima differenza tra i due tipi di Web service è la visione del Web come piattaforma di elaborazione che viene implicitamente proposta: REST propone una visione del Web incentrata sul concetto di risorsa mentre i SOAP Web service mettono in risalto il concetto di servizio. Infatti l'approccio dei SOAP Web service ha mutuato un'architettura applicativa denominata SOA – Service Oriented Architecture, a cui si è contrapposta l'architettura ROA – Resource Oriented Architecture, ispirata ai principi REST. L'approccio adottato dai Web service basati su SOAP è derivato dalle tecnologie di interoperabilità esistenti al di fuori del Web e basate essenzialmente su chiamate di procedura remota, come DCOM, CORBA e RMI. In sostanza questo approccio può essere visto come una sorta di adattamento di queste tecnologie al Web.

L'approccio REST, invece, tende a evidenziare le caratteristiche del Web come piattaforma leggera per l'elaborazione distribuita. Non è, in prima istanza, necessario aggiungere nulla a quanto è già esistente sul Web per consentire ad applicazioni remote di interagire.

I Web Service basati su SOAP prevedono lo standard WSDL, Web Service Description Language, per definire l'interfaccia di un servizio. Questa è un'ulteriore evidenza del tentativo di adattare al Web l'approccio all'interoperabilità basato su chiamate remote; infatti il WSDL non è altro che un IDL (Interface Description Language) per un componente software; l'esistenza di WSDL favorisce l'uso di tool per creare automaticamente client in un determinato linguaggio di programmazione. REST non prevede esplicitamente nessuna modalità per descrivere come interagire con una risorsa. Le operazioni sono implicite nel protocollo HTTP. Qualcosa di analogo a WSDL è WADL, Web Application Definition Language, un formato XML per definire risorse, operazioni ed eccezioni previste da un Web service di tipo REST. WADL è stato sottoposto al W3C per la standardizzazione nel 2009, ma allo stato attuale non ci sono piani per la sua discussione ed eventuale approvazione. In realtà esso non ha avuto un'accoglienza molto favorevole da parte delle comunità di sviluppatori REST.

OpenAPI è un formato di descrizione per Web service REST; un file OpenAPI consente di descrivere l'intera interfaccia (endpoint disponibili e operazioni su ciascun endpoint, parametri di input e output per ogni operazione, metodi di autenticazione, informazioni di contatto, licenza, termini di utilizzo, ecc.). Le specifiche possono essere scritte in YAML o JSON. Swagger è un insieme di strumenti open source che, basandosi sulle specifiche OpenAPI, permette di supportare il progetto, costruzione e documentazione di REST Web service.

4.2 Tecnologie ed architetture ad eventi

I middleware basati su eventi permettono l'architettura event-driven (EDA – Event Drive Architecture) che si basa sulla produzione, l'individuazione, il consumo e la reazione agli eventi.

Un evento può essere definito come "un cambiamento significativo nello stato". Ad esem-

pio, quando un consumatore acquista un'auto, lo stato dell'auto passa da "in vendita" a "venduto". L'architettura del sistema informativo di un concessionario può trattare questo cambiamento di stato come un evento, il cui accadimento può essere reso noto ad altre applicazioni all'interno dell'architettura. Da un punto di vista formale, ciò che viene prodotto, pubblicato, propagato, rilevato o consumato è un messaggio (tipicamente asincrono) chiamato notifica dell'evento, e non l'evento stesso, che è la modifica dello stato che ha attivato l'emissione del messaggio. Gli eventi non viaggiano, si verificano semplicemente. Tuttavia, il termine evento è spesso usato per denotare il messaggio di notifica stesso, il che può portare ad una certa confusione. Ciò è dovuto al fatto che le architetture event-driven sono spesso progettate su architetture message-driven, dove tale modello di comunicazione richiede che uno degli input sia il solo testo, il messaggio, per differenziare il modo in cui ogni comunicazione deve essere gestita.

Spesso nelle EDA si parla di publish & subscribe (talvolta abbreviato in pub/sub), che si riferisce a un design pattern o stile architetturale utilizzato per la comunicazione asincrona. In questo schema, mittenti e destinatari di messaggi dialogano attraverso un tramite, che può essere detto dispatcher o broker. Il mittente di un messaggio (detto publisher) non deve essere consapevole dell'identità dei destinatari (detti subscriber); esso si limita a "pubblicare" (in inglese to publish) il proprio messaggio al dispatcher. I destinatari si rivolgono a loro volta al dispatcher per "abbonandosi" (in inglese to subscribe) alla ricezione di messaggi. Il dispatcher quindi inoltra ogni messaggio inviato da un publisher a tutti i subscriber interessati a quel messaggio. In genere, il meccanismo di sottoscrizione consente ai subscriber di precisare nel modo più specifico possibile a quali messaggi sono interessati. Per esempio, un subscriber potrebbe "abbonarsi" solo alla ricezione di messaggi da determinati publisher, oppure aventi certe caratteristiche. Questo schema implica che ai publisher non sia noto quanti e quali siano i subscriber e viceversa. Questo può contribuire alla scalabilità del sistema.

Un sistema basato sugli eventi è in genere costituito da emettitori di eventi (o agenti), consumatori di eventi (o sink) e canali di eventi. Gli emettitori hanno la responsabilità di rilevare, raccogliere e trasferire eventi. Un emettitore di eventi non conosce i consumatori dell'evento, non sa nemmeno se esiste un consumatore e, nel caso esista, non sa come l'evento viene utilizzato o ulteriormente elaborato. I sink hanno la responsabilità di applicare una reazione non appena viene presentato un evento. La reazione potrebbe o non potrebbe essere completamente fornita dal sink stesso. Ad esempio, il sink potrebbe avere la responsabilità di filtrare, trasformare e inoltrare l'evento a un altro componente o potrebbe fornire una reazione autonoma a tale evento. I canali degli eventi sono canali in cui gli eventi vengono trasmessi dagli emettitori di eventi ai consumatori di eventi. La conoscenza della corretta distribuzione degli eventi è presente esclusivamente all'interno del canale degli eventi.

L'architettura event-driven può integrare l'architettura orientata ai servizi (SOA) perché i servizi possono essere attivati da trigger attivati su eventi in ingresso.

Riferimenti bibliografici

[1] Miguel Ceriani e Massimo Mecella. *D3.1 Report sul Censimento*. Deliverable Progetto EcoDigit. 2019.

[2] Chris Dijkshoorn, Lora Aroyo, Jacco van Ossenbruggen e Guus Schreiber. "Modeling cultural heritage data for online publication." In: *Applied Ontology* 13.4 (2018), pagine 255–271. URL: http://dblp.uni-trier.de/db/journals/ao/ao13.html#DijkshoornAOS18.

[3] Thomas Erl, Benjamin Carlyle, Cesare Pautasso e Raj Balasubramanian. *SOA with REST - Principles, Patterns and Constraints for Building Enterprise Solutions with REST. ISBN 9780137012510*. 2013.

[4] Giorgia Lodi, Luigi Asprino, Andrea Giovanni Nuzzolese, Valentina Presutti, Aldo Gangemi, Diego Reforgiato Recupero, Chiara Veninata e Annarita Orsini. "Semantic Web for Cultural Heritage Valorisation". In: *Data Analytics in Digital Humanities*. A cura di Shalin Hai-Jew. Springer International Publishing, 2017, pagine 3–37. ISBN: 978-3-319-54499-1. DOI: 10.1007/978-3-319-54499-1_1.

[5] Mike P. Papazoglou. *Web Services and SOA: Principles and Technology (2nd edition). ISBN: 9780273732167*. 2013.