ECODIGIT

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

D1.4 Piano di sostenibilità del Middleware

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

D1.4

Work Package:	WP1 T1.4	
Data di Sottomissione:		2 Gennaio 2020
Inizio Progetto:		2 Ottobre 2018
Durata Progetto:		15 Mesi
Reponsabile Deliverable:		Valentina Presutti
Versione:		1.0
Stato:		Finale
Autore:	Luigi Asprino	ISTC-CNR
	Ludovica Marinucci	ISTC-CNR
	Valentina Presutti	ISTC-CNR
	Alessandro Russo	ISTC-CNR
	Miguel Ceriani	RM1
	Massimo Mecella	RM1
Altri contribuenti al lavoro riportato nel deliverable:	Marialuisa Mongelli	ENEA
	Antionio Budano	INFN
	Maria Prezioso	RM2
	Marco Canciani	RM3
	Giovanni Fiorentino	UNITUS

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

Luigi Asprino, Ludovica Marinucci, Valentina Presutti, Alessandro Russo, Miguel Ceriani e Massimo Mecella. *D1.4 Piano di Sostenibilitá del Middleware*. Deliverable Progetto EcoDigit. 2020

Revisioni			
Versione	**Data**	**Modificata da**	**Commento**
0.1	10/1/2020	Luigi Asprino	Prima Versione del documento
0.2	15/1/2020	Luigi Asprino e Alessandro Russo	Azioni di Consolidamento per la base di Conoscenza
0.3	17/1/2020	Valentina Presutti e Ludovica Marinucci	Revisione
0.4	10/2/2020	Miguel Ceriani e Massimo Mecella	Piano di Sostenibilità per il software

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 di Eccellenza 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.

Nel presente documento viene descritto il piano di sostenibilità del Middleware e di tutti gli artefatti sviluppati durante il progetto. In particolare, comprende il piano di sostenibilità discute: le risorse necessarie per il consolidamento, le azioni da eseguire per creare una rete collaborativa aperta per consentire e garantire il contributo di attori interessati anche esterni al progetto, le tecnologie usate e raccomandate e le metodologie di sviluppo. Questo documento è un complemento ai documenti di specifica e architetturale prodotti come risultato del WP2 e raccoglie le esperienze e le raccomandazioni derivanti dal lavoro di progettazione del WP3 e del WP4.

Indice

1 Introduzione

1.1 Obiettivi Work Package

Il Work Package 1 è dedicato alla gestione del progetto e al coordinamento dei partner partecipanti con lo scopo di assicurare sia una comunicazione efficace tra i partner sia una divulgazione adeguata dei risultati. In questo work package è prevista un'attività di analisi dei rischi e di monitoraggio e mitigazione degli stessi. Infine, sarà oggetto di elaborazione in questo work package un piano di sostenibilità di EcoDigit che consideri le risorse necessarie all'evoluzione e manutenzione del sistema dopo la sua messa in opera.

1.2 Obiettivo del deliverable

In questo documento viene descritto il piano di sostenibilità del Middleware DTC e di tutti gli artefatti sviluppati durante il progetto. In particolare, si discuteranno le pratiche e metodologie che garantiranno al Middleware DTC di esistere, evolvere e di consolidarsi nel tempo. Si prenderanno in considerazione strumenti di sviluppo collaborativo open source e si valuteranno quali estensioni ai servizi e ai modelli studiati in fase prototipale siano più desiderabili o necessari in fase di consolidamento. Inoltre verranno discusse le risorse necessarie per il consolidamento, le azioni da eseguire per creare una rete collaborativa aperta per consentire e garantire il contributo di attori interessati anche esterni al progetto (ad esempio con approccio e collegamento alle comunità open source), le tecnologie usate e raccomandate e le metodologie di sviluppo.

1.3 Relazione con le altre attività del progetto

Il lavoro descritto in questo documento influisce beneficia del lavoro e delle scelte fatte in tutte le attività del progetto. Questo documento è un complemento ai documenti di specifica e architetturale prodotti come risultato del WP2 e raccoglie le esperienze e le raccomandazioni derivanti dal lavoro di progettazione del WP3 e del WP4.

1.4 Outline documento

Nella Sezione 2 verrà discusso il piano di sostenibilità per le componenti software. La Sezione 3 discuterà il piano di sostenibilità per la base di conoscenza.

2 Piano di Sostenibilità per il software

Il software di EcoDigit, chiamato collettivamente *Middleware DTC*, si compone di un insieme di componenti che si interfacciano da un lato con la piattaforma Digital Library, da un altro con sorgenti di contenuti oppure utenti del sistema. La progettazione di questi componenti è descritta nel Deliverable D2.2 mentre la loro implementazione ai fini del dimostratore è descritta nel Deliverable D2.3. In questa sezione si discuterà il piano di sostenibilità per questi componenti software, descrivendo prime le strategie adottate e poi le infrastrutture utilizzate per perseguirle.

2.1 Strategie

Questa sezione descrive le strategie utilizzate per la sostenibilità dei componenti software. Anche se nel contesto di progetto l'obiettivo principale del software è "soltanto" quello di dimostrare la fattibilità del sistema, l'adozione di appropriate strategie può favorire la prosecuzione ed estensione della soluzione software realizzata.

2.1.1 Supporto al ciclo di vita del software

Il ciclo di vita del software non si esaurisce al suo rilascio, poiché qualunque software che sia utilizzato deve essere adeguato alle esigenze (necessariamente mutevoli) degli utenti, alla necessità di risolvere i problemi e colli di bottiglia che si riscontrino, al bisogno di integrare dati di fonte, tipologia o formato diversi. Si rende necessario adottare degli strumenti metodologici e tecnologici adatti a gestire questo processo in forma potenzialmente continuativa.

Innanzitutto i componenti software devono utilizzare sistemi di **versionamento** in modo da tenere traccia dell'evoluzione del software in forma affidabile. In particolare si adotterà il **versionamento semantico** che è uno standard che prevede la rappresentazione delle versioni in forma numerica e dando un indicazione implicita del tipo di modifiche effettuate.

Sia nel corso del progetto EcoDigit che dopo il progetto, lo sviluppo si svolge per lo più in forma collaborativa. E' quindi necessario che il codice relativo ad ogni componente si trovi su un **repository online** accessibile a tutte le persone interessate a collaborare. Ed è necessario che il repository abbia un sistema di gestione e tracciatura delle modifiche apportate (commit) in modo da poter gestire eventuali conflitti, tornare a versioni precedenti de serve, o gestire lo sviluppo su diverse versioni in parallelo (branch). Esistono diversi software sviluppati per questo, il più diffuso e supportato al momento è git.

Un'altra cosa fondamentale per supportare l'evoluzione del software è quella di avere un sistema di **tracciamento di errori e richieste di modifiche** che possa essere utilizzato dagli utenti direttamente o indirettamente e che chi sviluppa può utilizzare per controllare quali modifiche devono essere effettuate, quali problemi sono stati risolti e quali no.

Nel caso di dover organizzare il lavoro di un gruppo di sviluppo è consigliabile adottare una metodologia di sviluppo (ad esempio, Scrum) che permetta la gestione trasparente ed efficace dei processi. Ma poiché al momento non vi è un gruppo di sviluppo definito post-progetto, su questo punto è preferibile non fornire vincoli dato che l'organizzazione e l'eventuale adozione di una specifica metodologia dovrà dipendere dai partecipanti e dal contesto concreto.

2.1.2 Favorire il contributo allo sviluppo

Per favorire lo sviluppo collaborativo del software e non avendo vincoli commerciali, il codice può essere distribuito con una **licenza di software libero**. Oltre a permetterne l'uso senza vincoli, il fatto di avere una licenza libera favorisce la collaborazione nello sviluppo da parte di terzi.

Al fine di facilitare il mantenimento del software da parte di una comunità di sviluppatori, è anche opportuno che il codice sia opportunamente documentato e segua un'organizzazione razionale, possibilmente corrispondente agli standard diffusi. In particolare, per quanto riguarda l'organizzazione, ogni repository dovrà avere un **file README** con le indicazioni per compilare ad eseguire il codice.

Per quanto riguarda l'organizzazione del codice, dovrà essere utilizzato un **sistema di gestione di pacchetti** (ad es. npm per Node.js o composer per PHP) per automatizzare l'acquisizione delle librerie necessarie e un **build tool** (ad es. gulp per Node.js, Phing per PHP o Make) per automatizzare il più possibile il processo di build.

2.1.3 Gestione modulare del software

Per favorire il mantenimento, il riuso ed il deployment del software, il codice sarà **organizzato in moduli** il più possibile indipendenti. Quindi si utilizzeranno i meccanismi dei diversi linguaggi di programmazione per produrre moduli che poi, attraverso un approccio bottom-up, possano poi venire usati in altri moduli attraverso un sistema di gestione di pacchetti.

Al tempo stesso, l'architettura generale del sistema seguirà un'organizzazione a **micro-servizi**, in cui blocchi di funzionalità vengono gestiti da sistemi indipendenti. I sistemi di virtualizzazione a livello di sistema operativo permettono di realizzare questo tipo di soluzione con un over-head ridotto rispetto ai sistemi di virtualizzazione completa (a livello di processore). L'applicazione viene quindi costruita collegando fra loro, attraverso protocolli di rete (principalmente HTTP) diversi micro-servizi, ciascuno eseguito in un container (i moduli eseguibili o in esecuzione del sistema di virtualizzazione). I canali di rete utilizzati per la comunicazione tra container possono essere virtuali e quindi invisibili all'esterno.

2.1.4 Sfruttare le infrastrutture già accessibili

Per facilitare la messa in servizio e ridurre i costi di manutenzione in questa fase, si utilizzeranno le infrastrutture hardware/software che sono già accessibili. Quindi innanzitutto le **infrastrutture dei partner di progetto** che si possano utilizzare senza costi aggiuntivi. Questa sarà l'opzione scelta per il deployment del prototipo e potenzialmente la sua progressiva estensione.

Esistono inoltre diverse **infrastrutture online disponibili liberamente** che possono essere utilizzate per abbattere i costi e garantire la persistenza dei servizi, codice e risultati del progetto. Il limite dei servizi gratuiti e che spesso le risorse di sistema disponibili sono molto limitate, in termini di tempo di processore, RAM, risorse per la persistenza dei dati, banda, uptime. Per questo, come anticipato sopra, per la messa di servizio si preferisce l'uso di infrastrutture interne dei partner. Le infrastrutture pubbliche libere sono invece utilizzate per tutti i tool di sviluppo in cui qualche limite alla performance con rappresenta una criticità, come la gestione dei repository con il codice ed il sistema di tracciamento di errori.

2.1.5 Stimolare i fornitori di contenuto

Al di là delle scelte tecniche che supportino la manutenzione ed evoluzione del software, sarà in ogni caso necessario trovare risorse umane ed economiche per consentire una minima continuità nella gestione e sviluppo della piattaforma. Una strategia può essere quella di pubblicizzare la piattaforma ai fornitori di contenuto (tra cui vi sono molti enti pubblici presenti nell'area della Regione Lazio) in modo che essi vedano la possibilità di facilitare l'accesso e valorizzare i propri contenuti attraverso la piattaforma. L'obiettivo è di creare un circolo virtuoso in cui la disponibilità di maggiore quantità, diversità e qualità di contenuti può migliorare e produrre più interesse per la piattaforma e viceversa.

2.2 Infrastrutture di supporto

In questa sezione si descrivono sinteticamente le infrastrutture software/hardware scelte per supportare le strategie descritte sopra. Questa lista non è da considerarsi esaustiva e definitiva per quanto riguarda gli sviluppi futuri, in quanto le tecnologie cambieranno nel tempo e può darsi che nuove tecnologie più adatte ai fini della piattaforma si vadano ad aggiungere o sostituire a quelle al momento utilizzate.

2.2.1 GitHub

GitHub[1] è una piattaforma online che offre gratuitamente l'hosting di repository di codice aperto, utilizzando il software di versioning git. La piattaforma oltre a facilitare la gestione dei

[1] `https://github.com`

repository, associati a singoli utenti o a organizzazioni, contiene un sistema di tracciamento degli errori e problemi rilevati e delle richieste di modifica. Facilita inoltre la collaborazione nello sviluppo anche da parte di chi non fa parte del gruppo di sviluppo principale. Chiunque può proporre un miglioramento del codice attraverso una *pull request* che il gruppo di sviluppo può decidere di accettare e quindi integrare in maniera automatica (se non ci sono conflitti) nel branch principale o in un branch di sviluppo.

Per lo sviluppo del codice nel progetto è stata creata un'organizzazione su GitHub[2] in cui si trovano tutti i repository di progetto.

2.2.2 Docker

Docker[3] è una piattaforma, composta da diversi moduli software, per la virtualizzazione a livello di sistema operativo. Permette di predisporre dei moduli statici, dette *immagini*, che contengo lo stack software necessario ad un'applicazione, a partire dal sistema operativo fino ad arrivare all'applicazione stessa con eventuali dati di cui abbia bisogno all'avvio. Il *Docker daemon*, ovvero il software che gestisce la virtualizzazione, è in grado di caricare queste immagini sotto forma di *container*, che oltre alle immagini contengono l'ambiente di esecuzione dinamico[4]. I container possono essere mandati in esecuzione o fermati a piacimento. Comunicano tra di loro ed all'esterno attraverso la mappatura delle porte di rete interne ai container con quelle esterne dell'host oppure su reti virtuali interne.

Docker, Inc, la società che mantiene la piattaforma Docker (che dal punto di vista software è libero), ha creato anche un repository publico per immagini, *Docker Hub*. Questo facilita la ricerca di immagini e permette un alto livello di automazione del processo di composizione di micro-servizi.

Le principali componenti software che si sviluppano nell'ambito del progetto vengono "dockerizzate", ovvero contengono un *dockerfile*, file di configurazione che permette di creare dal repository la corrispondente immagine del micro-servizio. Quest'immagine può poi eseguita su qualunque host su cui giri il Docker daemon, senza dover gestire altre dipendenze.

2.2.3 Server CNR

Il CNR ha messo a disposizione un proprio server per l'hosting del prototipo. Il prototipo si compone di un insieme di micro-servizi che dialogano tra di loro ed espongono (per quanto riguarda i servizi di front-end) interfacce web verso l'esterno. I servizi di front-end vengono mappati sulla porta 80, su sotto-percorsi diversi. In questo modo compongono collettivamente il front-end del prototipo.

[2] `https://github.com/ecodigit`
[3] `https://www.docker.com/`
[4] Il rapporto tra immagine e container può essere visto in analogia al rapporto tra applicazione e processo.

3 Piano di Sostenibilità per la base di conoscenza

In questa sezione verrà discusso il piano di sostenibilità per la base di conoscenza. La discussione verrà articolata secondo i due livelli che compongono la base di conoscenza, cioè: il *livello intensionale*, cioè schema concettuale della base di conoscenza formalizzato nelle ontologie presentate nel Deliverable D3.2 [1]; e il *livello estensionale*, cioè i dati raccolti nella realizzazione della Proof-of-Concept del Task 3.3 (cf. Deliverable D3.4 [3]) tramite gli strumenti presentati nei Deliverable D3.3 [2].

3.1 Piano di Sostenibilità per il Livello Intensionale della Base di Conoscenza

In questa sezione elencheremo i principi metodologici che favoriscono la sostenibilità dei modelli concettuali nel tempo. Alcuni di questi principi sono stati già adottati sin da inizio progetto, altri se ne consiglia l'adozione nelle fasi successive del Centro di Eccellenza.

Sono diversi i problemi da affrontare per mantenere gli schemi concettuali (ontologie) nel tempo. Alcuni di questi sono di natura prettamente tecnica (e.g. garantire la persistenza dell'URI dell'ontologia) altri di natura più "sociale" (come ad esempio coinvolgere persone nel processo di elicitazione dei requisiti per l'ontologia). In questa sezione verrà effettuata una panoramica sui principali problemi e sulle relative strategie di mitigazione da adottare.

Garantire uno sviluppo sostenibile delle ontologie. Uno dei problemi principali nel mantenimento delle ontologie nel tempo è il dover modificare ontologie esistenti rispetto a nuovi requisiti che emergono da nuove sorgenti dati. Così come nello sviluppo software, le strategie che tendono a creare dei monoliti (in questo caso ontologie monolitiche) risultano essere poco mantenibili e poco prone al cambiamento. Al fine di superare questo problema sono state proposte diverse strategie di modularizzazione delle ontologie. Tra le più efficaci di queste troviamo eXtreme Design (XD) [4] che è una metodologia "agile" per il design delle ontologie che fa sì che la formalizzazione di una certa ontologia evolva nel tempo così come evolvono i requisiti provenienti dalle sorgenti dati. Inoltre, questa metodologia, adottando l'uso di pattern ontologici modulari, favorisce sia la stabilità dell'ontologia stessa (essendo il pattern ontologico associato a dei requisiti molto specifici tende a non cambiare nel tempo) e favorisce il riuso di altre pattern ontologici alleggerendo così il carico di modellazione sui designer delle ontologie. In EcoDigit è stata adottata eXtreme Design sin dall'inizio del processo di sviluppo delle ontologie.

Garantire la stabilità delle ontologie. Il modello di integrazione sviluppato da EcoDigit prevede che le sorgenti dati che vogliono entrare nel sistema devono conformarsi al modello di ingresso definito nel Deliverable D3.2 [1]. Questo richiede che il modello di ingresso proposto alle sorgenti sia stabile. Quindi, risulta necessario minimizzare il più possibile i

cambiamenti (soprattutto semantici) nel tempo delle ontologie definite. Tre sono le tecniche adottate per raggiungere questo obiettivo. La prima prevede l'uso di ontologie stabili e di larga diffusione per il dominio (e.g. FOAF per i dati anagrafici della persona, Dublin Core per la metadazione ecc.). Queste ontologie sono consolidate nel tempo e quindi è poco probabile che avvenga un cambiamento. Quando allo stato dell'arte non sono presenti ontologie stabili e di riferimento che possono soddisfare i requisiti di modellazione si consiglia di seguire un riuso indiretto delle ontologie esistenti (vedi [5]), nel caso in cui si possa riusare parzialmente la soluzione definita in una ontologia esistente. Se non esistono soluzioni esistenti si consiglia di seguire la metodologia eXtreme Design e definire nuovi pattern ontologici.

Accessibilità e Interoperabilità delle ontologie. L'accessibilità e interoperabilità delle ontologie viene garantita definendo una URI persistente per l'ontologia e usando linguaggi di metadatazione standard. Per le ontologie definite nel contesto del progetto è stato creato l'URI permanente `https://w3id.org/ecodigit/` e sono stati usati i linguaggi XML per la serializzazione dell'ontologia, RDF come data model della formalizzazione dell'ontologia e OWL come linguaggio per definire la semantica dei costrutti dell'ontologia.

Favorire l'arricchimento, la diffusione e l'adozione delle ontologie. Per rendere sostenibile l'evoluzione di un modello è necessario favorire la scalabilità dell'evoluzione delle ontologie, cioè, garantire che la definizione di nuovi requisiti si tramuti repentinamente in una evoluzione dell'ontologia. Questo obiettivo può essere raggiunto creando una community di adopter intorno all'ontologia. In progetti simili come ArCo[5] e OntoPiA[6] la community che si è creata intorno alle ontologie si occupa di definire nuovi requisiti e di proporre modifiche alle ontologie esistenti al fine di catturare i nuovi requisiti o corregere eventuali errori. In un proseguimento del progetto si ritiene indispensabile l'istituzione di una community di adopter delle ontologie che si occupino di mantenere e raffinare le ontologie.

Costi e Pianificazione temporale. Fatta esclusione per l'istituzione della community di early adopter, tutte le pratiche e metodologie discusse in questa sezione sono state già avviate durante il progetto. Per il mantenimento di quanto fatto si ritiene necessario l'istituzione di un team (di almeno due esperti) per lo sviluppo e il mantenimento delle ontologie. Per quanto riguarda l'istituzione della community di adopter è stata parzialmente attivata durante il progetto coinvolgendo altri team di sviluppatori esterni (cioè, i team di RM1 e CNR-RSI del progetto Anagrafe). Questi team hanno contribuito all'evoluzione dell'ontologia definendo nuovi requisiti e valutando le ontologie sviluppate nella modellazione dei propri dati. In una fase successiva del progetto il programma di adoption andrà rafforzato con iniziative come hackathon o datathon. Per quanto riguarda l'attività di moderazione della community si ritiene che questa possa essere affidata al team di mantienimento e sviluppo delle ontologie.

[5] `https://github.com/ICCD-MiBACT/ArCo`
[6] `https://github.com/italia/daf-ontologie-vocabolari-controllati`

Mentre per l'attività di organizzazione di iniziative come hackathon o datathon bisognerà incaricare una persona e stanziare dei fondi per l'organizzazione di eventi e per metter in palio premi per incentivare la partecipazione di attori esterni.

3.2 Piano di sostenibilità per il Livello Estensionale della Base di Conoscenza

Per assicurare la sostenibilità del livello estensionale della base di conoscenza bisogna garantire il raffinamento e l'arricchimento dei contenuti della base di conoscenza, il mantenimento e lo sviluppo degli strumenti che permettono l'integrazione (cf. Deliverable D3.3 [2]) e la continuità di esercizio dei servizi software pubblicati sul web.

Garantire il raffinamento e l'arricchimento dei contenuti della base di conoscenza. Tra le strategie più comuni per garantire il raffinamento dei contenuti già presenti nella base di conoscenza troviamo il crowdsourcing e le iniziative che favoriscono i riuso dei dati. Le tecniche di crowdsourcing mirano a coinvolgere persone comuni nell'attività di pulizia e arricchimento dei dati. Queste tecniche incentivano o ricompensano le persone che partecipano a questa attività con dei crediti in denaro o con altro tipi di benefici che gli utenti potrebbero guadagnare. Leggermente diverse sono le tecniche di crowsourcing chiamate game-with-a-purpose che trasformano l'attività di pulizia e arricchimento dei dati in un gioco. Così facendo gli utenti che partecipano al gioco, contribuiscono indirettamento al raffinamento della base di conoscenza. Una seconda possibilità per il raffinamento dei contenuti della base di conoscenza è quella istituire delle iniziative che favoriscano il riuso dei dati. Infatti, il riuso dei dati fa emergere eventuali problemi o errori nei contenuti. Queste iniziative potrebbero essere incentivate da premi messi in palio durante eventi come hackathon. Per arricchire la base di conoscenza con nuovi contenuti bisognerà continuare l'attività di survey. Questa attività già iniziata durante il progetto con il Task 3.1 dovrà proseguire durante una successiva fase del centro di eccellenza. Le nuove fonti identificate, oltre a ricevere un supporto per la trasformazione dei propri contenuti per renderli conformi al modello di ingresso, dovranno ricevere un incentivo, ad esempio in termini di visibilità dei dati conferiti.

Garantire il mantenimento e lo sviluppo degli strumenti di trasformazione dei dati. Alcuni strumenti scelti per la trasformazione dei dati delle sorgenti sono stati sviluppati in seno a comunità open source consolidate (ad esempio le comunità Apache). Avendo fatto questa scelta si beneficia del mantenimento del software effettuato dalla comunità. Per gli strumenti che invece sono stati sviluppati ex-novo nel progetto bisognerà creare una comunità a supporto dello sviluppo software. La scelta di aver adottato una politica di open source si da inizio progetto incentiva la creazione di una comunità intorno al software sviluppato. Infatti, i partecipanti alla comunità potrebbero sia beneficiare che contribuire allo sviluppo software.

Garantire la continuità di esercizio dei servizi software pubblicati via web. I dati raccolti ed integrati (così come tutti i servizi software sviluppati) durante il progetto sono attualmente ospitati su una macchina virtuale del Data Center del CNR. Per garantire che i dati restino accessibili nel tempo bisognerà prevedere le risorse per il mantenimento dei server.

Riferimenti bibliografici

[1] Luigi Asprino, Ludovica Marinucci, Andrea Giovanni Nuzzolese e Valentina Presutti. *D3.2 Modello di ingresso (v.2)*. Deliverable Progetto EcoDigit. 2019.

[2] Luigi Asprino, Ludovica Marinucci, Andrea Giovanni Nuzzolese e Valentina Presutti. *D3.3 Studio sugli strumenti di supporto*. Deliverable Progetto EcoDigit. 2019.

[3] Luigi Asprino, Ludovica Marinucci, Andrea Giovanni Nuzzolese, Valentina Presutti, Massimo Mecella e Miguel Ceriani. *D3.4 Proof-of-Concept*. Deliverable Progetto EcoDigit. 2019.

[4] Eva Blomqvist, Valentina Presutti, Enrico Daga e Aldo Gangemi. "Experimenting with eXtreme design". In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer. 2010, pagine 120–134.

[5] Valentina Presutti, Giorgia Lodi, Andrea Giovanni Nuzzolese, Aldo Gangemi, Silvio Peroni e Luigi Asprino. "The Role of Ontology Design Patterns in Linked Data Projects". In: *Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings*. A cura di Isabelle Comyn-Wattiau, Katsumi Tanaka, Il-Yeol Song, Shuichiro Yamamoto e Motoshi Saeki. Volume 9974. Lecture Notes in Computer Science. 2016, pagine 113–121. ISBN: 978-3-319-46396-4. DOI: 10.1007/978-3-319-46397-1_9. URL: https://doi.org/10.1007/978-3-319-46397-1%5C_9.