On Designing a Content Management System for the Documents Related to Past Civil Engineering Projects or Call-for-Tender Responses

Christian Esposito[1][a] and Oscar Tamburis[2][b]
[1]Department of Electrical Engineering and Information Technology (DIETI), University of Naples "Federico II", via Claudio 21, 80125 Napoli, Italy
[2]Department of Veterinary Medicine and Animal Production (DMVPA), University of Naples "Federico II", Via Federico Delpino 1, 80137 Napoli, Italy

Keywords: Content Management System, Data Storage & Retrieval, Civil Engineering Industrial Practice, Access Control.

Abstract: The progressive dematerialization of paper-based documents in favour of digital ones held within servers, and processed/exchanged by means of the ICT technologies have massively revolutionized several aspects of our daily lives and industrial practice. The large volume of data handled by the companies has an undeniable value for their business and must be exploited so as to strengthen their competitiveness. In certain domains, such as the healthcare or manufacturing, this is nowadays an accepted practice within the context of Big data Analytics, but in other domains this is not the reality. As an example, we can see the case of the civil engineering companies, where a large volume of digital documents on past projects or response to public/private tender procedures are stored, but not efficiently used. The main obstacle for the effective use of such data is related to their storage approach, which is mainly considered as an archive rather than the company's knowledge to be inferred and effectively used in the business. The driving idea of this paper is to devise a content management system within the context of civil engineering to pave the way for a better use of the data held by such a kind of companies.

1 INTRODUCTION

The current industrial practice encompasses computer-based communications and the production, processing or exchange of digital documents. This is the foundation of our digital society and the current fourth industrial revolution, where ICTs and computers play the main role in our daily activity and business. Such a revolution reached also the civil engineering, where at the beginning we have witnessed the proliferation of Computer-Aided Drafting (CAD) tools (Luzadder, 1992) able to substitute manual with electronic drafting when designing building, maintenance system or interior design. The next step has been represented by the arrival of the Building Information Modeling, or BIM (Eastman et al., 2018), which targets the creation and management of digital representations of artefacts and support the plan, design, construction, and maintenance of diverse physical infrastructures. By its definition, such a standard provides architects and civil engineers with a shared knowledge resource for information about any kind of facilities, thus forming a reliable basis for decisions during its life-cycle. At the moment, another revolution in Europe is undergoing with the EU Directive 2014/24/EU on public procurement (EUR-Lex, 2014), and the relative regulation issued by the State Members, as of the adoption of digital representations in open formats, like BIM, for the design of civil infrastructure and the submission of responses to call-for-tenders by means of certified emails. This is considerably boosting the dematerialization of all civil engineering documents for public procurement and augmenting the volume of digital data held by such a kind of companies.

When a call-for-tender is issued by a public authority, the time left for writing a response is extremely limited. Starting by scratch likely means starting to be ineffective, when it must instead be viable to leverage on the past experience so that to quickly draft the response, and to pay all the available efforts to refine such a draft. Currently, despite the past documents are stored by the companies, all

[a] https://orcid.org/0000-0002-0085-0748
[b] https://orcid.org/0000-0002-0130-7915

Esposito, C. and Tamburis, O.
On Designing a Content Management System for the Documents Related to Past Civil Engineering Projects or Call-for-Tender Responses.
DOI: 10.5220/0007750906500657
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 650-657
ISBN: 978-989-758-372-8

the lesson learned from past projects and responses to call are not inferred from these documents, but by interacting with the senior employees of the company. Leaving the company knowledge within the senior employees is not a winning choice, as they may be unavailable due to possible diseases, or have left the company for retirement or new job offers. The best solution should be to take advantage of the digital documents within the company archives and infers the lessons learnt from them. However, currently this is not possible as many storage approaches have not been designed for retrieval but only for conversation and historical reasons. Moreover, such documents are not kept at a main server of the company or within the cloud, but at the terminal of each employee so that their existence is not well known by the rest of the company and their accessibility is not guaranteed.

The driving idea of this work is to design and implement a proper data management solution so that all the documents produced during the company lifecycle are centrally stored and accessible by all the employees of a civil engineering company. To this aim we have implemented a series of RESTful Web Services (Richardson and Ruby, 2007), so that the explicit use of the HTTP methods can be used to implement the ”create, read, update, and delete” (CRUD) operations for the digital documents for past civil engineering projects or tender responses. We have found, as described in Section 2 that the best strategy for the storage of such a kind of data set is a Content Management System (CMS) as Alfresco, on top of which we have built our system, whose design and implementation is detailed in Section 3. Such a work represents only a starting point of our research, and it does not have the ambition to resolve all the challenging imposed by our vision. Section 4 describes our vision on the future research efforts needed to be spent in order to cover all the demands of the modern civil engineering companies. Section 5 concludes our paper with the sum up on our research conducted so far.

2 BACKGROUND AND RELATED WORK

In their practice to respond to call-for-tender issues by public administrations, employees at civil engineering companies needs to collect a precise set of documents describing the call of interest and to produce another set of documents to respond to the tender itself. The first set describes the demands of the tender issuer, and the second one is required to participate in the selection to win the tender. The time needed to write the documents in the second set is very limited, and the related efforts can be considerable as many details need to be considered so as to set up a successful response and design. Such documents can belong to one of the following groups. The first documents contain the administrative description of the company, its technical and managerial characteristics, and a statement of satisfaction of the requirements in the call. Mainly, the content of these documents is always invariable despite of a specific call, so the effort to write them are very negligible by having similar documents produced for past calls. The second group contains the technical response to the call with the design of a building, a maintenance plan or the facility management solution. The last group of documents are related to the economic-financial offer, and the scheduling of the needed work to implement the envisioned design. Such documents are particularly demanding in terms of knowledge to collect, time for their finalization or efforts to check their correctness. Starting by scratch to approach them is not viable and it should be possible to re-use past similar documents, according to a similar approach put in place for the re-use within the context of software engineering (Land et al., 2009). Not all the documents share the same format and structure, but the ones within the first and third groups are mainly produced by using Microsoft Office (Word is used to write DOCX documents for the administrative part, and Excel used for datasheets containing the economic offer). Such files can be archived as they are or in PDF or PNG after being printed, signed and scanned. The documents for the technical part of the tender response are made of digital representations of a given design produced by a CAD tool, which can have a binary encoding (each tool adopts a given format), a structured format derived from XML and so on. To this end, a concrete example is provided by the DXF and DWG formats (Çetiner, 2010) of files saved with AUTOCAD or the Initial Graphics Exchange Specification (IGES) (Liewald, 1985) for the interoperability among the available CAD tools. Image formats as PNG or JPG or a vector graphics format such a HPGL Plotter File (PLT) (Grabowski, 1999) with the rendering of a given design can be used as well. The BIM standard has also its own format to represent the digital artefacts of a design, called Industry Foundation Classes (IFC) (Venugopal et al., 2012).

Traditionally, such an heterogeneous set of files and digital documents are stored in folders within the file systems of their terminals and/or servers, with a flat organization where each folder contains all the documentation for a given project/tender response, or a deep organization where the folders have a hierarchical structure (for example a first level divided in years, a second one in classes of project/tender responses, and a third one distinguishing among won and lost applications). This is a naive solution whose main advantage is to be very simple to define, even by operators within a computer science major. The main drawback, on the other side, is related to the difficulties when a specific document needs to be retrieved, as it demands to traverse the overall structure, open each file and check if its content matches the terms of interest. Moreover, all the data of interest for a given project or tender response initiative are not only contained in the produced documents, but also in a set of contextual information and meta-data within the employees' notes and discussions. These meta-data cannot be stored by using such a solution, but on the contrary they can be easily kept within a relational database. Such a solution is associated with a well-structured and expressing query language that helps to retrieve the data of interest by using a precise query expression, which requires operators having a solid background in computer science. Within the tables and tuples, the meta-data can find place, but relational database are not suitable when dealing with files, unless each tuple contains a pointer (such as the file absolute path within the file system) to the file related to the meta-data contained in such a tuple. This requires the user to be aware to keep such a pointer consistent in case of file movements. A Content Management System, on the contrary, represents a hybrid solution among the previous ones: as the first approach, files are contained within hierarchy of folders managed by the file system; while as the second one a relational database with meta-data per each file is defined. The CMS products keeps the two sides of the systems consistent so that by querying the meta-data it becomes possible to retrieve the files of interest. In our envisioned system, we have used a CMS product named Alfresco (Shariff et al., 2009), and on top of it we have designed and implemented our solution.

3 PROPOSED SOLUTION

Figure 1 depicts the overall architecture of the envisioned system, where we have the file system with the documents of the company, and the database holding the relative meta-data. The Alfresco solution is provided as a war file executed by the web container represented by Apache Tomcat. Such a product provides a management and user console represented by the share.war, so that all the functionalities of the product can be directly invoked by authorized administrators and users, while we have the DocMan.war implementing all the operations of our system (such as the CRUD operations for the kind of documents managed by the system), exposing a RESTful web services implemented within the .NET framework, and providing a set of web pages as GUI for the web clients. More details on key aspects of our system are provided in the following subsections.

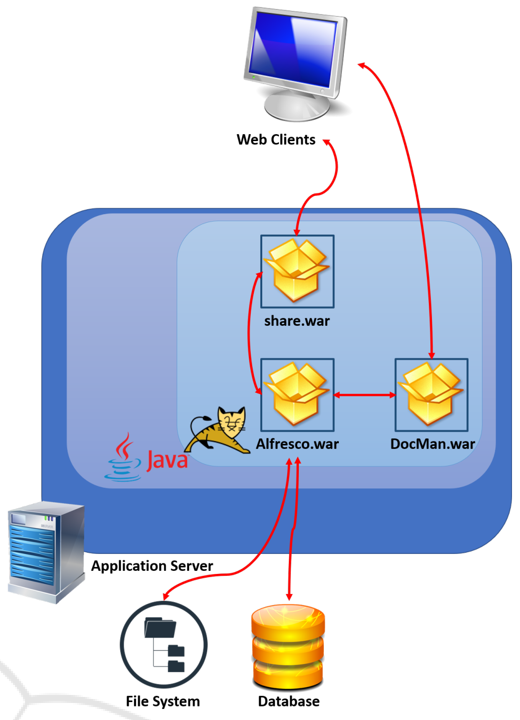


Figure 1: High-level architecture of the proposed system.

3.1 Alfresco Data Modelling

The user is not able to directly manipulate files and tuples in Alfresco, respectively within the repository in the file system or the tables of the relational database, but these aspects are abstracted by proper modelling the data of interest in types. More specifically, the data managed by Alfresco are represented in terms of nodes and links among them. A node can be anything, such as an image, a fragment of XML, and so on. A node can contain a set of other nodes, and in this case, we represent a folder. Each node exhibits a unique identifier and a set of properties, each characterized by a name and a value. The containment is not the only relationship among nodes: the other ones are the specification (*i.e.*, a child node inherits all the property definitions of the father node in addition to its specific ones) and the association (*i.e.*, a node is semantically related to another one and holds properties with the identifier or the associated node, similarly to

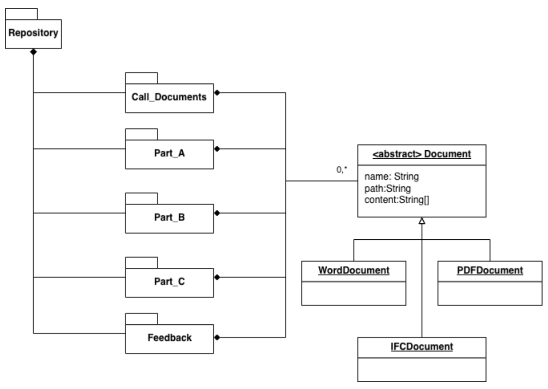


Figure 2: Repository structure.

a foreign key among tuples in a relational database). The property definition is done by assigning a type to a node, in a similar way of classes on the Object-Oriented programming languages. Alfresco provides a set of predefined node types, but the user is able to customize them by defined new types as well.

In our system, we have identified two types of nodes: the documents describing the call-for-tender and their annexes, and the documents related to the tender response and/or project. To this aim we have structured the repository in five folders, as depicted in Figure 2:

- The folder holding the documents describing the calls;
- The first part of the response documents with the administrative details of the company;
- The second part of the response documents with the technical design of the project;
- The third part of the response documents with the economic and financial offer;
- All the feedback received for the submitted responses requested by the company.

Within each folder, a series of documents can be stored, each exhibiting a different type such as Microsoft office documents, PDF or IFC, all modelled as a Document with a unique identifier, a path (*i.e.*, the pointer of the file within the file system), and its byte content. We have envisioned two types in Alfresco: one for the documents of the calls, and one for the documents of the responses. The first one has the following properties:

- CIG – represents the identification code of the call as a string, and allows to trace the call within the Italian public administration.
- CUG – represents the identification code of the public procurement as a string.
- Identifier – specifies an internal identification code for the company as a string.
- Contracting Entity – specifies which entity of the public administration has issued the call-for-tender.
- Text – contains a brief description of the call.
- Type – is an enumeration of the specific type of the call, such as building a new facility, maintain an existing one and so on.
- Publication Year.
- Expiration Date.
- Place – indicates where (in Italy) the project should take place.
- Inspection Needed – is a flag whose true value indicates that an inspection of the work area is needed before submitting the response.
- State – is the state of the call and is an enumeration with elements such as Open, Close or Forthcoming.

The second type has the following properties:

- Identifier – represents a local identifier of the document within the company.
- Object – is a string summarizing the content of the document.
- Support – characterizes the type of document as an enumeration containing DOC, PDF, IFC, Image, Binary, and so on so that all the possible formats of the files are represented.
- Class – is another enumeration for the specific role of the document within the response and project, such as Administrative, Technical, Financial, Call Description and Feedback.

These two types are designed based on the Document type offered by Alfresco, and share with it the two properties of Creator, indicating the user that has created the document in Alfresco, and Creation Date. Moreover, some documents may be related to a response that has been evaluated and for which we know whether the response has been successful or not. In this case, we have modelled such additional properties of an assessed document as an aspect, which is a type with properties to enhance existing content types. Our aspect is named Assessed and features the following properties:

- Assessor - represents the identifier of the person who made the evaluation or the entity in the public administration that has made it.
- Assessment - represents the received mark as a number or a letter.
- Positive Aspects - indicates what has been evaluated as positive within the document.

- Negative Aspects - indicates what has been evaluated as negative within the document.
- Distance from the Winner - represents how far the received rank is with respect to the one obtained by the winning response (this is 0 if the specific response was the successful one).
- Won - a Boolean value indicating if this document belongs to a response that has obtained the tender.

3.2 RESTful API of DocMan.war and its Interconnection with Alfresco

The API we offer contains a series of functionalities to create, modify and infer files and their meta-data according to the previous modelling:

- Loading new files, within the repository and setting their properties based on the above model.
- Change the values given to the properties for a document that exists within the repository.
- Removal of one or more documents from the repository, uniquely identified by means of an id or belonging to a specific call-for-tender.
- Search for one or more documents. The search methods can be of two types:
  - Free search, where a search string is passed to the engine, so as to give the users the freedom to express their own search criteria, but requiring them to have an ICT background. Therefore, this operation will be offered only to administrators or IT technicians in service at the company.
  - Pre-coded searches, meaning a series of methods for which the search strings are already coded within the engine. The users can easily recall them by selecting the value of the search parameters. Examples can be:
    - Return documents of a specific type (administrative, technical or other) for a given set of calls;
    - Return the winning responses in a given time period, that is an adjacent set of years;
    - Return the winning responses for a certain type (design of buildings, maintenance or other);
    - Return documents for which the object meets a certain regular expression if applied to its content;
    - Return call-for-tenders for one or more specific contracting authorities.

Each of these functionalities is offered by means of an HTTP method, such as the POST method (used to create a new instance of the Call type or the Document type), and the GET method (used to obtain a node with a given identifier), and so on.

The interaction between Alfresco and our software can occur by means of Alfresco's API, but we have preferred to use the standard interface offered by the Content Management Interoperability Services (CMIS) (OASIS, 2015), which is a standard for the interoperability with CMS, so that if we decide in the future to substitute Alfresco with another similar product we do not have to make any changes to our system. CMIS defines in fact an abstraction layer for controlling diverse document management systems and repositories using web protocols, such as REST, AtomPub and so on, which any product like Alfresco implements. CMIS provides a query language whose syntax is based on SQL-92, with its clauses such as SELECT, FROM, JOIN and so on. When the user needs to express a query string for the free search, they must be compliant to the CMIS Query Language. For example, to have all the documents in the folder Part_A the command should be: $SELECT * FROMcmis : folder\ WHERE\ IN\_FOLDER('Part\_A')$. Within our work, we have used a library so as to interact with Alfresco by means of CMIS, called Apache Chemistry (specifically its distribution for .NET named DotCMIS), where the binding endpoint for the Alfresco repository was http://localhost:8080/open/service/CMIS.

3.3 Security Aspects

The interactions of the user with the RESTful web services of our solution needs to be protected. One level of protection is to implement an authentication and authorization mechanism so that only legitimate users can invoke the operations of our system. Within the context of the web services, the WS-Security standard (Atkinson, B. et al., 2002) indicates the mechanisms to be used in such a case. Specifically, we have preferred to use JSON Web Token (JWT) (Jones et al., 2015) given its simplicity and stateless nature. The users provide its credential as username and password to the server, which checks if they are present in its identity storage and returns a token in case of success. The adopted model is, therefore, the Access Control List (ACL) (Barkley, 1997), where we can find which users are granted access to objects/functionalities of the system. It is possible to have a more complex model, even by integrating JWT within the context of the eXtensible Access Control Markup Language (XACML) (OASIS, 2017) and its architecture with the standardized authorization flow. Later on, when

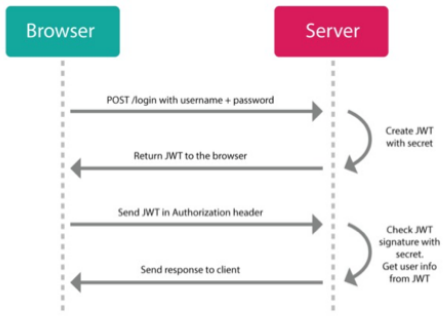

Browser
Server
POST /login with username + password
Create JWT with secret
Return JWT to the browser
Send JWT in Authorization header
Check JWT signature with secret. Get user info from JWT
Send response to client

Figure 3: Client-Server interactions with JWT.

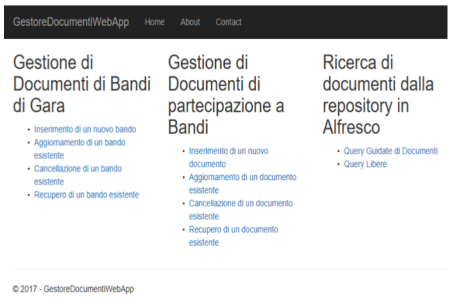
requesting a HTTP method, the token is inserted in the SOAP header of the request. The server grants the access to the incoming requests with valid tokens. Figure 3 shows such interaction. To this aim, we have extended the architecture in Figure 1 by adding an additional war hosting the functionalities of managing the identities, requesting token by giving a valid couple of username and password, checking the validity of a given token, revoke the grant for a given user. It is evident that all our services strongly interact with such an additional set of RESTful web services so as to decide to grant or deny a received request.

3.4 Prototype

We have implemented our solution within the context of the .NET framework, and run a set of unit tests to verify the correct behaviour of the invoked operations by using the GUI offered by the share.war of the Alfresco product. Figure 4a shows the realized web pages as GUI of our implemented web services, and Figure 4b shows the form for the insertion of a new document filled with some testing text. Figure 4c shows that the new document is eventually available within the repository managed by Alfresco.

4 FUTURE RESEARCH DIRECTIONS

The proposed system is just the starting point for further speculations and research on the addressed topic. Several open challenges are still open, among which the most interesting one is how to improve the search expressiveness. Specifically, CMIS and Alfresco support a SQL-like query language, consisting in SQL-92 SELECT statements. This exhibits a twofold drawback. On the one hand, operators are required to have a background on databases and SQL, which is not always the case as in civil engineering companies it is more reasonable to have users not skilled



Gestione di Documenti di Bandi di Gara

- Inserimento di un nuovo bando
- Aggiornamento di un bando esistente
- Cancellazione di un bando esistente
- Recupero di un bando esistente

Gestione di Documenti di partecipazione a Bandi

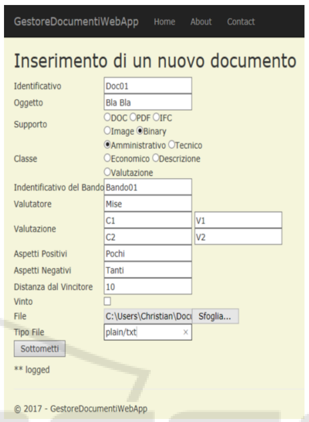
- Inserimento di un nuovo documento
- Aggiornamento di un documento esistente
- Cancellazione di un documento esistente
- Recupero di un documento esistente

Ricerca di documenti dalla repository in Alfresco

- Query Guidate di Documenti
- Query Libere

© 2017 - GestoreDocumentiWebApp

(a) Implemented GUI for the implemented RESTful web services.



Inserimento di un nuovo documento

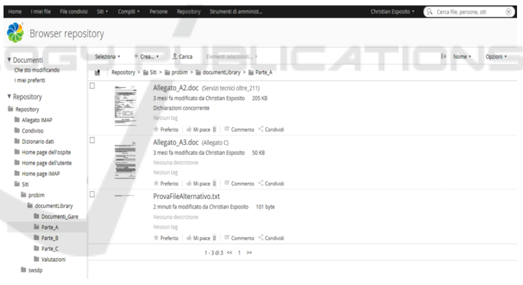
Identificativo	Doc01
Oggetto	Bla Bla
Supporto	DOC PDF IFC Image [✓ Binary]
Classe	[✓ Amministrativo] Tecnico Economico Descrizione Valutazione
Indentificativo del Bando	Bando01
Valutatore	Mise
Valutazione	C1 V1
	C2 V2
Aspetti Positivi	Pochi
Aspetti Negativi	Tanti
Distanza dal Vincitore	10
Vinto	☐
File	C:\Users\Christian\Docu Sfoglia...
Tipo File	plain/txt

Sottometti

** logged

© 2017 - GestoreDocumentiWebApp

(b) Example of the use of the prototype to insert a new document.



Repository > Siti > probim > documentLibrary > Parte_A

Allegato_A2.doc (Servizi tecnici oltre_211)
3 mesi fa modificato da Christian Esposito 205 KB
Dichiarazioni concorrente
Nessun tag
Preferito | Mi piace 0 | Commento | Condividi

Allegato_A3.doc (Allegato C)
3 mesi fa modificato da Christian Esposito 50 KB
Nessuna descrizione
Nessun tag
Preferito | Mi piace 0 | Commento | Condividi

ProvaFileAlternativo.txt
2 minuti fa modificato da Christian Esposito 101 byte
Nessuna descrizione
Nessun tag
Preferito | Mi piace 0 | Commento | Condividi

1 - 3 di 3 << 1 >>

(c) Check the presence of the new document in Alfresco.

Figure 4: Prototype testing (local GUI in Italian).

on computer science basic concepts. On the other hand, the expressiveness of the query is limited as it is performed on the term-matching. In the WHERE clause, the user indicates some terms upon which the search must be done. Such a query term must be found in the meta-data of the documents in order to have such documents returned by the query. Slight changes (such as the query term expressed in its singular form, while some documents have it as plural) imply a lower number of documents being retrieved. Moreover, semantic-based queries may be needed as various synonymous can be used within the documents that we want to be returned as a result of a query. To this aim more advanced solutions need to be coupled with Alfresco, without recurring to ontologies and their supporting framework, as their effectiveness is questionable and their usability has a huge learning trend (Durán-Muñoz and Bautista-Zambrana, 2017; Schulz et al., 2009).

Another aspect worth deepening may be how to transfer such a system within the cloud, so that it becomes accessible even in case of companies that have multiple premises spanning around the world. However, a change like that implies that the sensitive company data may be stored outside the security domain and control of the company, thus vulnerable to possible leakage and misuses. To this aim, it is crucial to protect its confidentiality and integrity by means of proper cryptographic primitives (Singh et al., 2016), without having to pay the cost of a reduced processing and retrieval capacity offered by the system (Fu et al., 2016). Moreover, our security only means allowing the management of the authentication and authorization when interacting with the system, but other security-related issues, such as protecting the SOAP messages exchanging, detecting a possible misbehaviour and managing the trust degree of the interacting users, are not minor concerns and need to be properly investigated. Last, JWT applies a very simple access control model, while more advanced and dynamic models are available in the literature, such as in (Esposito, 2018) and may be integrated within the solution by extending what provided by JWT. This may allow to achieve a more fine-grained control over the access requests and to implement more resilient solutions when companies encompasses multiple consultants on specific projects.

5 CONCLUSIONS

The present work intended to provide some insights as to the challenging and novel issue of storing and managing all the documents that a civil engineering company may produce in its life time when participating to a project and/or responding to a call-for-tender issues by a public administration. Our starting point was represented by a content management system tailored to this application domain built on top of Alfresco, by implementing a set of RESTful web services within the .NET framework. We have indicated the possible research direction on improving the easiness and expressiveness of the query language and dealing with the key security demands in the system, especially in the case of its cloudification. We believe that similar issues arise in other domains, from the healthcare to the manufacturing industry, where unstructured documents must be stored and queried. Apart from the details of the properties described in Section 3 and the repository structure in Figure 2, all the design and relative implementation can be easily adapter for other domains.

ACKNOWLEDGMENT

The described work has been partially supported by the PROBIM research project, funded by the Italian Ministry of Economic Development within the context of Horizon 2020 - PON I&C 2014-20.

REFERENCES

Atkinson, B. et al. (2002). Web services security (ws-security). specification. In *Available at https://www.it.iitb.ac.in/∼madhumita/research_topics/authentication/WS%20Security.pdf.*

Barkley, J. (1997). Comparing simple role based access control models and access control lists. In *Proceedings of the second ACM workshop on Role-based access control.*

Çetiner, O. (2010). A review of building information modeling tools from an architectural design perspective. in: Handbook of research on building information modeling and construction informatics: Concepts and technologies. In *IGI Global.*

Durán-Muñoz, I. and Bautista-Zambrana, M. R. (2017). Applying ontologies to terminology: Advantages and disadvantages. In *HERMES-Journal of Language and Communication in Business, 26(51):65-77.*

Eastman, C., Teicholz, P., Sacks, R., and Liston, K. (2018). Bim handbook - a guide to building information modeling for owners, managers, designers, engineers and contractors. In *John Wiley & Sons Inc, 3rd edition.*

Esposito, C. (2018). Interoperable, dynamic and privacy-preserving access control for cloud data storage when integrating heterogeneous organizations. In *Journal of Network and Computer Applications, 108: 124-136.*

EUR-Lex (2014). Directive 2014/24/eu of the european parliament and of the council of 26 february 2014 on public procurement and repealing directive 2004/18/ec text with eea relevance. In *D. Available on line at https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32014L0024.*

Fu, Z., Ren, K., Shu, J., Sun, X., and Huang, F. (2016). Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems*, 27(9):2546–2559.

Grabowski, R. (1999). Hpgl overview. In *Available at cstep.luberth.com.*

Jones, M., Bradley, J., and Sakimura, N. (2015). Json web token (jwt). no. rfc 7519. In *Available at https://jwt.io/.*

Land, R., Sundmark, D., Lüders, F., Krasteva, I., and Causevic, A. (2009). Reuse with software components-a survey of industrial state of practice. In *International Conference on Software Reuse, 150-159*.

Liewald, M. H. (1985). Initial graphics exchange specification: successes and evolution. In *Computers & graphics 9.1*.

Luzadder, W. J. (1992). Introduction to engineering drawing: The foundations of engineering design and computer aided drafting. In *Prentice Hall PTR*.

OASIS (2015). Content management interoperability services (cmis) - version 1.1. In *Available at http://docs.oasis-open.org/cmis/CMIS/v1.1/CMIS-v1.1.html*.

OASIS (2017). extensible access control markup language (xacml) v3.0 approved as an oasis standard. In *Available at https://www.oasis-open.org/committees/xacml/*.

Richardson, L. and Ruby, S. (2007). Restful web services. In *O'Reilly Media*.

Schulz, S., Stenzhorn, H., Boeker, M., and Smith, B. (2009). Strengths and limitations of formal ontologies in the biomedical domain. In *Revista electronica de comunicacao, informacao & inovacao em saude: RECIIS, 3(1):31*.

Shariff, M., Bhandari, A., and V. Choudhary, P. M. (2009). Alfresco 3 enterprise content management implementation. In *Packt Publishing*.

Singh, S., Jeong, Y.-S., and Park, J. H. (2016). A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications*, 75:200 – 222.

Venugopal, M., Eastman, C., Sacks, R., and Teizer, J. (2012). Semantics of model views for information exchanges using the industry foundation class schema. In *Advanced Engineering Informatics, 26(2)*.