



NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

**SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS**

BSc THESIS

**Re-engineering Nomothesi@ API Web Application:
Improvements and Support of new features**

Georgios C. Apostolopoulos

Supervisors:

**Manolis Koubarakis, Professor
Ilias Chalkidis, PhD Candidate**

ATHENS

JUNE 2018



ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Αναδιοργάνωση της Διαδικτυακής Πλατφόρμας Νομοθεσί@
API: Βελτιώσεις και Προσθήκη νέων λειτουργιών**

Γεώργιος Χ. Αποστολόπουλος

Επιβλέποντες:

**Μανόλης Κουμπάρκης, Καθηγητής
Ηλίας Χαλκίδης, Υποψήφιος Διδάκτωρ**

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2018

BSc THESIS

Re-engineering Nomothesi@ API Web Application: Improvements and Support of new features

Georgios C. Apostolopoulos

S.N.: 1115201200006

Supervisors:

Manolis Koubarakis, Professor
Ilias Chalkidis, PhD Candidate

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναδιοργάνωση της Διαδικτυακής Πλατφόρμας Νομοθεσί@ API: Βελτιώσεις και Προσθήκη νέων λειτουργιών

Γεώργιος Χ. Αποστολόπουλος

A.M.: 1115201200006

Επιβλέποντες:

**Μανόλης Κουμπάρκης, Καθηγητής
Ηλίας Χαλκίδης, Υποψήφιος Διδάκτωρ**

ABSTRACT

The purpose of this thesis is to radically re-engineer Nomothesi@ API web platform and to add new features. The starting point was the previous works titled "Nomothesi@: Greek Legislation Platform" (2014) and "Nomothesi@ API: Re-engineering the Electronic Platform" (2015). The existing platform, based on a consolidated legal XML/RDF template, presented inaccuracies in both the presentation of legal documents and the functionality it provided to the user. This work emphasizes on replacing the storage of legal documents in classes, by a N-ary tree structure in each level. This replacement, as well as all the other minor modifications, were made possible as a result of the new ontology that was created, always based on the European Legislation Identifier (ELI). Simultaneously, features were added to extend user's interaction with the application and to transmit a larger amount of information. The most important addition is the insertion of entities that allow the user to obtain further information about a person, a place, etc. Thus, this thesis contributes more to the European Union's effort to enhance e-Government by its member states, through the open publication of the whole of the Greek legislative act. It essentially develops a new way of storing information, as it is derived from the RDF data schema and it introduces innovative ideas on how to make use of it, providing additional features.

SUBJECT AREA: Semantic Web, Linked Data, Artificial Intelligence, Web Applications

KEYWORDS: RDF/OWL Metadata, Legal Document, E-Government, XML, N-ary Tree, Entities, References

ΠΕΡΙΛΗΨΗ

Σκοπός της συγκεκριμένης εργασίας είναι η ριζική αναδιοργάνωση του τρόπου λειτουργίας της διαδικτυακής πλατφόρμας Νομοthesi@ API και η προσθήκη νέων λειτουργιών. Βάση εκκίνησης αποτέλεσαν οι προγενέστερες εργασίες με τίτλους “Νομοthesi@: Πλατφόρμα για την Ελληνική νομοθεσία” (2014) [1] και “Νομοthesi@ API: Αναδιοργάνωση της Ηλεκτρονικής Πλατφόρμας” (2015) [2]. Η ήδη υπάρχουσα ηλεκτρονική πλατφόρμα, η οποία στηρίζεται σε ένα ενοποιημένο νομικό XML/RDF πρότυπο, παρουσίαζε προβλήματα, τόσο στην παρουσίαση των νομικών εγγράφων, όσο και στις λειτουργίες που παρείχε στο χρήστη. Στην εργασία αυτή, δόθηκε έμφαση στην αντικατάσταση της αποθήκευσης σε κλάσεις του νομικού εγγράφου, από ένα σύστημα το οποίο αποτελείται από μια δενδρική δομή N κόμβων σε κάθε επίπεδο. Στην πραγματοποίηση αυτής της αλλαγής αλλά και όλων των υπολοίπων μικρότερων τροποποιήσεων, συνέβαλε η νέα οντολογία που δημιουργήθηκε, έχοντας πάντα ως βάση το European Legislation Identifier (ELI) [3]. Ταυτόχρονα, προστέθηκαν λειτουργίες που σκοπό είχαν να επεκτείνουν το βαθμό αλληλεπίδρασης του χρήστη με την εφαρμογή και να του μεταδώσουν πληθώρα πληροφοριών. Η βασικότερη προσθήκη που έλαβε χώρα είναι αυτή των οντοτήτων, οι οποίες επιτρέπουν στο χρήστη να λάβει πληροφορία περαιτέρω των νομικών θεμάτων, σχετικά με κάποιο πρόσωπο, κάποιον τόπο κλπ. Κατά τον τρόπο αυτό, η εργασία συμβάλει ευρύτερα στην προσπάθεια της Ευρωπαϊκής Ένωσης για ενίσχυση της ηλεκτρονικής διακυβέρνησης από τα κράτη-μέλη της [4], μέσω της ανοιχτής δημοσίευσης της ελληνικής νομοθετικής πράξης στο σύνολο της. Υποδεικνύει, ουσιαστικά, έναν νέο τρόπο αποθήκευσης πληροφορίας, όπως αυτή λαμβάνεται από το RDF σχήμα δεδομένων και εισάγει καινοτόμες ιδέες για την αξιοποίηση της, παρέχοντας επιπλέον λειτουργίες στο χρήστη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Σημσιολογικός Ιστός, Διασυνδεδεμένα Δεδομένα, Τεχνητή Νοημοσύνη, Εφαρμογές Διαδικτύου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: RDF/OWL Μεταδεδομένα, Νομικό Έγγραφο, Ηλεκτρονική Διακυβέρνηση, XML, Δένδρο N κόμβων, Οντότητες, Αναφορές

ACKNOWLEDGMENTS

To my family, for being my personal life coach since the day I was born. Thank you for your unceasing encouragement and support!

Having finished this thesis, I would like to express my sincere gratitude to my supervisor Prof. Manolis Koubarakis for his assistance and careful guidance through the whole writing process.

I am also grateful to Ilias Chalkidis, PhD Candidate and Researcher in the Dpt. of Informatics and Telecommunications of National Kapodistrian University of Athens. I am thankful for his choice to believe in me and allow me to be a part of the project he has started.

Finally, I place on record my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

CONTENTS

1. INTRODUCTION.....	11
1.1 Nomothesi@ in contemporary society.....	11
1.2 Objectives of the thesis.....	11
1.3 Nomothesi@ Ontology.....	12
1.4 Thesis structure.....	13
2. N-ARY TREE IMPLEMENTATION.....	14
2.1 Old storing system issues.....	14
2.2 How is a N-ary Tree beneficial.....	17
2.3 Tree Implementation.....	19
2.3.1 LegislationTreeNode.....	20
2.3.2 Brief explanation of the two main functions.....	21
2.3.2.1 The getByld(...) method.....	21
2.3.2.2 The getUpdatedByld(...) method.....	22
2.3.3 Modification Nodes.....	22
2.3.3.1 Modification type: Insertion.....	23
2.3.3.2 Modification type: Substitution.....	23
3. ENTITIES.....	25
3.1 What is a reference and how it works.....	25
3.2 Entity types.....	26
3.2.1 Person entity.....	26
3.2.2 Landmark entity.....	27
3.2.3 Geopolitical entity.....	28
3.2.4 Organization entity.....	29
3.3 Entity Queries.....	30
3.4 Entity search page.....	32
4. MINOR ADDITIONS AND NEW FUNCTIONALITIES.....	34
5. GENERAL MODIFICATIONS.....	39
6. CONCLUSION.....	42
ABBREVIATIONS – ACRONYMS.....	43
TECHNOLOGIES – LIBRARIES USED.....	44
REFERENCES.....	45

LIST OF FIGURES

Figure 1: Nomothesi@ Ontology for the Greek legislation.....	12
Figure 2: Legal Document representation in unsorted format.....	18
Figure 3: Legal Document representation after being sorted.....	18
Figure 4: Simple modification in tree representation.....	19
Figure 5: Modification insertion example.....	23
Figure 6: Legislation Tree Node before and after substitution.....	24
Figure 7: Reference example.....	25
Figure 8: Person entity example page.....	27
Figure 9: Landmark entity example page.....	28
Figure 10: Geopolitical entity example page.....	29
Figure 11: Organization entity sample page.....	30
Figure 12: Entity search page.....	33
Figure 13: References found inside the document sample page.....	34
Figure 14: Legal resources' references to the current legal document.....	35
Figure 15: Ministry stats graph.....	36
Figure 16: Ministries in signers tab (page screenshot).....	37
Figure 17: Insertion in timeline.....	38
Figure 18: Substitution in timeline.....	38
Figure 19: PDF file sample.....	39
Figure 20: Statistics graph sample.....	40
Figure 21: Modification appearance in the document.....	40
Figure 22: Legal document main web page.....	41

LIST OF TABLES

Table 1: Legislation Tree Node attributes.....	20
Table 2: Greek Local Government architecture based on "Kallikratis Plan"	28
Table 3: Libraries used for the builders.....	39
Table 4: Table of abbreviations.....	43
Table 5: Libraries used in the project.....	44

1. INTRODUCTION

1.1 Nomothesi@ in contemporary society

Nowadays, it has been clarified in many ways, that law enforcement can be assisted by technology [5]. Observing the technological growth over the past decades, anyone can understand the necessity of developing online platforms and tools that simplify legislation services. Therefore, legislative act should not be used only by lawyers, but also support a wider group of people in their jobs and, most importantly, in their lives. Nomothesi@'s objective has been our ambition to establish justice, starting by bringing the public closer to legislation.

1.2 Objectives of the thesis

The main objective of this thesis is the implementation of a database-free web application, based upon Linked Data and the Semantic Web [6]. A previous platform had been developed in order to achieve this goal, but even though it was on the right direction, the details were insufficient at some points, leading to incorrect outcomes.

To overcome this problem, we decided to develop a more efficient storing system for the law components, because the old one, in which every part of the law belonged to a different class, didn't produce the expected results. Therefore, we came up with the idea of keeping all the components in the same class, in a manner that would maintain its consistency, but also link them properly. Using a N-ary tree¹ seemed to be a reliable solution, since every law component has its subparts as a list of children nodes and the component it belongs to as a parent node. The main advantage of this implementation is that every single part of the law is characterized by the same attributes with its parent and its children.

Furthermore, we wanted to provide additional information to the user, not only about the the law, but also about what is referenced in the document. We found it would be useful for someone to learn details about a person, a place, or even about an organization. For this purpose, we inserted the concept of the entities. An entity has specific attributes, depending on what it refers to.

Finally, we focused on adding some new functionalities to the platform and improving some that didn't work properly. Thus, we have asked ourselves what we'd like to be included in the application and what is malfunctioning. This motivated us to find innovative ideas on what a user would need to be offered and how it should be implemented.

¹ A tree with no more than N children for each node.

1.3 Nomothesi@ Ontology

Recently the European Council introduced the European Legislation Identifier (ELI) as a framework which has to be adopted by the national legal publishing systems in order to link national legislation with European legislation. ELI proposes a URI schema for the identification of legal resources on the web and it also provides an OWL ontology, which is used for expressing metadata of legal documents and legal events. ELI, like MetaLex, has to be extended to capture the particularities of national legislation systems. In our project, ELI is used for encoding Greek legislation, expressing its metadata and analyzing legislative modifications acting upon legal documents [7].

For better understanding the functionalities mentioned in this thesis, we provide our ontology's schema. This chart contains basic information for the comprehension of the manner that data is linked.

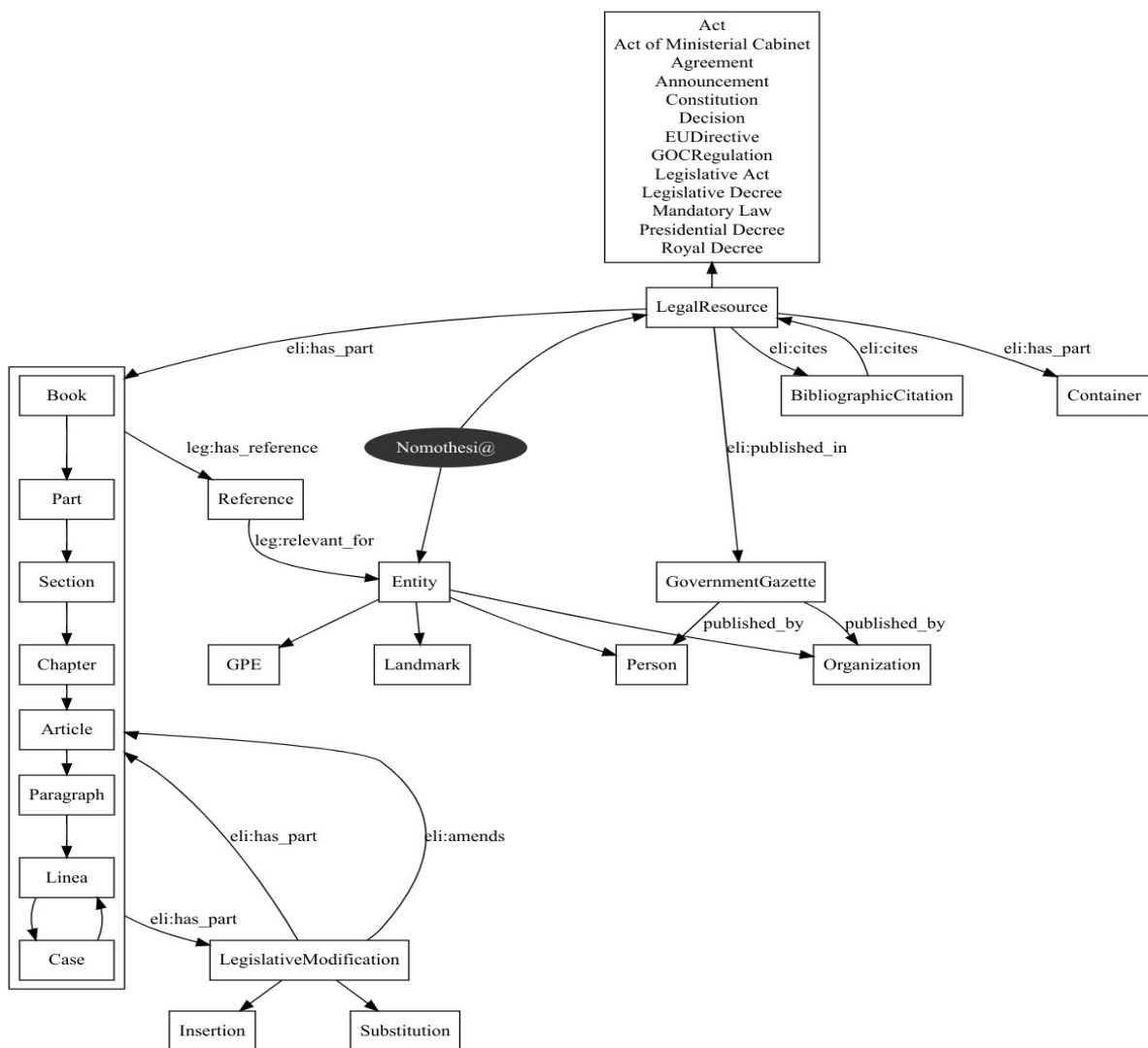


Figure 1: Nomothesi@ Ontology for the Greek legislation

1.4 Thesis structure

This thesis is separated into six chapters, with the first and the last one being the introduction and the conclusion respectively. Chapter B refers to the N-ary tree and analyzes both the issues that the old implementation had and how the tree deals with them. Finally, it scrutinizes its implementation. Chapter C mentions the entities, clarifying their necessity, their categories and how they are developed in terms of code. Chapters D and E make reference to additions and modifications that have been made, with regard to minor, but important, functionalities.

2. N-ARY TREE IMPLEMENTATION

In this chapter, we will be discussing the necessity of replacing the old storing method by a new one which will be based on a N-ary tree. More precisely, we will mention the insufficiencies of the previous storage system and we will explain in detail how the tree will help us overcome those issues. Finally, we will examine the implementation of the tree in terms of code.

2.1 Old storing system issues

The previous storing system had been keeping information in classes which had been different for every subpart type. An Article, for example, belonged to a different class than a Paragraph, a Part was a different class than a Chapter and so on. However, we have noticed that apart from the different sense of those legal resource components, the main information they hold is the same and they can easily be differentiated by their URI.

In most cases, RDF registrations are randomly distributed in the file. It is therefore difficult for the system to determine whether a class is a part of another. Linked data need to be carefully structured to let us know the plot of a legal document. The problem occurs when in the data file, there is a triple that refers to a legal part and then, after putting references to other components, it refers to this part again. For instance, consider the following set of triples:

```
<http://legislation.di.uoa.gr/eli/pd/2016/72/article/1>  
<http://data.europa.eu/eli/ontology#has_part>  
<http://legislation.di.uoa.gr/eli/pd/2016/72/article/1/paragraph/1>  
  
<http://legislation.di.uoa.gr/eli/pd/2016/126/article/17>  
<http://data.europa.eu/eli/ontology#has_part>  
<http://legislation.di.uoa.gr/eli/pd/2016/126/article/17/paragraph/3>  
  
<http://legislation.di.uoa.gr/eli/law/2016/38/article/1/paragraph/1>  
<http://data.europa.eu/eli/ontology#has_part>  
<http://legislation.di.uoa.gr/eli/law/2016/38/article/1/paragraph/1/linea/2>  
  
<http://legislation.di.uoa.gr/eli/pd/2016/72/article/1/paragraph/1>  
<http://data.europa.eu/eli/ontology#has_part>  
<http://legislation.di.uoa.gr/eli/pd/2016/72/article/1/paragraph/1/linea/1>
```

We can observe that the first triple sets paragraph 1 to be a part of article 1 of the Presidential Decree 72 of 2016. A couple of triples follow and then, in the fourth triple, linea 1 is set to be a part of the previously mentioned paragraph. This, however, seems hard to manage in terms of code, as we must keep track of every single class instance in order to add new parts and link them with the existing ones. This explains the lack of

consistency that can occur, especially when we deal with a large amount of randomly placed RDF registrations.

Furthermore, sorting elements used to be a problematic task because of the fact that they were not stored using the same class. For example, based on the URI, we can note that `<http://legislation.di.uoa.gr/eli/law/2016/4441/part/1/article/12>` would go first compared to `<http://legislation.di.uoa.gr/eli/law/2016/4441/article/3>`, because 12 is before 3 alphabetically. However, even after solving this issue, we realized that this solution wasn't always reliable. In case we had to compare the URI `<http://legislation.di.uoa.gr/eli/law/2016/4441/part/3/chapter/2>` with the URI of a subpart, e.g `<http://legislation.di.uoa.gr/eli/law/2016/4441/part/3/chapter/2/article/28>`, the second one would go first. But this leads to a result that doesn't make sense. Taking into consideration the titles of those URIs can help clarify the issue. More specifically:

```
<http://legislation.di.uoa.gr/eli/law/2016/4441/part/3/chapter/2>  
<http://data.europa.eu/eli/ontology#has_part>  
<http://legislation.di.uoa.gr/eli/law/2016/4441/part/3/chapter/2/article/28>
```

```
<http://legislation.di.uoa.gr/eli/law/2016/4441/part/3/chapter/2>  
<http://data.europa.eu/eli/ontology#title>  
"ΔΙΑΤΑΞΕΙΣ ΑΡΜΟΔΙΟΤΗΤΑΣ ΥΠΟΥΡΓΕΙΟΥ ΠΟΛΙΤΙΣΜΟΥ ΚΑΙ ΑΘΛΗΤΙΣΜΟΥ  
ΣΕ ΕΦΑΡΜΟΓΗ ΣΥΣΤΑΣΕΩΝ ΕΡΓΑΛΕΙΟΘΗΚΗΣ ΤΟΥ ΟΟΣΑ"
```

```
<http://legislation.di.uoa.gr/eli/law/2016/4441/part/3/chapter/2/article/28>  
<http://data.europa.eu/eli/ontology#title>  
"Καταργητικές διατάξεις"
```

After printing the sorted legal document we would expect the following outcome:

ΔΙΑΤΑΞΕΙΣ ΑΡΜΟΔΙΟΤΗΤΑΣ ΥΠΟΥΡΓΕΙΟΥ ΠΟΛΙΤΙΣΜΟΥ ΚΑΙ ΑΘΛΗΤΙΣΜΟΥ
ΣΕ ΕΦΑΡΜΟΓΗ ΣΥΣΤΑΣΕΩΝ ΕΡΓΑΛΕΙΟΘΗΚΗΣ ΤΟΥ ΟΟΣΑ

Καταργητικές διατάξεις

However, we would be given the exact opposite, which, in terms of legal comprehension, seems meaningless.

Additionally, we must mention the insufficiency that occurs due to the fact that we used to have the modifications in a different class than the other parts. Modifications are parts of a legal document and can be a single part of the law, or multiple parts combined. For instance, a passage can be a modification itself and an article can be a modification along with its subcomponents. Thus, it is obvious that it would be beneficial to use a universal class that can be manipulated in any case. Handling modifications can be quite complicated. For instance, consider the following triple:

```
<http://legislation.di.uoa.gr/eli/law/2014/4268/article/5/paragraph/2/passage/1/
modification/7>
<http://data.europa.eu/eli/ontology#has_part>
<http://legislation.di.uoa.gr/eli/law/2014/4268/article/5/paragraph/2/passage/1/
modification/7/paragraph/9>
```

This triple indicates that modification/7 contains paragraph/9. It would be rather easy if modification had multiple subcomponents, but what happens in case paragraph/9 had its own children? This scenario is actually what happens almost every time. As a matter of fact, in our case:

```
<http://legislation.di.uoa.gr/eli/law/2014/4268/article/5/paragraph/2/passage/1/
modification/7/paragraph/9>
<http://data.europa.eu/eli/ontology#has_part>
<http://legislation.di.uoa.gr/eli/law/2014/4268/article/5/paragraph/2/passage/1/
modification/7/paragraph/9/passage/1>
```

We would have to find the URI of the modification and then search to find all its subparts recursively until no other component existed. But even if we could do it reliably, it would be of great cost to sort all the parts in each level and for every single modification. Above this issue of course, comes the first problem we have mentioned in this section, that there can be various triples between the ones that refer to the same modification. It is worth noting, however, that there could be a reliable solution to this problem, but it wouldn't be the optimal one, as it is programmatically inefficient to include multiple different classes in another class. Besides, as we have previously commented, a modification is nothing more than a legal part along with its subparts.

Another problem that indicates the lack of efficiency of the old storing system is that there is not a methodical programming way to store the legal parts in their super classes. To illustrate this issue, notice this single triple:

```
<http://legislation.di.uoa.gr/eli/agr/2016/1_10.08.2016/article/2>
<http://data.europa.eu/eli/ontology#has_part>
<http://legislation.di.uoa.gr/eli/agr/2016/1_10.08.2016/article/2/paragraph/1>
```

Now, imagine what a program must do to store paragraph/1 in article/2. The Article class contains a list of Paragraphs, in which this paragraph must be added. This means we have to search and find the specific Article class instance, which means we have to find the parent instance of this article and probably level up further.

There can be a vast amount of cases to link properly every part with its parent class. It is worth mentioning that in the previous implementation the function that used to deal with this task consisted of more than 1200 lines of code, full of if-else statements and lead to an absolute confusion when debugging was needed. This issue also occurred in the case of the .jsp page that previewed the legal document. This page had to take a huge number of decisions in order to determine the type of the legal part. The partitioning, used to be a complicated work, especially for larger legal documents.

Therefore, anyone can realize the necessity of developing a new storing system, which could definitely reduce coding complexity. Nomothesi@ needed a fresh, brand new implementation and this is how we came up with the idea of storing the data into a N-ary Tree.

2.2 How is a N-ary Tree beneficial

The most important characteristic of every component is to be able to keep track of both its subcomponents and its parent. Having knowledge of those two attributes, we can easily form the legal document's structure, no matter if it's a Law, a Presidential Decree or any other type of legislation.

Using a tree, simplifies the comprehension of a legal document's structure. Every time we need to add a new part to the document, we insert a new node to its predecessor (a.k.a parent node). The only thing we need to know is the URI of its parent to be able to insert the new node. This is easily implemented by a simple DFS traversal on the tree, searching each node for a matching URI with the one of the parent, which is accessible by a SPARQL query.

For example, the following triple indicates a parent-child relationship between two legal document's components:

```
<http://legislation.di.uoa.gr/eli/law/2016/4360/part/2/article/23/paragraph/1>  
<http://data.europa.eu/eli/ontology#has_part>  
<http://legislation.di.uoa.gr/eli/law/2016/4360/part/2/article/23/paragraph/1/linea/2>
```

This triple is accessible by the following query:

```
select ?p ?c  
where {  
    ?p eli:has_part ?c .  
}
```

Two tree nodes will be connected. The first node will represent the parent node and will have URI `<http://legislation.di.uoa.gr/eli/law/2016/4360/part/2/article/23/paragraph/1>`. It will also have a list of children, in which there will be a node possessing the URI `<http://legislation.di.uoa.gr/eli/law/2016/4360/part/2/article/23/paragraph/1/linea/2>` and it will keep track of its parent node.

Thus, it becomes easy to obtain the whole structure of the legal document using a SPARQL query and then create the tree by inserting new children by implementing DFS algorithm to search for its parent and add the newly created node to its children list. This can be helpful, especially if we consider the fact that the document will be finely structured even if the query returns the triples in a totally random order. Thanks to the DFS algorithm, our system's reliability is granted, as well as efficiency, having noticed that this algorithm is used in many major web applications and computer softwares.

Furthermore, sorting the legal document is a simple task. There is a node comparator that compares the URIs of all the nodes that are on the same level and it repeats the process for the whole tree. For this reason, a legal document can be easily sorted by

ordering the nodes on each level, traversing the tree using DFS. To make it more clear, we can consider the following unsorted tree:

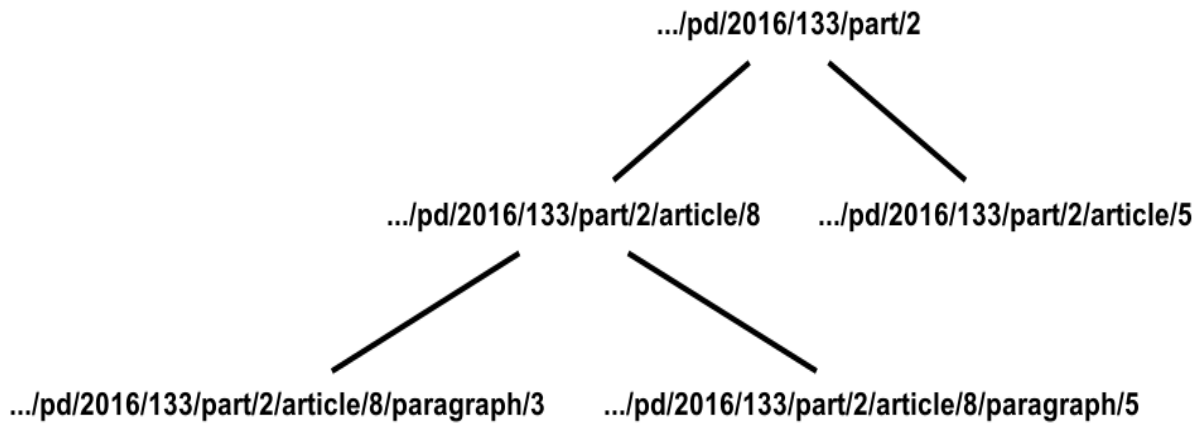


Figure 2: Legal Document representation in unsorted format

After being sorted, the legal document will appear like the image below:

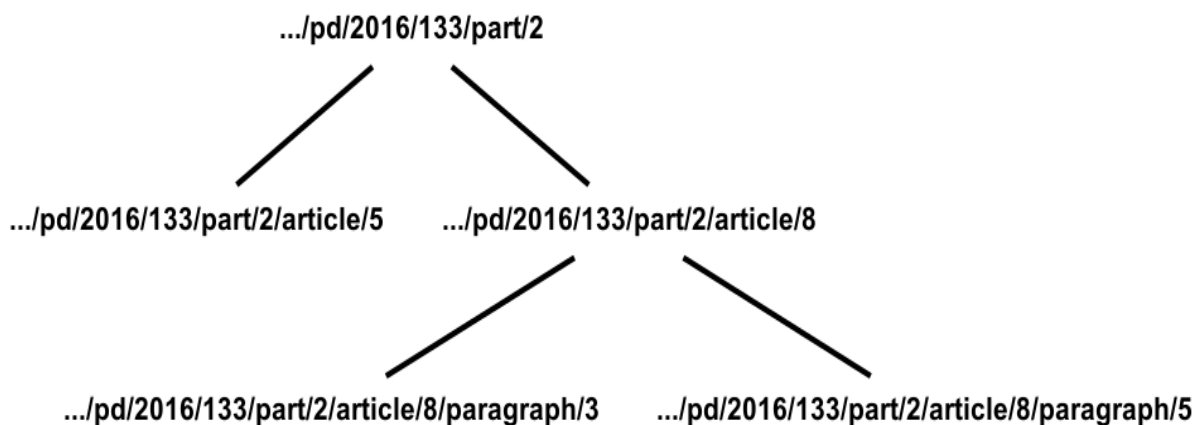


Figure 3: Legal Document representation after being sorted

As a result of the tree being easily used, it becomes much easier to convert java code into HTML and show the legal document on a web page. Once the legal resource is finally sorted, applying a DFS traversal is enough to preview the whole structure of the document. The same applies also on the PDF and XML builders, as they follow the same procedure in order to be generated. Therefore, the 1200 lines of code that we have noted in the previous section have now been replaced by a function that consists of only 200 lines approximately. In those lines, we have furthermore managed to reduce

the number of cases and if-else statements by more than 90%. This constitutes a great achievement, considering the fact that we managed to diminish code complexity significantly and make our application even faster, preserving reliability and efficiency.

Finally, the tree has changed our entire perception as far as the modifications are concerned. Modifications, as we have already noted, used to be a separate class, but now they are simple trees, like every legal document part. It has been our primary goal to create a universal type of modification that can be applied in any type of legal resource component. Therefore, a tree seems to be a perfect structure to achieve this objective. Providing a snapshot of how a modification can be illustrated in the current implementation, can be quite convincing:

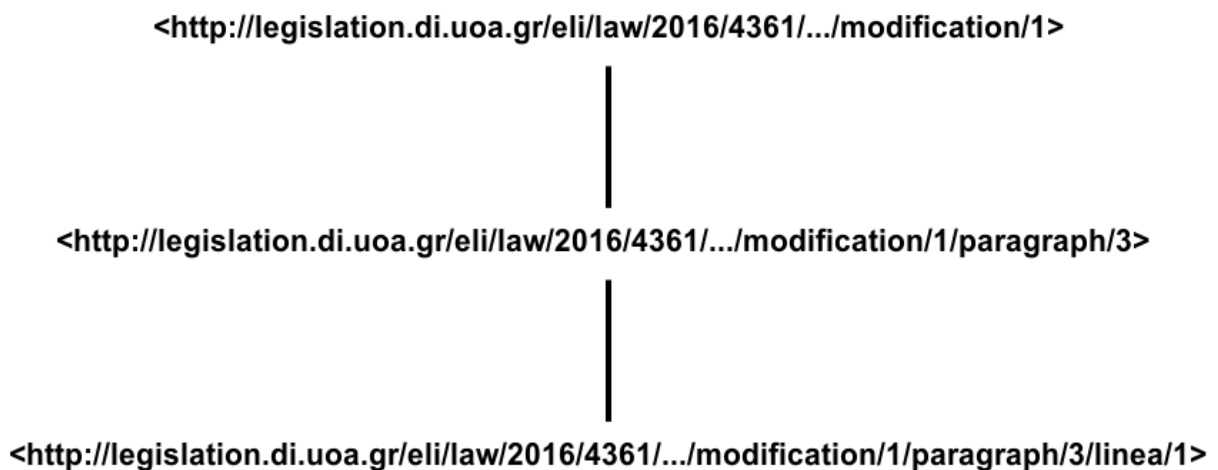


Figure 4: Simple modification in tree representation

A modification is usually referenced in another legal document's passage. Then, we form the structure as a tree. Now, the modification has a complete body and the program is fully aware of the replacement or the Insertion that must be executed. We will thoroughly explain how modifications work in the next part.

2.3 Tree Implementation

The N-ary tree we have chosen to implement for storing the legal document's structure, consists basically of one class acting as a tree node. This node contains a list of its children and also keeps track of its parent. This way, consistency is achieved for the whole document, as long as the linking of the nodes seems to be a rather simple process.

On the first part, we will indicate the basic attributes of the node. Additionally we will mention the modification nodes and in the final subsection we will explain everything about the tree nodes.

2.3.1 LegislationTreeNode

The tree node consists of the following attributes:

Table 1: Legislation Tree Node attributes

String URI	Node's URI (e.g. .../paragraph/2)
String id	Node's id (e.g. .../paragraph/4 has id=4)
String text	Node's text. May be empty (e.g. Sections don't have text normally)
String title	Node's title. May be empty (e.g. Passages don't have title normally)
Boolean isMod	True in case of Modification Node
List <LegislationTreeNode> children	List of node's children
List <Reference> references	List of referenced nodes and parts
LegislationTreeNode parent	Node's parent node
List <LegislationTreeNode> previousEditions	Every list item is a tree that contains a previous edition of this node. When a modification occurs, a new entry is inserted
String patient	For modification nodes. Patient node's URI
String patientType	For modification nodes. Patient node's type
String modType	For modification nodes. Can be either Substitution or Insertion
String pubDate	For modification nodes. Publication date
String fek	The FEK in which the modification is published

As we have already stated, those attributes can be acquired by a single SPARQL query. This query is provided below and the parameters decisionType, year and id are used by a function depending on the user's choice:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n+
PREFIX leg: <http://legislation.di.uoa.gr/ontology#>
PREFIX eli: <http://data.europa.eu/eli/ontology#>
```

```
SELECT DISTINCT ?part ?parent ?text ?ref ?reflabel ?legreftitle ?start ?end ?
entity ?type ?title ?filename
WHERE{
    <http://legislation.di.uoa.gr/eli/decisionType/year/id> eli:has_part+ ?part.
    ?parent eli:has_part ?part.
    ?part rdf:type ?type.
    OPTIONAL{ ?part leg:has_text ?text.
    OPTIONAL{?part leg:has_reference ?ref.
    ?ref leg:starts ?start.
    ?ref leg:ends ?end.
    ?ref rdfs:label ?reflabel.
    OPTIONAL{ ?ref eli:title ?legreftitle.}.
    ?ref leg:relevant_for ?entity. } }.
    OPTIONAL{ ?part eli:title ?title.}.
    OPTIONAL{ ?part leg:imageName ?filename.}.
}
ORDER BY ?part
```

2.3.2 Brief explanation of the two main functions

In the process of developing an efficient tree structure, we have also implemented two important functions to make sure data is stored correctly in the nodes. The code for these two is significantly big and, therefore, we will explain briefly their functionalities.

2.3.2.1 The getByld(...) method

This functions forms the legal resource structure, or basically, the tree. The main iteration behind this work is the following:

- It executes a SPARQL query that provides the legal parts.
- If the tree is empty, it creates the root node.
- It creates a node including the information given by the query (e.g. Uri, id, etc.).
- It searches for the parent of this node inside the tree.
- It inserts the nodes in the parent's children list.
- It adds the references

In the end, when the process is finished, it sorts the tree.

2.3.2.2 The `getUpdatedByld(...)` method

This function applies the modifications upon the tree.

- It creates a list of `LegislationTreeNode`s which includes modification nodes.
- It executes a query for the modifications and forms small modification trees, which adds to the list.
- It sorts the list.
- For every list item (a.k.a modification) it applies one of either `addModificationTree(...)` function, or `replaceModificationTree(...)` function.

2.3.3 Modification Nodes

A Modification Node is a `LegislationTreeNode` that acts as a modification. Therefore, the boolean value of `isMod` is set to `True`. A modification has the structure that is indicated in the figure 4. Once we have discovered the word “modification” in the URI, then, we must set `True` the “`isMod`” attribute of the node characterized by the wanted uri (e.g. `<http://legislation.di.uoa.gr/eli/law/2016/4361/part/2/article/11/paragraph/1/linea/1/modification/1>`). Every child is returned by a SPARQL query using the `eli:has_part` (e.g. `<http://legislation.di.uoa.gr/eli/law/2016/4361/part/2/article/11/paragraph/1/linea/1/modification/1/paragraph/3>`). Thus, we can link those two nodes together and every other node that follows the same pattern, in order to form a little modification tree. This tree is, nothing more than a N-ary tree like the one we use to form the legal document.

Once the modification tree is ready, the application uses the patient variable of the node. This is a `String` variable that keeps the URI of the node in which this modification is going to act. There are two basic actions that can take place using a modification tree and they are expressed by the variable `modType`. It can either be an `Insertion`, or a `substitution`. Both modification types can be explained in three steps. We first notice the patient's uri. Then, we search the legal document in which the patient is published, to find the node that is characterized by this URI. We will now explain through code the different handling for those two modification types.

First of all, we quote the function that we use to search based on the node's URI.

```
public static LegislationTreeNode search(String name, LegislationTreeNode node) {
    if (node.getUri().equals(name)) {
        return node;
    }
    LegislationTreeNode res = null;
    for ( LegislationTreeNode ch: node.getChildren()) {
        res = search(name, ch);
        if( res!= null ) {
            return res;
        }
    }
    return null;
}
```

2.3.3.1 Modification type: Insertion

Having found the patient's URI we insert the new node in its children's list.

```
public static void addModificationTree(LegislationTreeNode root,
LegislationTreeNode position, LegislationTreeNode newroot) {
    LegislationTreeNode s = search(position.getUri(), root);
    s.getChildren().add(newroot);
    sortTree(root);
}
```

This function adds the modification tree (newroot) to the legal resource's tree (root) in the patient (position) children's list. By example, consider the following imaginary URIs, in which, the dotted line expresses the newly added node:

<http://legislation.di.uoa.gr/eli/law/2014/3300/article/23/paragraph/2> which already has children, is modified by:
 <http://legislation.di.uoa.gr/eli/law/2017/2222/article/2/paragraph/1/linea/2/modification/31> which has part:
 <http://legislation.di.uoa.gr/eli/law/2017/2222/article/2/paragraph/1/linea/2/modification/31/linea/4>

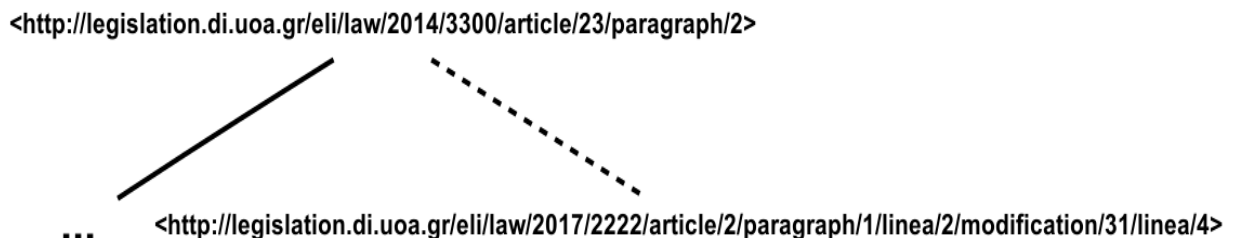


Figure 5: Modification insertion example

2.3.3.2 Modification type: Substitution

In this case we want to insert the new node in its patient's parent and remove the replaced node.

```
public static void replaceModificationTree(LegislationTreeNode
root, LegislationTreeNode position, LegislationTreeNode newroot) {
    LegislationTreeNode s = search(position.getUri(), root);
    LegislationTreeNode p = s.getParent();
    newroot.getPreviousEditions().add(s);
    p.getChildren().add(newroot);
    sortTree(root);
}
```

This function adds the new modification tree (newroot) to its patient's parent (position) children list. It also stores in a list the node that will be replaced to keep it as a previous edition.

In this case we quote two graphs to explain how this process works. First of all, we have `<http://legislation.di.uoa.gr/eli/law/2016/4361/part/2/article/11/paragraph/1/linea/1/modification/1>` which replaces: `<http://legislation.di.uoa.gr/eli/law/2003/3205/article/51/paragraph/3>` with: `<http://legislation.di.uoa.gr/eli/law/2016/4361/part/2/article/11/paragraph/1/linea/1/modification/1/paragraph/3>` and all its subparts.

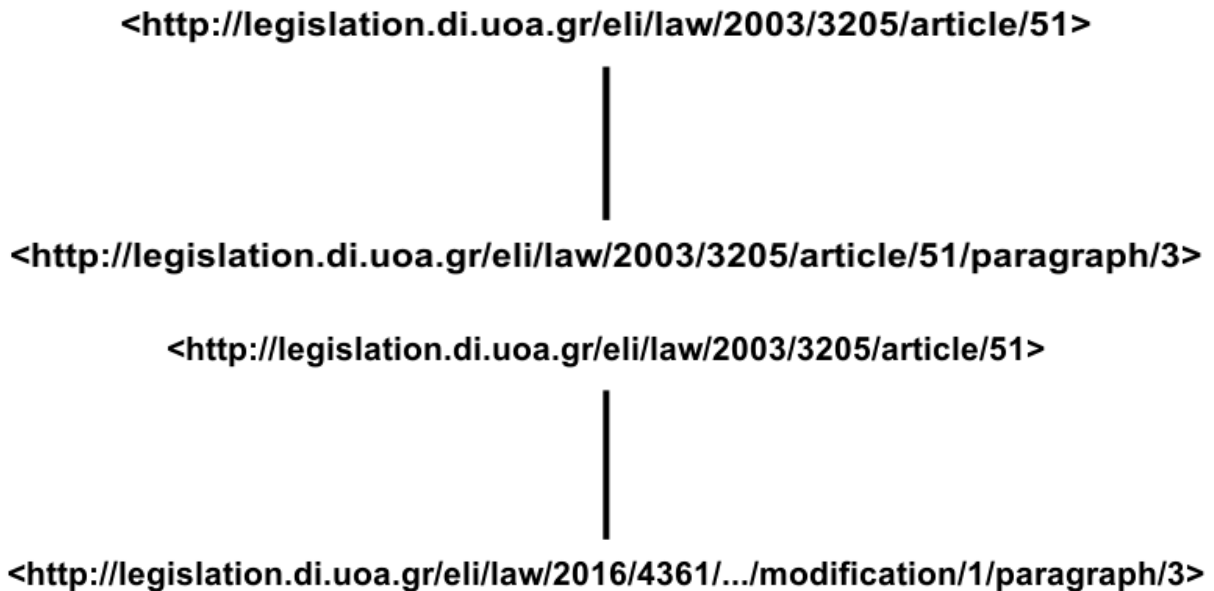


Figure 6: Legislation Tree Node before and after substitution

Being able to apply substitutions but also keep track of the older structure of each legal resource, permits a better statistical analysis over the general information concerning the Greek legislative act.

3. ENTITIES

In this chapter, we introduce a brand new concept. The idea was conceived in our effort to offer more valuable information to the user, which doesn't only specialize in legal data but also in more general knowledge. Occasionally we have found ourselves wondering “who is this President that signed all those Presidential Decrees?”, or “where is this place to which this paragraph is referred?”. To answer this kind of questions we developed the entities and a smart system that provides the information needed from inside the legal resource. At first, we will mention the types of entities that are currently supported by our application. Additionally, we will provide basic functionalities that act upon the entities and finally, we will examine through code how they work. But in the beginning, we must discuss briefly about the references, which are included in the node's attributes.

3.1 What is a reference and how it works

A reference, in legal terms, is the act of mentioning or citing one document (as a statute) in another [8]. References can be highly important to understand the meaning of the legal resource, especially in cases where the document refers to another without repeating the statements. Most commonly a reference appears as follows:

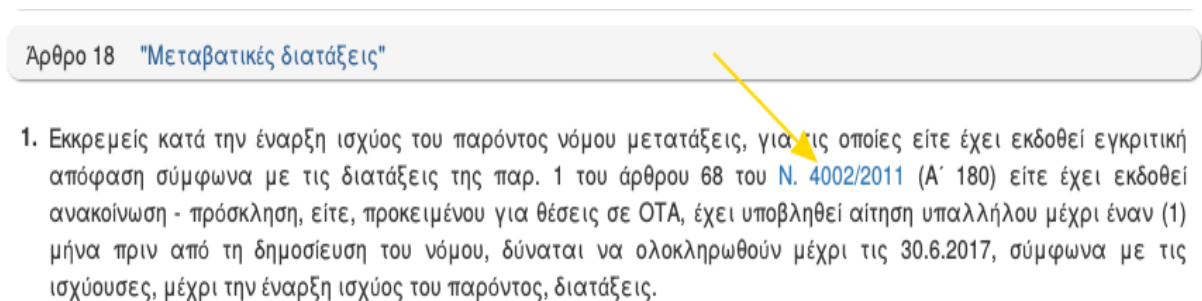


Figure 7: Reference example

In RDF format, this reference is activated with the triple below:

```
<http://legislation.di.uoa.gr/eli/law/2016/4440/chapter/1/article/18/paragraph/1/linea/1>
<http://legislation.di.uoa.gr/ontology#has_reference>
<http://legislation.di.uoa.gr/eli/law/2016/4440/chapter/1/article/18/paragraph/1/linea/1/reference/lea/1>
```

To enable a hyperlink with this reference we need to know where it starts, where it ends and it's original label. In this example case, those values are provided by the following triples:

```
<http://legislation.di.uoa.gr/eli/law/2016/4440/chapter/1/article/18/paragraph/1/linea/1/reference/leg/1>
<http://legislation.di.uoa.gr/ontology#starts>
169
```

```
<http://legislation.di.uoa.gr/eli/law/2016/4440/chapter/1/article/18/paragraph/1/linea/1/reference/leg/1>  
<http://legislation.di.uoa.gr/ontology#ends>  
181
```

```
<http://legislation.di.uoa.gr/eli/law/2016/4440/chapter/1/article/18/paragraph/1/linea/1/reference/leg/1>  
<http://legislation.di.uoa.gr/ontology#has_original_label>  
"N. 4002/2011"
```

In order to obtain this valuable information, we have developed a tool to recognize the references inside the text. Therefore, data is stored in triples and then the application uses it respectively.

3.2 Entity types

Entities become available using references. An entity can be either:

- A person (e.g. <http://legislation.di.uoa.gr/entity/person/2>)
- A landmark (e.g. <http://legislation.di.uoa.gr/entity/landmark/38>)
- A geopolitical entity (e.g. <http://legislation.di.uoa.gr/entity/gpe/5>)
- An organization (e.g. <http://legislation.di.uoa.gr/entity/org/2016_2016_18>)

Those entities are connected with references in the RDF schema. For instance:

```
<http://legislation.di.uoa.gr/eli/pd/2016/14/citation/3/reference/landmark/1>  
<http://legislation.di.uoa.gr/ontology#relevant_for>  
<http://legislation.di.uoa.gr/entity/landmark/38>
```

An entity can be found in the document by the entity recognizer and is stored in a triple in the format above. This means that this particular reference is relevant for the landmark entity with id=38. The reference keeps track of the previously mentioned values (start, end, label), with which the landmark is saved in the system.

3.2.1 Person entity

This entity stores data for a specific person. For this reason, we have linked our system with dbpedia¹. The process to retrieve information for the person we are interested in, is to connect the dbpedia page that keeps its data to the entity that our system develops. Thus, after recognizing the reference in the document and linking it with the entity, the connection occurs like in the instance below:

```
<http://legislation.di.uoa.gr/entity/person/2>  
<http://www.w3.org/2002/07/owl#sameAs>  
<http://el.dbpedia.org/resource/Προκόπης_Παυλόπουλος>
```

¹ See <http://el.dbpedia.org/>

The outcome of this specific page is the following:



Πληροφορίες οντότητας	
Όνομα:	ΠΡΟΚΟΠΗΣ ΠΑΥΛΟΠΟΥΛΟΣ
Τύπος:	http://legislation.di.uoa.gr/ontology#Person
Επιγραφή:	ΠΡΟΚΟΠΗΣ ΠΑΥΛΟΠΟΥΛΟΣ
Τόπος Γέννησης:	http://el.dbpedia.org/resource/Καλαμάτα
Πολιτικό Κόμμα:	[ΝΔ, Νέα Δημοκρατία, Ν.Δ.]
Ευρωπαϊκό Πολιτικό Κόμμα:	[Ευρωπαϊκό Λαϊκό Κόμμα, ΕΛΚ]

Figure 8: Person entity example page

3.2.2 Landmark entity

For the landmark entities, we link our system with geographical linked open data¹. For example:

```
<http://legislation.di.uoa.gr/entity/landmark/2>
<http://legislation.di.uoa.gr/ontology#belongs_to>
<http://geo.linkedopendata.gr/gag/id/9121>
```


```
<http://legislation.di.uoa.gr/entity/landmark/2>
<http://legislation.di.uoa.gr/ontology#belongs_to>
<http://legislation.di.uoa.gr/entity/gpe/9121>
```

This is beneficial because we want to create a relationship between a landmark and a gpe entity. Therefore, we notice that `<http://legislation.di.uoa.gr/entity/landmark/2>`, also belongs to a geopolitical entity developed by our system.

A sample page for the landmark appears as follows:

¹ See e.g. <http://geo.linkedopendata.gr/gag/page/id/9187>

ΟΙΚΙΣΜΟΣ ΚΟΛΥΒΑΤΩΝ



Πληροφορίες οντότητας

Όνομα:	ΟΙΚΙΣΜΟΣ ΚΟΛΥΒΑΤΩΝ
Τύπος:	http://legislation.di.uoa.gr/ontology#LocalDistrict
Ανήκει σε:	ΔΗΜΟΣ ΛΕΥΚΑΔΑΣ

Αναφέρεται Από

Τίτλος	Κωδικός	Ημερομηνία
Επανασύσταση της Ανδρώας Ιεράς Κοινοβιακής Μονής Αγίου Γεωργίου, της Ιεράς Μητροπόλεως Λευκάδος και Ιθάκης, που βρίσκεται στους πρόποδες των «Σκάρων», του οικισμού Κολυβάτων, της Τοπικής Κοινότητας Αλεξάνδρου, της Δημοτικής Κοινότητας Λευκάδας, της Δημοτικής Ενότητας Λευκάδας, του Δήμου Λευκάδας, του Νομού Λευκάδας.	Προεδρικό Διάταγμα (Π.Δ.)/71	2016

Figure 9: Landmark entity example page

3.2.3 Geopolitical entity

In this thesis, we endeavor to familiarize ourselves and the public with Geospatial Linked Data which refers to an extension of the Semantic Web in order to provide geographical information for real life entities [9].

Greek geopolitical structure is based upon “Kallikratis plan” ¹ [10]. Therefore, we have included in our dataset the file “kallikratis.ttl” which contains triples that store information according to the Law 3852/2010, that explain the architecture of the Greek local government. The main structure is available on the next table:

Table 2: Greek Local Government architecture based on "Kallikratis Plan"

Degree	Local Government Organization	Subdivision
a'	Municipality	<ul style="list-style-type: none"> Municipal Unit Community
b'	Administrative Region	Regional Unit

Apart from those subdivisions, it is worth mentioning that Greece is also divided into seven Decentralized Administrations [11], which are administration units with activities particularly in state audit and executive tasks within the area of their responsibility.

¹ See <https://www.kallikratis.org/>

Data is linked with triples like the following:

```
<http://legislation.di.uoa.gr/entity/gpe/5>  
<http://www.w3.org/2002/07/owl#sameAs>  
<http://geo.linkedopendata.gr/qaq/id/5>
```

Geopolitical entities are linked, thus, with URIs from “kallikratis.ttl”. Below (figure 10), we can observe a screenshot from the entity's view.

ΠΕΡΙΦΕΡΕΙΑ ΘΕΣΣΑΛΙΑΣ

Πληροφορίες οντότητας	
Επιγραφή:	ΠΕΡΙΦΕΡΕΙΑ ΘΕΣΣΑΛΙΑΣ
Τύπος:	http://geo.linkedopendata.gr/gag/ontology/Περιφέρεια
Επιγραφή:	ΠΕΡΙΦΕΡΕΙΑ ΘΕΣΣΑΛΙΑΣ
Πληθυσμός:	722343
Ανήκει σε:	ΑΠΟΚΕΝΤΡΩΜΕΝΗ ΔΙΟΙΚΗΣΗ ΘΕΣΣΑΛΙΑΣ-ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ

Αναφέρεται Από

Τίτλος	Κωδικός	Ημερομηνία
Τροποποιήσις της υπ' αριθμ. 61/1975 Κανονιστικής διατάξεως «Περί διοικήσεως και διαχειρίσεως του Ιερού Προσκυνημάτων Αγίας Παρασκευής Τεμπών», της Ιεράς Μητροπόλεως Λαρίσης και Τυρνάβου.	Κανονισμός (Κ.Ε.Ο.Ε.)/291_2016	2016

Figure 10: Geopolitical entity example page

3.2.4 Organization entity

Organizations are commonly used by our platform as legal document signers, along with persons and most commonly they refer to Ministries.

They are relevant for entities in the following way:

```
<http://legislation.di.uoa.gr/eli/law/2016/4363/signer/org/8>  
<http://legislation.di.uoa.gr/ontology#relevant_for>  
<http://legislation.di.uoa.gr/entity/org/2016_2016_18>
```

A screenshot from a sample page is provided:



Figure 11: Organization entity sample page

3.3 Entity Queries

1. Landmark query:

```

SELECT * WHERE{
  OPTIONAL{ < uri > rdf:type ?type. }
  OPTIONAL{
    SELECT ?label
    WHERE{
      ?ref leg:relevant_for < uri >.
      ?ref rdfs:label ?label .
    }LIMIT 1
  }
  OPTIONAL{ ?ref leg:relevant_for < uri >. ?ref rdfs:label ?reflabel. }
  OPTIONAL{ < uri > leg:belongs_to ?entity .
    ?entity rdfs:label ?entlabel.
  }
}
    
```

2. Person query:

```

SELECT *
WHERE{
OPTIONAL{ < uri > rdf:type ?type. }
OPTIONAL{
    SELECT ?label
    WHERE{
        ?ref leg:relevant_for < uri > .
        ?ref rdfs:label ?label.
    }LIMIT 1
}
OPTIONAL{
    < uri > owl:sameAs ?dbentity .
    ?dbentity rdf:type dbpedia-owl:Politician .
    OPTIONAL
        { ?dbentity rdfs:label ?dblabeled }
    OPTIONAL
        { ?dbentity dbpedia-owl:birthYear ?by }
    OPTIONAL
        { ?dbentity dbpedia-owl:birthPlace ?bp }
    OPTIONAL
        {
            ?dbentity dbpedia-owl:party ?party .
            ?party rdfs:label ?party_name .
        }
    OPTIONAL
        {
            ?party dbpedia-owl:europeanAffiliation ?affil .
            ?affil rdfs:label ?affil_name .
        }
}
OPTIONAL
{ ?dbentity foaf:depiction ?img }
OPTIONAL
{ ?dbentity dbpprop-el:μικρηΠεριγραφη ?desc }}
}

```

3. Organization query:

```

SELECT *
WHERE{
    < ministry uri > rdf:type ?type.
    ?ref leg:relevant_for < ministry uri > .
    ?ref rdfs:label ?label.
}
LIMIT 1

```

4. GPE query:

```

SELECT *
WHERE{
OPTIONAL{ < uri > rdf:type ?type. }
OPTIONAL{
  SELECT ?label
  WHERE{
    ?ref leg:relevant_for < uri >.
    ?ref rdfs:label ?label.
  }LIMIT 1
}
OPTIONAL
{ < uri > owl:sameAs ?entity .
  ?entity rdf:type ?gagtype
  OPTIONAL
  { ?entity rdfs:label ?gaglabel }
  OPTIONAL
  { ?entity gag:έχει_πληθυσμό ?gagpop }
  OPTIONAL
  { ?entity gag:ανήκει_σε ?gagbelongs }
  OPTIONAL
  { ?entity gag:έχει_γεωμετρία ?gaggeom }
  OPTIONAL
  { ?gagbelongs rdfs:label ?gagbelongslabel }
}
}

```

We can notice that both GPE and Person entity queries refer to Kallikratis and Dbpedia datasets respectively. They obtain information and finally provide it to the application user.

3.4 Entity search page

As we have already commented, the application uses a recognizer to produce a large dataset of entity triples. This large amount of knowledge needs to be available to the user in order to facilitate the way he collects important data. But, having a large dataset requires to be able to manipulate the results, so as not to get confused by unnecessary information. Therefore, the API offers a search page specifically designed for entities. The main idea behind this work had been originally implemented in the legislation search page. Entity search basically follows the same concept. There is a Lucene¹ index pointing on the dataset, a method that searches the index and a .jsp page that produces the outcome. Below (figure 12), we can see the results after demanding GPE entities.

¹ See <http://lucene.apache.org/>

Τύπος Οντότητας	Όνομα Οντότητας	Τύπος Οντότητας
<input checked="" type="checkbox"/> Γεωπολιτική Οντότητα (ΓΠΟ)	ΔΗΜΟΣ ΑΣΠΡΟΠΥΡΓΟΥ	Γεωπολιτική Οντότητα (ΓΠΟ)
<input type="checkbox"/> Πρόσωπο	ΔΗΜΟΣ ΑΣΤΥΓΙΑΛΛΙΑΣ	Γεωπολιτική Οντότητα (ΓΠΟ)
<input type="checkbox"/> Σημείο Αναφοράς	ΔΗΜΟΣ ΑΧΑΡΝΩΝ	Γεωπολιτική Οντότητα (ΓΠΟ)
<input type="checkbox"/> Οργανισμός (ΟΡΓ)	ΔΗΜΟΣ ΒΕΡΟΙΑΣ	Γεωπολιτική Οντότητα (ΓΠΟ)
	ΔΗΜΟΣ ΒΟΛΒΗΣ	Γεωπολιτική Οντότητα (ΓΠΟ)
	ΔΗΜΟΣ ΒΟΛΟΥ	Γεωπολιτική Οντότητα (ΓΠΟ)

Αναζήτηση

Figure 12: Entity search page

Concluding, we can state that entities introduce a new era for the application. Not only they offer important data to the public, but also they produce a great amount of opportunities to expand the platform into new services. Aiming to this goal, some tools have already been developed and they will be presented in the next chapter.

4. MINOR ADDITIONS AND NEW FUNCTIONALITIES

Apart from implementing the N-ary Tree to store the legal resources and referencing Entities to obtain further information, in this chapter, we will indicate various tools that we have constructed to add new functionalities to the API. Apparently, the entities form the most significant new service supported by the platform. However, noticing the previous edition we have agreed that there was a need to add new features to the project, that would arguably bring it a level further in knowledge production. We have to also note that those additions were made possible as a result of the tree implementation and the enrichment of our ontology.

To begin with, signers constitute an entity and they are no longer indicated as plain text. This may be a minor change but it has aided in creating triples and enhancing the dataset. This way, we can connect a signer with a person or organization entity, which has lead the application to perceive the signer as an object and not as a character sequence. A signer is linked with Legislation Ontology with a triple that appears like in the following example:

```
<http://legislation.di.uoa.gr/eli/amc/2016/4_22-2-2016/signer/person/1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://legislation.di.uoa.gr/ontology#Signer>
```

Additionally, we have added a table in the legal document's page, that indicates both the references it contains and the legal resources that refer to it. Below (figures 13,14) we quote two sample screenshots from this table.

Ενιαίο Σύστημα Κινητικότητας στη Δημόσια Διοίκηση και την Τοπική Αυτοδιοίκηση, υποχρεώσεις των προσώπων που διορίζονται στις θέσεις των άρθρων 6 και 8 του Ν. 4369/2016, ασυμβίβαστα και πρόληψη των περιπτώσεων σύγκρουσης συμφερόντων και λοιπές διατάξεις.

[ΠΕΡΙΕΧΟΜΕΝΑ](#)
[ΠΑΡΑΠΟΜΠΕΣ](#)
[ΧΡΟΝΟΔΙΑΓΡΑΜΜΑ](#)
[ΑΝΑΦΟΡΕΣ](#)
[ΟΝΤΟΤΗΤΕΣ](#)
[ΕΙΚΟΝΕΣ](#)

Νομικές Αναφορές
Αναφέρεται Από

Τίτλος	Κωδικός	Ημερομηνία
ΑΠΟΦΑΣΗ 2011/ΔΙΔΚ/ΚΤΠ/ΟΙΚ.21588	Απόφαση (ΑΠΟΦ.)/ΔΙΔΚ_ΚΤΠ_ΟΙΚ_21588	2011
ΑΠΟΦΑΣΗ 2015/ΔΙΔΚ/ΟΙΚ.35181	Απόφαση (ΑΠΟΦ.)/ΔΙΔΚ_ΟΙΚ_35181	2015
ΑΠΟΦΑΣΗ 2016/ΔΙΠΑΑΔ/Φ.Κ./73/ΟΙΚ.20325	Απόφαση (ΑΠΟΦ.)/ΔΙΠΑΑΔ_Φ_Κ_73_ΟΙΚ_20325	2016
ΝΟΜΟΣ 1980/1069	Νόμος (Ν.)/1069	1980

Figure 13: References found inside the document sample page

Ενιαίο Σύστημα Κινητικότητα στη Δημόσια Διοίκηση και την Τοπική Αυτοδιοίκηση, υποχρεώσεις των προσώπων που διορίζονται στις θέσεις των άρθρων 6 και 8 του Ν. 4369/2016, ασυμβίβαστα και πρόληψη των περιπτώσεων σύγκρουσης συμφερόντων και λοιπές διατάξεις.

ΠΕΡΙΕΧΟΜΕΝΑ	ΠΑΡΑΠΟΜΠΕΣ	ΧΡΟΝΟΔΙΑΓΡΑΜΜΑ	ΑΝΑΦΟΡΕΣ	ΟΝΤΟΤΗΤΕΣ	ΕΙΚΟΝΕΣ
Νομικές Αναφορές			Αναφέρεται Από		
Τίτλος	Κωδικός	Ημερομηνία			
Εθνικός Μηχανισμός Συντονισμού, Παρακολούθησης και Αξιολόγησης των Πολιτικών Κοινωνικής Ένταξης και Κοινωνικής Συνοχής, ρυθμίσεις για την κοινωνική αλληλεγγύη και εφαρμοστικές διατάξεις του ν. 4387/2016 (Α' 85) και άλλες διατάξεις.	Νόμος (Ν.)/4445	2016			
Χωρικός σχεδιασμός - Βιώσιμη ανάπτυξη και άλλες διατάξεις.	Νόμος (Ν.)/4447	2016			

Figure 14: Legal resources' references to the current legal document

The “Referenced By” tab is structured by applying the following SPARQL query:

```
SELECT DISTINCT ?legref ?reflabel ?legreftitle
WHERE{ ?ref rdf:type leg:Reference .
       ?ref rdfs:label ?reflabel .
       ?ref leg:relevant_for < uri >.
       ?part leg:has_reference ?ref.
       ?legref eli:published_in ?gaz.
       OPTIONAL{ ?legref eli:title ?legreftitle.}
       ?legref eli:has_part ?part.
}
```

Furthermore, the current API version supports statistics concerning Ministries' legislative act. The query to select this data is:

```
SELECT ?minlabel ?lawtype ?year (COUNT (?law) as ?sum)
WHERE{?law leg:published_by ?min.
      ?law eli:date_publication ?date.
      BIND (year(?date) as ?year).
      ?law rdf:type ?lawtype.
      ?min leg:relevant_for ?minent.
      ?minent rdf:type leg:Organization.
      ?min rdfs:label ?minlabel.
}GROUP BY ?minlabel ?lawtype ?year
ORDER BY ?minlabel ?lawtype ?year
```

The feature provides a graph for each ministry, indicating the amount of legal documents that were signed, sorted by year. A sample image from this graphical environment is presented below:

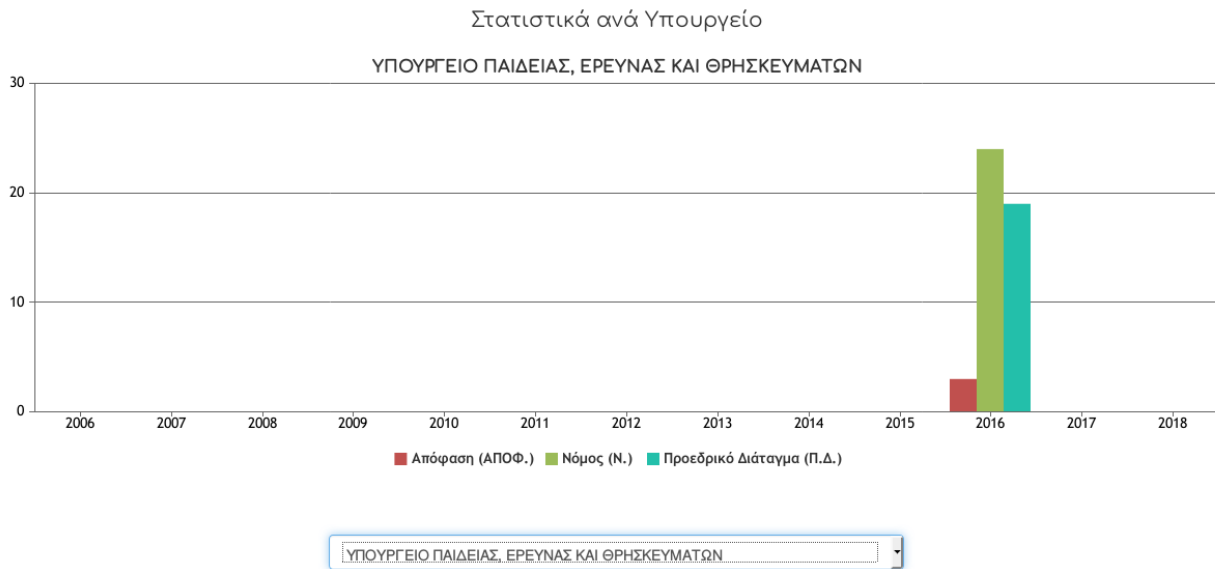


Figure 15: Ministry stats graph

Using the dropdown menu, we can choose the Ministry we are interested in.

Following our main goal to offer more data to the user, we added one more signers' table, containing the Organizations. As we have noted earlier in the thesis, a signer can be either a person or an Organization (mostly a Ministry). The query that brings the signers is almost identical to the old one, but having modified the ontology, a signer can also be a Ministry.

```

SELECT DISTINCT ?signer ?entity ?label ?type ?sameas ?dblabeled
WHERE{
  < uri > leg:published_by ?signer.
  ?signer rdfs:label ?label.
  ?signer leg:relevant_for ?entity.
  ?entity rdf:type ?type.
  OPTIONAL {
    ?entity owl:sameAs ?sameas .
    ?sameas rdfs:label ?dblabeled.
  }
} ORDER BY ?signer
    
```

The table looks like the one we use for the persons. A modification we have made, however, is that we have added links to the signers to link them to their entities' pages (see figures 8,11). More specifically:

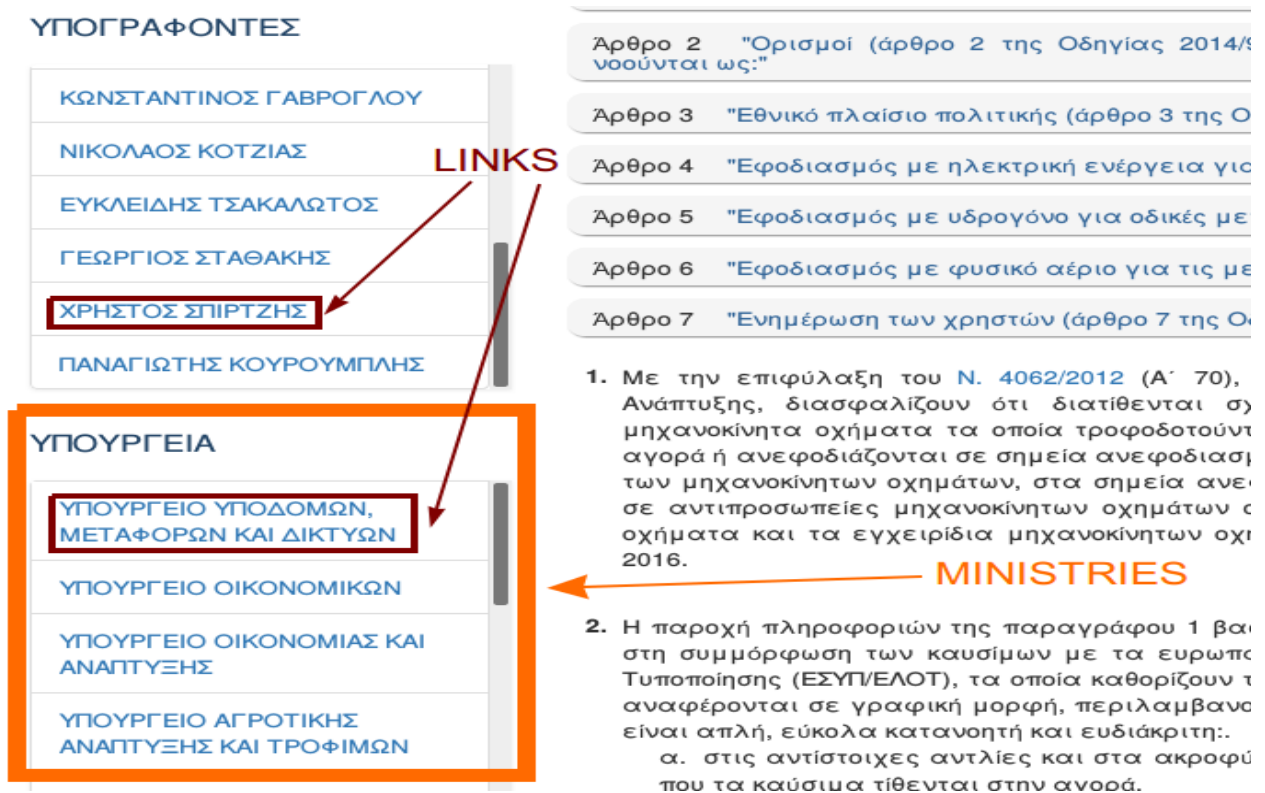


Figure 16: Ministries in signers tab (page screenshot)

Concluding, the most important functionality that has been added in the recent platform is the timeline tab. This tab contains every modification that have acted over the legal resource, indicating the document in which it has been published. The query is the following:

```

SELECT ?mod ?parent ?part ?parttype ?text ?type ?patient ?
timelinedate ?timelinetitle ?timelinegaz ?work
WHERE {
  < uri> eli:has_part ?patient.
  ?mod eli:amends ?patient.
  ?mod rdf:type ?type.
  ?mod eli:has_part ?part.
  ?parent eli:has_part ?part.
  ?part rdf:type ?parttype.
  ?work eli:has_part ?mod.
  ?work eli:date_publication ?timelinedate.
  ?work eli:published_in ?gaz.
  OPTIONAL {?gaz dc:title ?timelinegaz . }
  OPTIONAL {?work eli:title ?timelinetitle . }
  OPTIONAL {?part leg:has_text ?text . }
}
    
```

The method that deals with constructing the timeline follows this pattern:

- It creates a modification list for every legal document. This lists consists of timeline nodes (class TimelineNode).
- The first node inserted in the list is the legal document itself.
- It parses query results and constructs the timeline node.
- If a modification occurs, it inserts the timeline node into the list.

Timeline is a part of legal document's metadata and therefore is created inside the function that stores the metadata of the resource. Below (figures 16,17) we provide two screenshots as sample images of the timeline table.

Εθνικός Μηχανισμός Συντονισμού, Παρακολούθησης και Αξιολόγησης των Πολιτικών Κοινωνικής Ένταξης και Κοινωνικής Συνοχής, ρυθμίσεις για την κοινωνική αλληλεγγύη και εφαρμοστικές διατάξεις του ν. 4387/2016 (Α' 85) και άλλες διατάξεις.

ΠΕΡΙΕΧΟΜΕΝΑ	ΠΑΡΑΠΟΜΠΕΣ	ΧΡΟΝΟΔΙΑΓΡΑΜΜΑ	ΑΝΑΦΟΡΕΣ	ΟΝΤΟΤΗΤΕΣ	ΕΙΚΟΝΕΣ			
Ημερομηνία	Τίτλος	ΦΕΚ						
2016-12-19	Εθνικός Μηχανισμός Συντονισμού, Παρακολούθησης και Αξιολόγησης των Πολιτικών Κοινωνικής Ένταξης και Κοινωνικής Συνοχής, ρυθμίσεις για την κοινωνική αλληλεγγύη και εφαρμοστικές διατάξεις του ν. 4387/2016 (Α' 85) και άλλες διατάξεις.	A/2016/236						
2016-12-23	Χωρικός σχεδιασμός - Βιώσιμη ανάπτυξη και άλλες διατάξεις. ⓘ	A/2016/241						
	<table border="1"> <thead> <tr> <th>Τροποποίηση</th> <th>Τύπος</th> </tr> </thead> <tbody> <tr> <td>Η εφάπαξ οικονομική ενίσχυση της παραγράφου 1 του παρόντος άρθρου είναι αφορολόγητη, δεν υπόκειται σε οποιαδήποτε κράτηση, δεν κατάσχεται ούτε συμψηφίζεται με ήδη βεβαιωμένα χρήη προς το Δημόσιο ή πιστωτικά ιδρύματα και δεν υπολογίζεται στα εισοδηματικά όρια για την καταβολή οποιουδήποτε επιδόματος ή παροχής κοινωνικού ή προνοιακού χαρακτήρα.</td> <td>Προσθήκη</td> </tr> </tbody> </table>	Τροποποίηση	Τύπος	Η εφάπαξ οικονομική ενίσχυση της παραγράφου 1 του παρόντος άρθρου είναι αφορολόγητη, δεν υπόκειται σε οποιαδήποτε κράτηση, δεν κατάσχεται ούτε συμψηφίζεται με ήδη βεβαιωμένα χρήη προς το Δημόσιο ή πιστωτικά ιδρύματα και δεν υπολογίζεται στα εισοδηματικά όρια για την καταβολή οποιουδήποτε επιδόματος ή παροχής κοινωνικού ή προνοιακού χαρακτήρα.	Προσθήκη			
Τροποποίηση	Τύπος							
Η εφάπαξ οικονομική ενίσχυση της παραγράφου 1 του παρόντος άρθρου είναι αφορολόγητη, δεν υπόκειται σε οποιαδήποτε κράτηση, δεν κατάσχεται ούτε συμψηφίζεται με ήδη βεβαιωμένα χρήη προς το Δημόσιο ή πιστωτικά ιδρύματα και δεν υπολογίζεται στα εισοδηματικά όρια για την καταβολή οποιουδήποτε επιδόματος ή παροχής κοινωνικού ή προνοιακού χαρακτήρα.	Προσθήκη							

Figure 17: Insertion in timeline

Εθνικό Μητρώο Επιτελικών Στελεχών Δημόσιας Διοίκησης, βαθμολογική διάρθρωση θέσει συστήματα αξιολόγησης, προαγωγών και επιλογής προϊστάμενων (διαφάνεια - αξιοκρατία και αποτελεσματικότητα της Δημόσιας Διοίκησης) και άλλες διατάξεις.

ΠΕΡΙΕΧΟΜΕΝΑ	ΠΑΡΑΠΟΜΠΕΣ	ΧΡΟΝΟΔΙΑΓΡΑΜΜΑ	ΑΝΑΦΟΡΕΣ	ΟΝΤΟΤΗΤΕΣ	ΕΙΚΟΝΕΣ			
Ημερομηνία	Τίτλος	ΦΕΚ						
2016-02-27	Εθνικό Μητρώο Επιτελικών Στελεχών Δημόσιας Διοίκησης, βαθμολογική διάρθρωση θέσεων, συστήματα αξιολόγησης, προαγωγών και επιλογής προϊστάμενων (διαφάνεια - αξιοκρατία και αποτελεσματικότητα της Δημόσιας Διοίκησης) και άλλες διατάξεις.	A/201						
2016-05-27	Επείγουσες διατάξεις για την εφαρμογή της συμφωνίας δημοσιονομικών στόχων και διαρθρωτικών μεταρρυθμίσεων και άλλες διατάξεις. ⓘ	A/201						
2016-03-07	Συστήματα Εγγύησης Καταθέσεων (ενσωμάτωση Οδηγίας 2014/49/ΕΕ), Ταμείο Εγγύησης Καταθέσεων και Επενδύσεων και άλλες διατάξεις. ⓘ	A/201						
	<table border="1"> <thead> <tr> <th>Τροποποίηση</th> <th>Τύπος</th> </tr> </thead> <tbody> <tr> <td>Η ισχύς της διάταξης της παρ. 1 του άρθρου 72 του ν. 4342/2015 (Α' 143) παρατείνεται έως την 28η Απριλίου 2016 .</td> <td>Αντικατάσταση</td> </tr> </tbody> </table>	Τροποποίηση	Τύπος	Η ισχύς της διάταξης της παρ. 1 του άρθρου 72 του ν. 4342/2015 (Α' 143) παρατείνεται έως την 28η Απριλίου 2016 .	Αντικατάσταση			
Τροποποίηση	Τύπος							
Η ισχύς της διάταξης της παρ. 1 του άρθρου 72 του ν. 4342/2015 (Α' 143) παρατείνεται έως την 28η Απριλίου 2016 .	Αντικατάσταση							

Figure 18: Substitution in timeline

5. GENERAL MODIFICATIONS

In the fifth chapter of this thesis, we will be citing some minor but important modifications we have applied to the previous edition of the API. Those changes became necessary either because some functionalities didn't work properly or because we discovered more efficient ways to implement them.

Firstly, we coded the PDF, XML, RDF and JSON builders. All four of them didn't function properly because it used to be inefficient to structure the document. However, in the current edition, the PDF and XML files are built using the same logic that is used in the case of the .jsp pages, utilizing the facilities offered by the tree structure. On the contrary, JSON is built converting straight from XML to JSON and the RDF file is produced by a describe query.

Table 3: Libraries used for the builders

PDF	iText 7.0.3 ¹
XML	dom4j 1.6.1 ²

The PDF now looks perfect, whereas, in the old edition it used to be mixed up. An example part of the PDF appears as follows:

ΚΕΦΑΛΑΙΟ ΣΤ'

ΑΠΛΟΥΣΤΕΥΣΗ ΠΛΑΙΣΙΟΥ ΑΣΚΗΣΗΣ ΜΕΤΑΠΟΙΗΤΙΚΩΝ ΚΑΙ ΣΥΝΑΦΩΝ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ
ΤΡΟΦΙΜΩΝ ΚΑΙ ΠΟΤΩΝ

Άρθρο 17

Πεδίο εφαρμογής

1. Στο πεδίο εφαρμογής του παρόντος Κεφαλαίου εμπίπτουν οι δραστηριότητες μεταποίησης τροφίμων και ποτών με Κωδικούς Αριθμούς Δραστηριότητας (ΚΑΔ) 10 και 11 που περιλαμβάνονται στην 3η ομάδα του Παραρτήματος, καθώς και οι «δραστηριότητες συσκευασίας» με ΚΑΔ 82.92 της 11ης ομάδας στις περιπτώσεις που αφορούν τρόφιμα και ποτά.

Figure 19: PDF file sample

¹ See <https://itextpdf.com>

² See <https://dom4j.github.io/>

Moreover to the builders, we realized that general statistics concerning the Greek legislative act should be provided dynamically. Consequently, we have used the library `canvas.js`³ to offer statistics that user can define by scrolling a scrollbar to choose the date range. Additionally, the user can select the legal resource type, as long as the Ministry he wishes (see figure 15). In the example screenshot below (figure 20), we can mention that we have diminished the date range between 2015 and 2018, but the dataset we have used is for testing only, so it contains data from various years and, in this range, it contains data only from 2016.

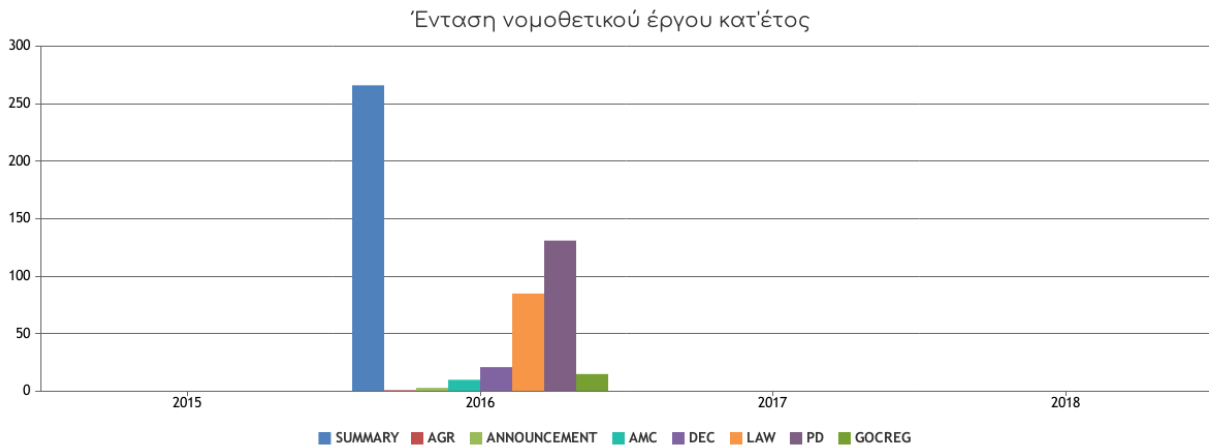


Figure 20: Statistics graph sample

One more change that is worth being noticed is the fact that we have improved the functionality and the appearance of the legal substitution modifications inside the document. More specifically, the old part is contained inside a red area, while the newly added is part of a large green container, like in the image that follows:

9. Αν Κοιν.Σ.Επ. ή Συνεταιρισμός Εργαζομένων διαγραφούν από το Μητρώο με πρωτοβουλία της Διοίκησης, σύμφωνα με τις προβλέψεις του παρόντος άρθρου, οφείλουν εντός ενενήντα (90) ημερών από την κοινοποίηση σε αυτούς της πράξης διαγραφής, να ενεργήσουν τη νόμιμη μετατροπή της Κοιν.Σ.Επ. ή του Συνεταιρισμού Εργαζομένων σε μορφή επιχείρησης που τα μέλη επιθυμούν ή να προβούν στη λύση της. Σε κάθε περίπτωση, μετά τη λύση ή πριν τη μετατροπή σε άλλη νομική μορφή, τίθεται σε εκκαθάριση, σύμφωνα με το άρθρο 22 του παρόντος. Οι ανωτέρω ενέργειες οφείλουν να γνωστοποιούνται από την Κοιν.Σ.Επ. ή το Συνεταιρισμό Εργαζομένων στο Μητρώο Φορέων Κοινωνικής και Αλληλέγγυας Οικονομίας.

↔

Αν Κοιν.Σ.Επ. ή Συνεταιρισμός Εργαζομένων διαγραφούν από το Μητρώο για οποιονδήποτε λόγο, οφείλουν εντός τριάντα (30) ημερών από την κοινοποίηση σε αυτούς της πράξης διαγραφής, να ενεργήσουν τη νόμιμη μετατροπή τους σε μορφή επιχείρησης που τα μέλη επιθυμούν και να καταχωρήσουν τη μετατροπή αυτή ή να προβούν σε διακοπή εργασιών στην αρμόδια ΔΟΥ.

Οι ανωτέρω ενέργειες πρέπει να γνωστοποιούνται στο Μητρώο.

Άρθρο 12 "Εθνική Επιτροπή για την Κοινωνική και Αλληλέγγυα Οικονομία"

Figure 21: Modification appearance in the document

³ See <https://canvasjs.com/>

As far as anyone can notice, there is a small blue icon that opens the deleted part when clicked.

Apart from the statistics, we have also ameliorated the appearance of the legal document in our main legislation web page. The main structure remains the same, but we have added some functionalities that have already been noted. An image is provided below to show the legal resource presentation page:

The screenshot displays a web page for a legal document titled "Ρύθμιση του επαγγέλματος του διαχειριστή αφερεγγυότητας". The page is structured as follows:

- GENIKA STOICHEIA:**
 - TYPOS:** Προεδρικό Διάταγμα (Π.Δ.)
 - KWΔIKOS:** ΦΕΚ
 - 2016/133:** A/2016/242
 - HMEPOMHNIΑ:** 2016-12-29
 - HMEPOMHNIΑ EPAPMOΓHΣ:** 2016-12-29
 - HMEPOMHNIΑ YΠOΓPAΦHΣ:** 2016-12-20
- YΠOΓPAΦONTEΣ:**
 - ΠΡΟΚΟΠΙΟΣ ΠΑΥΛΟΠΟΥΛΟΣ
 - ΔΗΜΟΣ ΠΑΠΑΔΗΜΗΤΡΙΟΥ
 - ΣΤΑΥΡΟΣ ΚΟΝΤΟΝΗΣ
- YΠOYΡΓEΙΑ:**
 - ΥΠΟΥΡΓΕΙΟ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΑΝΑΠΤΥΞΗΣ
- Navigation Tabs:** ΠΕΡΙΕΧΟΜΕΝΑ, ΠΑΡΑΠΟΜΠΕΣ, ΧΡΟΝΟΔΙΑΓΡΑΜΜΑ, ΑΝΑΦΟΡΕΣ, ΟΝΤΟΤΗΤΕΣ, ΕΙΚΟΝΕΣ
- Content:**
 - ΜΕΡΟΣ ΠΡΩΤΟ**
ΓΕΝΙΚΕΣ ΔΙΑΤΑΞΕΙΣ
 - Άρθρο 1 "Σκοπός"**
 - Άρθρο 2 "Ορισμοί"**
 - Άρθρο 3 "Πεδίο εφαρμογής"**
 - Το παρόν εφαρμόζεται στις διαδικασίες αφερεγγυότητας, όπως αυτές ορίζονται στο άρθρο 2 του παρόντος και για τα ειδικότερα θέματα τα οποία ρυθμίζει
 - Εξαιρούνται από το πεδίο εφαρμογής του παρόντος οι διαδικασίες αφερεγγυότητας σχετικά με τις ασφαλιστικές επιχειρήσεις και τα πιστωτικά ιδρύματα, τις επιχειρήσεις επενδύσεων τις παρέχουσες υπηρεσίες που συνεπάγονται κατοχή κεφαλαίων ή κινητών αξιών τρίτων, καθώς και τους οργανισμούς συλλογικών επενδύσεων. Ως προς τις ως άνω διαδικασίες αφερεγγυότητας ισχύουν οι ειδικές ρυθμίσεις που προβλέπονται αντίστοιχα από τις διατάξεις των άρθρων 220 επ. του ν. 4364/2016 (Α' 13), 1 επ., 44, 56, 145 επ. του ν. 4261/2014 (Α' 107) και 1 επ., 22 επ., 50 του ν. 3606/2007 (Α' 245) και τους κανόνες του ενωσιακού δικαίου.
 - Άρθρο 4 "Επιτροπή Διαχείρισης Αφερεγγυότητας"**

Figure 22: Legal document main web page

Additionally, we have fixed the search page to produce the correct results. The philosophy behind the page construction remained the same and no technologies were changed since the last update. Page structure is the same, so the main alteration is the repair of the search by date which didn't work properly. In the screenshot below (illustration 23), we can observe the search results when date range is confined between 2016-09-07 and 2016-09-09.

6. CONCLUSION

The idea to bring the public closer to the Greek legislative act seemed more topical than ever. Nowadays, people use technology for the majority of their daily tasks. Therefore, we figured it would be rational to facilitate people's lives, helping them to understand our country's legislation utilizing the web and other technological tools. It is a matter of vital importance for people, not only to know their rights but also to be aware of their obligations. This application aims to reach each and every citizen of Greece and it is not specifically designed for lawyers and judges, because it is a consequence of a well informed public.

Our mission is to publish Greek legislation data electronically as a part of Linked Data, following the example of other countries like the UK¹ [12] and the Netherlands². Open Data is massively grown over the last years. It is our major belief that data should be available to the public opinion for use or simple knowledge. Our country has done significant steps on this road to release data to the public. Developing Diavgeia³ has been the fundamental effort to achieve transparency by the mandatory disclosure of all government's decisions and acts. Diavgeia is a portal that aids public administration, in which government's decisions are modeled in OWL and queried using SPARQL [13]. Thus, having Diavgeia as a motive, we have realized it is crucial to develop an application that will totally make good use of open data.

In this thesis, we indicated and explained the updates that we have applied on the previous edition of Nomothesi@ API. Our task was to radically modify and improve the platform to guarantee its proper and efficient work. The changes referred to altering the storing system of the legal parts and inserting new functionalities to the system, fixing minor issues that had occurred in the old web application. We believe that we have achieved a significant level of efficiency and reliability using the tree structure to link the legal document's components. Additionally, entities introduction has been a major extension that allowed us to offer information that is not limited to legislation. We have expanded knowledge production using open data from large dataset sources like dbpedia. Finally, we have fixed various issues that complicated the use of the application and have developed minor additional features to improve the general function of the platform.

Concluding, we must note that the main idea behind our project belongs to a larger concept to bring free data to the public. Data is information and information is knowledge. Knowledge refers to a theoretical or practical understanding of a subject. Being able to understand various subjects leads to a society of justice, trust, peace, altruism, respect, freedom, solidarity and many more values and principles.

¹ See <https://data.gov.uk>

² See <https://data.overheid.nl>

³ See <https://diavgeia.gov.gr/>

ABBREVIATIONS – ACRONYMS

Table 4: Table of abbreviations

API	Application Programming Interface
CEN	European Committee for Standardization
CSS	Cascading Style Sheets
DFS	Depth-First Search
ELI	European Legislative Identifier
GPE	Geo-Political Entity
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
PDF	Portable Document Format
OWL	Web Ontology Language
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
URI	Universal Resource Identifier
XML	Extensible Markup Language

TECHNOLOGIES – LIBRARIES USED

Nomothesi@ API is implemented in Spring Web MVC Framework¹. The Model – View – Controller framework [14] supports application division in three components. The model contains the functionalities and data of the used classes. The controller handles user input and the view displays data to the user. Therefore, an MVC framework separates the UI (controller and view) from the data (model), simultaneously providing consistency between those three components.

Table 5: Libraries used in the project

Java	8 (or greater)
Spring Framework	4.2.7.RELEASE
Apache Maven	4.0.0
Apache Tomcat	7.0.55
Eclipse RDF4J	2.3.0
Apache Lucene	5.5.4
iText PDF	7.0.3
dom4j	1.6.1
Bootstrap	3.3.1
jQuery	1.8 & 1.11.1
jQuery DataTable	1.10.15
jQuery Calendar	1.12.4
CanvasJS	2.1
OpenLayers	4.6.5

¹ See <https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html>

REFERENCES

- [1] Chalkidis I. (2014). Nomothesi@: Greek Legislation Platform. B.Sc Thesis. National and Kapodistrian University of Athens.
- [2] Soursos P. (2015). Nomothesi@ API: Re-engineering the Electronic Platform. B.Sc Thesis. National and Kapodistrian University of Athens.
- [3] ELI Task Force (2015). [ELI A Technical Implementation Guide](#). Publications Office of the European Union.
- [4] European Commission. (2016). [eGovernment in the European Union](#).
- [5] Lodder A.R, Oskamp A. (eds. 2006). Information Technology & Lawyers. Springer. Chap. 1 & 7.
- [6] Tim Berners-Lee. (2006). [Linked Data. Design Issues](#).
- [7] Chalkidis I., Nikolaou C., Soursos P., Koubarakis M. (2017). [Modeling and Querying Greek Legislation using Semantic Web Technologies](#). National and Kapodistrian University of Athens.
- [8] Reference [Def.1]. [Merriam-Webster Online](#).
- [9] Karalis A. (2012). [Greek Linked Data Applications for Smartphones using Geospatial Data](#). Aristotle University of Thessaloniki.
- [10] N. 3852/2010. [Νέα Αρχιτεκτονική της Αυτοδιοίκησης και της Αποκεντρωμένης Διοίκησης – Πρόγραμμα Καλλικράτης](#). ΦΕΚ 87/Α/7-6-2010.
- [11] Council of Europe. (2012). [Structure and Operation of Local and Regional Democracy](#).
- [12] Sheridan J. (2010). [Legislation.gov.uk](#) .
- [13] Beris T., Koubarakis M. (2018). [Modeling and Preserving Greek Government Decisions using Semantic Web Technologies and Permissionless Blockchains](#).
- [14] Sarker I., Apu K. (2014). [MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application](#). International Journal of Information Technology.