Logo, company name

Description automatically generated

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΤΕΥΘΥΝΣΗ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

**Πτυχιακή Εργασία**

**CPSV-AP API: Ένας σύγχρονος τρόπος ανταλλαγής δεδομένων για τις Δημόσιες Υπηρεσίες**

Της

**ΜΑΚΡΙΔΟΥ ΗΛΕΚΤΡΑΣ**

Σεπτέμβριος 2021

**Αφιερώσεις**

Στην οικογένεια μου

**Ευχαριστίες**

Θα ήθελα αρχικά να ευχαριστήσω τον καθηγητή μου Ευθύμιο Ταμπούρη για την ευκαιρία που μου έδωσε να αναπτύξω αυτή την εργασία και τις συμβουλές που μου παρείχε κατά την διάρκεια εκπόνησής της.

Επίσης θα ήθελα να ευχαριστήσω τον συνάδελφο μου Νέστωρα Μιαρίτη για τα εύστοχα σχόλια του όσο αναφορά την ανάπτυξη του κώδικα της παρούσας εργασίας.

**Περίληψη (Abstract)**

Η Ευρωπαϊκή Ένωση τα τελευταία χρόνια έχει επενδύσει σημαντικά στον εκσυγχρονισμό των δημόσιων διοικήσεων της ώστε να παρέχονται πιο αποτελεσματικές, ανοικτές και διαφανείς υπηρεσίες προς τους πολίτες και τις επιχειρήσεις της. Μια από της σημαντικότερες προτάσεις της είναι η υιοθέτηση των Core Vocabularies για την περιγραφή των πληροφοριών που συλλέγουν όλες οι δημόσιες διοικήσεις, ώστε να μπορέσει να υλοποιηθεί η διαλειτουργικότητα ανάμεσα στα συστήματα όλης της Ευρωπαϊκής Ένωσης. Ένα από αυτά τα μοντέλα δεδομένων είναι και το CPSV-AP που παρέχει πληροφορίες σχετικά με τις δημόσιες υπηρεσίες κάθε οργανισμού σε μορφή Ανοικτών Συνδεδεμένων Δεδομένων. Σε αυτή την μορφή, οι πληροφορίες που συλλέγουν οι δημόσιες διοικήσεις μπορούν να αξιοποιηθούν και από τις ίδιες και από τρίτους για την δημιουργία χρήσιμων εφαρμογών. Παραδείγματος χάρη, η δημιουργία ενός REST API με βάση τα δεδομένα αυτά μπορεί βοηθήσει στην παροχή καλύτερης πληροφόρησης στους πολίτες και στην δημιουργία άλλων εφαρμογών που τους υποστηρίζουν στην καθημερινές του συναλλαγές με τον δημόσιο τομέα. Η δημιουργία ενός API είναι πλέον αρκετά εύκολη, καθώς το οικοσύστημα τους έχει γνωρίσει τεράστια άνθηση τα τελευταία χρόνια. Αυτό έχει και ως αποτέλεσμα, την παροχή πολλών εργαλείων και υποστηρικτού υλικού για την δημιουργία REST APIs που συνδράμουν στην περαιτέρω ανάπτυξη της «Οικονομίας των API», όπου πλέον οι φορείς μπορούν να επωφεληθούν από την χρήση των δεδομένων που συλλέγουν, προωθώντας τα προς τρίτους.

**Πίνακας περιεχομένων**

[**Πίνακας των εικονογραφήσεων** 7](#_Toc82683452)

[**Κατάλογος πινάκων** 7](#_Toc82683453)

[1. Εισαγωγή 8](#_Toc82683454)

[1.1 Περιγραφή του προβλήματος 8](#_Toc82683455)

[1.2 Αντικείμενο και στόχοι της μελέτης 11](#_Toc82683456)

[1.3 Περιεχόμενα της μελέτης 11](#_Toc82683457)

[2. Βιβλιογραφική Ανασκόπηση 13](#_Toc82683458)

[2.1 Ηλεκτρονική Διακυβέρνηση 13](#_Toc82683459)

[2.1.1 Η Ηλεκτρονική Διακυβέρνηση στην Ευρωπαϊκή Ένωση 14](#_Toc82683460)

[2.1.2 Βασικές Αρχές του Πλάνου Δράσης Ηλεκτρονικής Διακυβέρνησης 16](#_Toc82683461)

[2.2 Διαλειτουργικότητα 17](#_Toc82683462)

[2.2.1 ISA 19](#_Toc82683463)

[2.2.2 Core Vocabularies 19](#_Toc82683464)

[2.3 CPSV – AP 20](#_Toc82683465)

[2.3.1 Η λύση του CPSV-AP 21](#_Toc82683466)

[2.3.2 Οι κλάσεις του CPSV-AP 22](#_Toc82683467)

[2.4 APIS 25](#_Toc82683468)

[2.4.1 Η αρχιτεκτονική REST 26](#_Toc82683469)

[2.4.2 REST APIS 27](#_Toc82683470)

[2.4.3 API ECONOMY 28](#_Toc82683471)

[2.5 Συμπεράσματα 29](#_Toc82683472)

[3.Μεθοδολογία 30](#_Toc82683473)

[3.1 Τα εργαλεία ανάπτυξης του CPSV-AP REST API 30](#_Toc82683474)

[3.1.1 C# 31](#_Toc82683475)

[3.1.2 SPARQL και Ανοικτά Συνδεδεμένα Δεδομένα 32](#_Toc82683476)

[3.1.4 dotNetRDF 32](#_Toc82683477)

[3.2 O σχεδιασμός ενός API 33](#_Toc82683478)

[3.2.1 Το μοντέλο ανάπτυξης MVC 34](#_Toc82683479)

[3.3 Συμπεράσματα 35](#_Toc82683480)

[4.CPSV-AP API 36](#_Toc82683481)

[4.1 Η δομή του project CPSV-AP API 37](#_Toc82683482)

[4.2 Δημοσιεύοντας το CPSV-API 52](#_Toc82683483)

[4.2.1 HEROKU 53](#_Toc82683484)

[4.2.2 GitHub 58](#_Toc82683485)

[4.3 Δουλεύοντας με το CPSV-API 58](#_Toc82683486)

[4.3 Συμπεράσματα 63](#_Toc82683487)

[5. Συμπεράσματα 64](#_Toc82683488)

[5.1 Δράσεις της Ευρωπαϊκής Ένωσης 64](#_Toc82683489)

[5.2 Περιορισμοί στο API 64](#_Toc82683490)

[5.3 Βελτιώσεις στο API 65](#_Toc82683491)

[5.4 Προτάσεις 66](#_Toc82683492)

[6. ΒΙΒΛΙΟΓΡΑΦΙΑ 67](#_Toc82683493)

[6.1 Επίσημα κείμενα στο διαδίκτυο 67](#_Toc82683494)

[6.2 Άρθρα 67](#_Toc82683495)

[6.3 Δημοσιευμένες Εργασίες 68](#_Toc82683496)

[6.4 Βιβλία 68](#_Toc82683497)

[6.5 Ιστοσελίδες 69](#_Toc82683498)

[ΠΑΡΑΡΤΗΜΑΤΑ 70](#_Toc82683499)

[Α. Χρήσιμα links 70](#_Toc82683500)

## Πίνακας των εικονογραφήσεων

[**Εικόνα 1** Βασικοί Πυλώνες δράσεων του Πλάνου ΗΔ για το 2016-2020 15](file:///C:\Users\ilektra.makridou\Desktop\thesis%202%20-%20Copy.docx#_Toc82679525)

[**Εικόνα 2** Τα είδη της Διαλειτουργικότητας κατά τον EIF 18](#_Toc82679526)

[**Εικόνα 3** Οι λύσεις του προγράμματος ISA 19](#_Toc82679527)

[**Εικόνα 4** To πλήρες μοντέλο του CPSV-AP 22](file:///C:\Users\ilektra.makridou\Desktop\thesis%202%20-%20Copy.docx#_Toc82679528)

[**Εικόνα 5** Οι μέθοδοι REST όπως εμφανίζονται σε ένα REST API 27](file:///C:\Users\ilektra.makridou\Desktop\thesis%202%20-%20Copy.docx#_Toc82679529)

[**Εικόνα 6** Το μοντέλο MVC όπως το προτείνει το REST API Template της Microsoft 34](#_Toc82679530)

[**Εικόνα 7** Heroku Dashboard 53](#_Toc82679531)

[**Εικόνα 8** Heroku Create New App First Screen 54](#_Toc82679532)

[**Εικόνα 9** Οθόνη διαχείρισης μιας εφαρμογής στο Heroku 55](#_Toc82679533)

[**Εικόνα 10** Προσθήκη κώδικα από το Github 55](#_Toc82679534)

[**Εικόνα 11** Αποτελέσματα αναζήτησης repository 55](#_Toc82679535)

[**Εικόνα 12** Προσθήκη επιπλέον buildpacks σε μια εφαρμογή 56](#_Toc82679536)

[**Εικόνα 13** Διαχείριση πρόσβασης στην εφαρμογή 57](#_Toc82679537)

[**Εικόνα 14** Το βασικό παράθυρο εργασίας στο Postman 58](file:///C:\Users\ilektra.makridou\Desktop\thesis%202%20-%20Copy.docx#_Toc82679538)

[**Εικόνα 15** Αποτελέσματα κλήσης στο Postman 59](#_Toc82679539)

[**Εικόνα 16** Επιλογές μορφοποίησης αποτελεσμάτων 59](#_Toc82679540)

[**Εικόνα 17** Το περιβάλλον εργασίας του Virtuoso 60](#_Toc82679541)

[**Εικόνα 18** Αποτελέσματα ενός query στο Virtuoso 61](#_Toc82679542)

**Κατάλογος πινάκων**

[**Πίνακας 1** Ονομασίες κλάσεων του CPSV-AP μαζί με τα URI τους 24](#_Toc82677084)

# Εισαγωγή

## 1.1 Περιγραφή του προβλήματος

Η πρόοδος των Τεχνολογιών Πληροφορίας και Επικοινωνίας (ΤΠΕ),  μόνο την τελευταία δεκαετία, έχει δημιουργήσει πληθώρα νέων τεχνολογιών, εφαρμογών και εργαλείων, μετασχηματίζοντας αρκετές πτυχές της ανθρώπινης δραστηριότητας. Τεχνολογίες αυτοματισμού, μεταφοράς, επεξεργασίας και ανάλυσης μεγάλου όγκου δεδομένων καθώς και η τεράστια άνοδος στην υπολογιστική ισχύ των μηχανημάτων έχουν μετασχηματίσει ποικίλες πτυχές της ανθρώπινης δραστηριότητας. Όλοι οι τομείς παραγωγής έχουν πλέον εντάξει, λίγο ή πολύ, τις ΤΠΕ στην καθημερινότητα τους για να αυξήσουν την παραγωγικότητα τους και να μειώσουν τα κόστη τους. Ειδικότερα, τις τελευταίες δύο δεκαετίες, πολλές επιχειρήσεις του τριτογενή τομέα παραγωγής έχουν ιδρυθεί και εξελιχθεί βασιζόμενες σε καινοτόμα προϊόντα των ΤΠΕ. Εταιρείες όπως η Google και η Facebook έχουν καταφέρει μέσω των καινοτόμων  υπηρεσιών τους να δημιουργήσουν νέες επικερδής αγορές. Ενώ εταιρείες όπως η Amazon και η TESLA έχουν μετασχηματίσει εντελώς τον κλάδο τους, βασιζόμενες εντελώς στις δυνατότητες των σύγχρονων ΤΠΕ.  
Σε αυτό το πλαίσιο καινοτομίας και εξέλιξης, οι περισσότεροι άνθρωποι πλέον (ιδίως στις πιο ανεπτυγμένες οικονομίες) έχουν δει την καθημερινότητα τους να αλλάζει, με τις ΤΠΕ να κατέχουν καθοριστικό ρόλο τόσο στην εργασιακή όσο και στην προσωπική ζωή τους. Η τεράστια υπολογιστική ισχύ τόσο των υπολογιστών όσο και των κινητών τηλεφώνων των τελευταίων ετών σε συνδυασμό με τις προσιτές για τον μέσο καταναλωτή τιμές, έχουν δημιουργήσει ένα περιβάλλον φιλικό για την εξοικείωση και την ενασχόληση με τις νεότερες τεχνολογίες, ειδικότερα στις νεότερες γενιές που πλέον θεωρούνται “digital natives”. Απόρροια αυτού του γεγονότος είναι και η τάση πλέον πολλών ανθρώπων να προτιμούν τις ψηφιακές υπηρεσίες έναντι των παραδοσιακών ως πιο αποτελεσματικές και αξιόπιστες.   
Μάλιστα τόσο μεγάλη θεωρείται η συμβολή των ΤΠΕ στις σύγχρονες κοινωνίες που αρκετοί ακαδημαϊκοί και επιχειρηματίες της δώσει  έχουν δώσει το όνομα 4η Βιομηχανική Επανάσταση. Έχοντας ως επίκεντρο την γρήγορη μεταφορά τεράστιου όγκου δεδομένων μέσω των νέων δικτύων (5G, οπτικές ίνες), την Μηχανική Μάθηση, της τεχνολογίες Νέφους και άλλες τεχνολογίες αιχμής, η 4η Βιομηχανική Επανάσταση υπόσχεται πιο αξιόπιστη, πλήρη και χρήσιμη πληροφόρηση σε σχέση με τα παλαιότερα μέσα, βασιζόμενη σε αξιοποιήσιμα δεδομένα πολλαπλών πηγών. Τα δεδομένα που συλλέγονται παύουν πλέον να έχουν αξία μόνο εντός του οργανισμού που τα συλλέγει, αντιθέτως φαίνεται πως εξελίσσονται σε ένα νέο πολύτιμο πόρο τόσο για τις επιχειρήσεις όσο και για τον άνθρωπο. Απόδειξη της αξίας που έχουν αποκτήσει τα δεδομένα ως πηγή κέρδους αποτελεί και η ξαφνική διάκριση της Επιστήμης των Δεδομένων ως ξεχωριστό κλάδο στον τομέα της Πληροφορικής, κερδίζοντας τόσο το ενδιαφέρον των ακαδημαϊκών αλλά και της ίδιας της αγοράς.  
Σε αυτό όμως το ταχεία αναπτυσσόμενο κόσμο, αποδείχθηκε πως οι ευρωπαϊκές κυβερνήσεις δυσκολεύονται αρκετά να εκσυγχρονίσουν τις διαδικασίες τους. Λίγες είναι οι χώρες που έχουν καταφέρει να αξιοποιήσουν κάποιες από τις νέες δυνατότητες των ΤΠΕ, αφού οι περισσότερες ακόμη αντιμετωπίζουν σοβαρά προβλήματα και στους πιο στοιχειώδεις τομείς ψηφιακής ανάπτυξης.  
Κάτω από αυτές τις συνθήκες, η προσπάθεια της Ευρωπαϊκής Ένωσης να επενδύσει  σε έργα ηλεκτρονικής διακυβέρνησης είναι πολύ σημαντική, καθώς η δυσαρέσκεια των πολιτών και των επιχειρήσεων, ιδίως στις λιγότερα οικονομικά ανεπτυγμένες χώρες της Ευρώπης, προς τους κρατικούς θεσμούς είναι γεγονός. Ο εκσυγχρονισμός των θεσμών αυτών και η υιοθέτηση νέων προτύπων και διαδικασιών μπορούν να βοηθήσουν στην επίλυση αυτού το προβλήματος και στην δημιουργία μιας πιο διαφανούς και παραγωγικής κυβέρνησης.  
Ένα από τα μεγαλύτερα ζητήματα που καλούνται να ξεπεράσουν τα έργα ηλεκτρονικής διακυβέρνησης είναι η επίτευξη της διαλειτουργικότητας. Κάθε κράτος χρησιμοποιεί πληθώρα συστημάτων, ηλεκτρονικών εργαλείων και μοντέλων αποθήκευσης πληροφοριών. Αυτό το γεγονός καθιστά πάρα πολύ δύσκολη την επικοινωνία ανάμεσα σε αυτά τα συστήματα και την αξιοποίηση των δεδομένων τους για την δημιουργία αξίας είτε για τις ίδιες τις κυβερνήσεις είτε για τους πολίτες ή και τις επιχειρήσεις. Ως επακόλουθο, ενώ υπάρχουν κρατικά δεδομένα που μπορούν να αξιοποιηθούν υπάρχει έλλειψη στην ανάπτυξη πληροφοριακών εφαρμογών που θα βασίζονται σε αυτά για την παραγωγή έγκυρης και χρήσιμης πληροφόρησης.   
Η επίτευξη της διαλειτουργικότητας μπορεί να επιφέρει τεράστια οφέλη στους πολίτες, καθώς η εύκολη και αξιόπιστη ανταλλαγή πληροφοριών ανάμεσα σε κρατικούς φορείς θα μειώσει την γραφειοκρατία, θα διευκολύνει τις συναλλαγές ανάμεσα στους πολίτες και το κράτος και θα θέσει τα θεμέλια για την δημιουργία καινοτόμων εφαρμογών. Βέβαια, η επιτυχής ανάπτυξη ενός μεγάλης κλίμακας έργου ΤΠΕ στα πλαίσια της Διαλειτουργικότητας αποτελεί πρόκληση και απαιτεί λεπτομερή ανάλυση των απαιτήσεων και τεχνογνωσία για να μπορέσει να υλοποιηθεί επιτυχώς.  
Στον αντίποδα όμως η ανάπτυξη μικρών εφαρμογών σε συνεργασία με τους εμπλεκόμενους φορείς μπορεί να υλοποιηθεί πιο εύκολα και να βοηθήσει την καθημερινότητα των πολιτών. Σε αυτό πλαίσιο εντάσσεται και η ανάπτυξη ενός API για την άντληση πληροφοριών από τα ανοικτά δεδομένα του Πανεπιστημίου με στόχο την αξιοποίηση του τόσο από τους φοιτητές όσο και από το ακαδημαϊκό προσωπικό.

## Αντικείμενο και στόχοι της μελέτης

Η συγκεκριμένη, λοιπόν, εργασία έχει ως αντικείμενο την δημιουργία ενός REST API για την αξιοποίηση των ανοικτών δεδομένων του Πανεπιστημίου Μακεδονίας με στόχο την επίτευξη της διαλειτουργικότητας και την προαγωγή της ηλεκτρονικής διακυβέρνησης. Η δημιουργία ενός λειτουργικό REST API έχει απλοποιηθεί αρκετά και μπορεί να γίνει χωρίς ιδιαίτερα κόστη, γεγονός που αποδεικνύεται και στην παρούσα εργασία. Η μελλοντική αξιοποίηση των ανοικτών δεδομένων του πανεπιστημίου μπορεί να διευκολύνει την καθημερινότητα τόσο των φοιτητών όσο και των ακαδημαϊκών,  αφού  θα μπορούν να αντλήσουν την πληροφόρηση που χρειάζονται σχετικά με το Πανεπιστήμιο εύκολα μέσω του API, ενώ δεν αποκλείεται να γίνει και μελλοντική του χρήση για την ανάπτυξη πιο σύνθετων εφαρμογών που θα αξιοποιούν αντίστοιχα δεδομένα άλλων Πανεπιστημίων ή Δημόσιων Υπηρεσιών στην Ευρώπη.

## Περιεχόμενα της μελέτης

Για να μπορέσει ο αναγνώστης να καταλάβει το πλαίσιο ανάπτυξης του συγκεκριμένου έργου, η παρούσα εργασία μπορεί να χωριστεί σε δύο μέρη, ένα θεωρητικό και ένα τεχνικό.  
Στο πρώτο μέρος, στο κεφάλαιο 2, γίνεται μια προσπάθεια σύντομης αλλά απαραίτητης αποσαφήνισης βασικών εννοιών που σχετίζονται άμεσα με το παρόν έργο, ώστε ο αναγνώστης να μπορέσει να καταλάβει πλήρως το λόγο που έγινε η προσπάθεια ανάπτυξης του συγκεκριμένου έργου και την πιθανή του αξία. Αυτές σχετίζονται με τα έργα και τα προγράμματα ηλεκτρονικής διακυβέρνησης της Ευρωπαϊκής Ένωσης όπως είναι η δημιουργία των Core Vocabularies και τις συνθήκες που οδήγησαν στην δημιουργία και την επικράτηση των REST APIs στον τεχνολογικό κόσμο.  
Στο 3ο κεφάλαιο παρουσιάζονται αναλυτικά τα εργαλεία και οι αρχιτεκτονικές που χρησιμοποιήθηκαν για την ανάπτυξη του CPSV-AP και οι λόγοι για τους οποίους χρησιμοποιήθηκαν.  
Στο 4ο κεφάλαιο παρουσιάζεται πλήρως το τεχνικό κομμάτι της εργασίας όπου αναλύονται η ανάπτυξη του κώδικα της εφαρμογής καθώς και ο τρόπος με τον οποίο στήθηκε διαδικτυακά.  
 Τέλος, παρατίθενται τα συμπεράσματα της εργασίας, στο 5ο κεφάλαιο, όπου καταγράφονται τα αποτελέσματα της ανάπτυξης του API, οι περιορισμοί που υπάρχουν στην χρήση του καθώς και προτάσεις για την μελλοντική ανάπτυξη και αξιοποίηση του.

# 2. Βιβλιογραφική Ανασκόπηση

## 2.1 Ηλεκτρονική Διακυβέρνηση

Ο όρος Ηλεκτρονική Διακυβέρνηση μπορεί να χρησιμοποιηθεί για να περιγράψει την χρήση τεχνολογιών πληροφορικής και τηλεπικοινωνιών (ΤΠΕ) στην υποστήριξη των κυβερνητικών λειτουργιών και την βελτιωμένη παροχή Δημόσιων Υπηρεσιών στους πολίτες και στις επιχειρήσεις και την ενίσχυση των δημοκρατικών λειτουργιών.  Παρά ταύτα, από την γέννηση του όρου την δεκαετία του 90 έως και σήμερα, δεν υπάρχει συγκεκριμένος ορισμός για το τί είναι η Ηλεκτρονική Διακυβέρνηση, αντιθέτως αρκετοί φορείς και επιστήμονες έχουν προτείνει την δική τους ερμηνεία για τον όρο. Σύμφωνα με το άρθρο των Å. Grönlund και T.A. Horan (2004), οι ορισμοί που προέρχονται κυρίως από τις εθνικές στρατηγικές διάφορων χωρών συμφωνούν πως στόχοι της Ηλεκτρονικής Διακυβέρνησης είναι:

1. Πιο αποτελεσματική διακυβέρνηση
2. Καλύτερη παροχή υπηρεσιών στους πολίτες και στις επιχειρήσεις
3. Βελτιωμένες δημοκρατικές διαδικασίες

Παράλληλα, οι ορισμοί που προέρχονται από υπερεθνικούς οργανισμούς,  διευρύνουν τον ορισμό της Ηλεκτρονικής Διακυβέρνησης δίνοντας έμφαση στο μετασχηματισμό των Δημόσιων Υπηρεσιών μέσω των ΤΠΕ με στόχο την βελτίωση των σχέσεων των πολιτών και τον επιχειρήσεων με τους δημόσιους φορείς. Ο ορισμός, λοιπόν, της Ηλεκτρονικής Διακυβέρνησης, σε αυτήν την περίπτωση είναι ευρύτερος και αντιμετωπίζει τις νέες τεχνολογίες ως μέσο επίτευξης λιγότερης διαφθοράς, περισσότερης διαφάνειας και αύξησης των εσόδων μιας χώρας.

Γενικότερα δεν μπορεί να δοθεί ένας καθολικός ορισμός στην Ηλεκτρονική Διακυβέρνηση, καθώς αυτή είναι άμεσα συνυφασμένη με τις κρατικές λειτουργίες, οι οποίες επηρεάζονται από πολιτικούς και κοινωνικούς παράγοντες και διαφέρουν από χώρα σε χώρα. Κατά επέκταση, κάθε κυβερνητική στρατηγική τείνει να προσαρμόζει τον όρο στο εκάστοτε όραμα της. Όμως σε όλους του ορισμούς είναι προφανής η βαρύτητα που δίνεται στις ΤΠΕ ως μέσο ενίσχυσης των κρατικών μηχανισμών. Οι ΤΠΕ ουσιαστικά, μέσω των έργων που υλοποιούνται στα πλαίσια της Ηλεκτρονικής Διακυβέρνησης, συμβάλλουν στην ενδυνάμωση των σχέσεων του κράτους με τις επιχειρήσεις (Government to Business), τους εργαζομένους(Government to Employee), τους πολίτες (Government to Citizen) αλλά και των δημόσιων φορέων (Government to Government) προσφέροντας πιο φιλικές, διαφανείς, εύχρηστες υπηρεσίες.

### 2.1.1 Η Ηλεκτρονική Διακυβέρνηση στην Ευρωπαϊκή Ένωση

Η Ευρωπαϊκή Ένωση (ΕΕ) αναγνώρισε από νωρίς τα κοινωνικά και οικονομικά οφέλη που θα είχε η αξιοποίηση των ΤΠΕ για τα κράτη μέλη της για αυτό και έθεσε ως προτεραιότητά της τον εκσυγχρονισμό της ώστε να είναι “έτοιμη για την ψηφιακή εποχή”. Στο πλαίσιο αυτού του εκσυγχρονισμού, έκρινε πως ήταν απαραίτητο να γίνει ένας ψηφιακός μετασχηματισμός των δημόσιων φορέων της Ευρωπαϊκής Ένωσης για να αναπτυχθεί η ηλεκτρονική διακυβέρνηση στα κράτη μέλη της. Με την ανάπτυξη αυτή, η ΕΕ στοχεύει στην βελτίωση των δημόσιων υπηρεσιών και δημοκρατικών διαδικασιών και την ενδυνάμωση της συμμετοχής και στήριξης στα δημόσια πολιτικά. Για αυτό και σχεδίασε το «Πλάνο Δράσης Ηλεκτρονικής Διακυβέρνησης 2011-2015» που στην συνέχεια ανανέωσε στο «Πλάνο Δράσης Ηλεκτρονικής Διακυβέρνησης 2016-2020».

Το πλάνο αυτό στηρίζεται σε τρείς βασικές προτεραιότητες πολιτικής που θα πρέπει να ακολουθηθούν από τις κυβερνήσεις της Ευρώπης. Αυτές είναι:

* 1. Ο εκσυγχρονισμός των δημοσίων διοικήσεων με σκοπό να δημιουργηθούν αποτελεσματικές δημόσιες υπηρεσίες μέσω της χρήσης βασικών ψηφιακών μέσων όπως τα eID, eSignature, eDelivery.
  2. Η επίτευξη της δια-συνοριακής διαλειτουργικότητας ώστε να επιτραπεί η ευκολότερη κινητικότητα επιχειρήσεων και πολιτών εντός της ΕΕ
  3. Diagram

     Description automatically generated Η υλοποίηση ψηφιακής αλληλεπίδρασης μεταξύ των δημόσιων φορέων και των πολιτών/επιχειρήσεων ώστε να δημιουργηθούν υψηλής ποιότητας δημόσιες υπηρεσίες

**Εικόνα 1** Βασικοί Πυλώνες δράσεων του Πλάνου ΗΔ για το 2016-2020

Η εφαρμογή των προτεραιοτήτων αυτών σε όλα τα κράτη μέλη της ΕΕ αναμένεται να ωφελήσει τόσο τους πολίτες όσο και τις επιχειρήσεις στην Ευρώπη αφού θα μπορούν πλέον να επωφεληθούν πλήρως από τις δημόσιες υπηρεσίες, που συχνά έως τώρα χαρακτηρίζονταν χρονοβόρες, απαρχαιωμένες, πολύπλοκες, μη παραγωγικές και χρησιμοποιούνταν κυρίως από ανάγκη. Ο εκσυγχρονισμός τους, με την βοήθεια των ΤΠΕ, αναμένεται να βοηθήσει στο να ξεπεραστούν οι παραπάνω παθογένειες καλλιεργώντας ένα περιβάλλον καινοτομίας και οικονομικής ανάπτυξης, στο οποίο οι δημόσιοι οργανισμοί συνδράμουν θετικά και όχι αποθαρρυντικά στην καθημερινή ζωή.

### Βασικές Αρχές του Πλάνου Δράσης Ηλεκτρονικής Διακυβέρνησης

Για να υλοποιήσει τα παραπάνω η Ευρωπαϊκή Ένωση, έθεσε εφτά βασικές αρχές για την ανάπτυξη και τον εκσυγχρονισμό των δημόσιων υπηρεσιών. Οι δημόσιες υπηρεσίες θα πρέπει, λοιπόν, να:

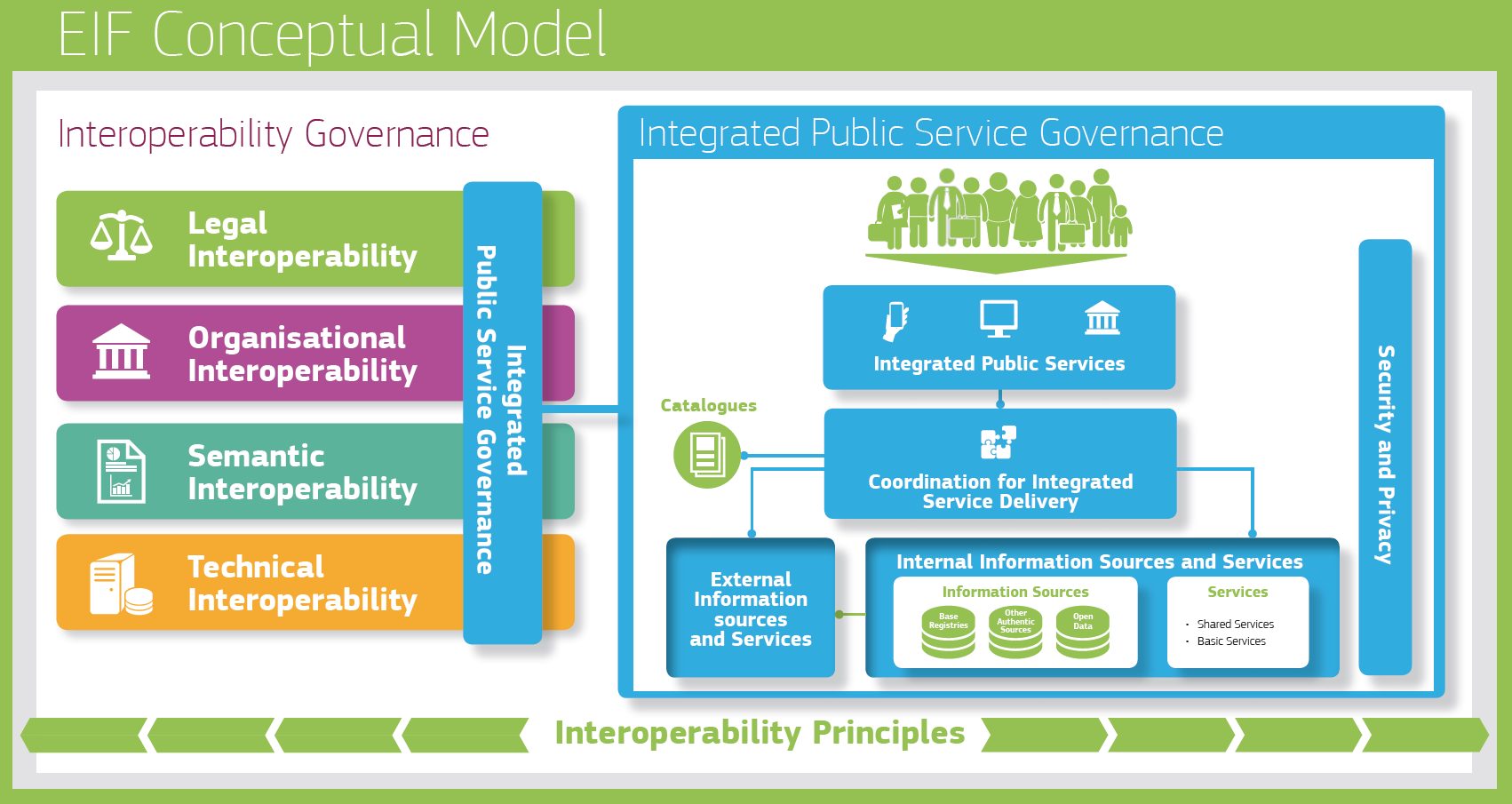
* 1. Είναι εξ ορισμού ψηφιακές: οι ΔΥ θα πρέπει πλέον να προσφέρονται ηλεκτρονικά και να έχουν ψηφιοποιηθεί στον μεγαλύτερο δυνατό βαθμό
  2. Ακολουθούν την «μόνον άπαξ» αρχή: η εισαγωγή στοιχειών απαραίτητων για την δημόσια διοίκηση θα πρέπει να γίνεται μόνο μια φόρα από τους πολίτες και τις επιχειρήσεις
  3. Είναι προσβάσιμες από όλους: ο ψηφιακός χαρακτήρας των ΔΥ προϋποθέτει πως όλοι οι πολίτες θα έχουν πρόσβαση σε αυτές, ανεξαρτήτως της τοποθεσίας τους, των ικανοτήτων τους ή άλλων παραγόντων
  4. Είναι ανοικτές και διαφανείς: οι ΔΥ θα πρέπει να δημοσιεύουν πληροφορίες που σχετίζονται με αυτές και μπορούν να αξιοποιηθούν από τρίτους
  5. Να λειτουργούν διασυνοριακά οι ΔΥ θα πρέπει να μπορούν να σχεδιάζονται και λειτουργούν σε ευρωπαϊκό επίπεδο, ξεπερνώντας τα γεωγραφικά σύνορα
  6. Να είναι αξιόπιστες και ασφαλείς: οι ΔΥ οφείλουν να προστατεύουν τις πληροφορίες των πολιτών και των επιχειρήσεων, λειτουργώντας χωρίς προβλήματα
  7. Να έχουν δια λειτουργικό χαρακτήρα: οι ΔΥ οφείλουν να προσφέρουν και να δέχονται δεδομένα από άλλες ΔΥ

Όπως φαίνεται από αυτές τις αρχές, η κατάρριψη των τεχνολογικών εμποδίων ανάμεσα στις Δημόσιες Υπηρεσίες διαφορετικών χωρών και η επίτευξη αξιόπιστης και ασφαλούς επικοινωνίας μεταξύ τους θεωρούνται πολύ σημαντικοί στόχοι για την Ευρωπαϊκή Ένωση. Αυτή την στιγμή ακόμη και ΔΥ που βρίσκονται στην ίδια χώρα αδυνατούν να επικοινωνήσουν μεταξύ τους και να ανταλλάξουν δεδομένα, δημιουργώντας την ανάγκη να κατατίθενται τα ίδια αποδεικτικά έγγραφα σε πολλούς οργανισμούς πολλές φορές από τον ίδιο πολίτη/επιχείρηση. Αυτό έχει ως αποτέλεσμα να αυξάνονται τα κόστη διαχείρισης για τους Δημόσιους Οργανισμούς καθότι ενισχύεται η γραφειοκρατία και αυξάνονται οι χρόνοι διεκπεραίωσης παροχής μιας υπηρεσίας και παράλληλα να παραμένουν δυσαρεστημένοι οι πολίτες, που καλούνται να εμπλέκονται σε διαδικασίες που απαιτούν πολύ χρόνο και κόπο άσκοπα.

Επιπλέον σε ένα τέτοιο περιβάλλον αναπτύσσεται πιο έντονα και η ανομοιογένεια στην δομή και περιγραφή των δημόσιων υπηρεσιών που μπορεί να έχουν όμως κοινούς σκοπούς. Αυτό ενισχύει την δημιουργία σύγχυσης τόσο στους ίδιους του Δημόσιους Οργανισμούς που τις παρέχουν όσο και στους πολίτες που επιθυμούν να τις χρησιμοποιήσουν.

## 2.2 Διαλειτουργικότητα

Για να αντιμετωπίσει όλα παραπάνω η Ευρωπαϊκή Ένωση, μέσω της στρατηγικής της για την ηλεκτρονική διακυβέρνηση, ανέπτυξε κάποιες πρωτοβουλίες για την επίτευξη της Διαλειτουργικότητας ανάμεσα στις δημόσιες υπηρεσίες της. Με τον όρο Διαλειτουργικότητα σύμφωνα με νέο Ευρωπαϊκό Πλαίσιο Διαλειτουργικότητας περιγράφεται “η ικανότητα των οργανισμών να αλληλοεπιδρούν μεταξύ τους προς αμοιβαία επωφελείς στόχους, που περιλαμβάνουν την ανταλλαγή πληροφοριών και γνώσης μεταξύ αυτών των οργανισμών, μέσω των επιχειρηματικών διαδικασιών που αυτοί υποστηρίζουν, διαμέσου της ανταλλαγής δεδομένων μεταξύ των συστημάτων ΤΠΕ τους».



**Εικόνα 2** Τα είδη της Διαλειτουργικότητας κατά τον EIF

Πιο αναλυτικά, η έννοια της Διαλειτουργικότητας αφορά τέσσερα επίπεδα:

* 1. Την νομοθετική διαλειτουργικότητα
  2. Την οργανωτική διαλειτουργικότητα
  3. Την σημασιολογική διαλειτουργικότητα
  4. Την τεχνολογική διαλειτουργικότητα

Η Διαλειτουργικότητα των ΔΥ της ΕΕ αναμένεται να επιφέρει τεράστια οφέλη τόσο στις ίδιες τις Δημόσιες Υπηρεσίες όσο και στους πολίτες και τις επιχειρήσεις της Ευρωπαϊκής Ένωσης, ενισχύοντας τις σχέσεις ανάμεσα στους δημόσιους οργανισμούς με άλλους δημόσιους οργανισμούς, τους πολίτες και τις επιχειρήσεις.

### 2.2.1 ISA

Το πρόγραμμα ISA (Interoperability solutions for public administrations, businesses and citizens) είναι υπεύθυνο για την στήριξη των ενεργειών διαλειτουργικότητας στην ΕΕ. Πιο συγκεκριμένα, στηρίζει την ανάπτυξη διαλειτουργικών λύσεων για τις δημόσιες υπηρεσίες τόσο σε εθνικό επίπεδο (cross-sector) και σε ευρωπαϊκό (cross-border). Αυτή τη στιγμή υπό την αιγίδα αυτού του προγράμματος προσφέρονται 24 λύσεις διαλειτουργικότητας για την ανάπτυξη καινοτόμων εφαρμογών.

A screenshot of a computer

Description automatically generated with low confidence

**Εικόνα 3** Οι λύσεις του προγράμματος ISA

### 2.2.2 Core Vocabularies

Μια σημαντική λύση που αναπτύσσεται στα πλαίσια του ISA είναι και τα Core Vocabularies. Τα Core Vocabularies είναι απλοποιημένα, επαναχρησιμοποιούμενα και επεκτάσιμα μοντέλα δεδομένων που μπορούν να χρησιμοποιηθούν από τις δημόσιες διοικήσεις για να:

1. Αναπτύξουν νέας συστήματα από ένα εννοιολογικό και λογικό μοντέλο δεδομένων
2. Να ανταλλάσσουν πληροφορίες μεταξύ διαφορετικών συστημάτων
3. Να ενσωματώνουν δεδομένα από διάφορες πηγές
4. Να δημοσιεύουν δεδομένα σε μια κοινή μορφή

Αυτή τη στιγμή υπάρχουν τέσσερα Core Vocabularies:

1. To Core Person: για την περιγραφή των βασικότερων χαρακτηριστικών ενός ανθρώπου όπως είναι το όνομα, η ηλικία, το φύλο κτλ.
2. Το Core Business: για την περιγραφή των βασικότερων χαρακτηριστικών μιας νομικής οντότητας
3. Το Core Location: για την περιγραφή μιας γεωγραφικής τοποθεσίας
4. Το Criterion and Core Evidence: περιγράφει τις αρχές και τα μέσα που χρειάζεται να εκπληρώνει μια ιδιωτική οντότητα ώστε να μπορεί να είναι κατάλληλη για την διεκπεραίωση μιας ΔΥ.
5. To Core Public Organisation: για την περιγραφή των Δημόσιων Οργανισμών της Ευρωπαϊκής Ένωσης

## 2.3 CPSV – AP

Το Core Public Services Vocabulary (CPSV-AP) είναι ένα πρότυπο μοντελοποίησης δεδομένων που αφορά δημόσιες υπηρεσίες και ανήκει στην οικογένεια των Βασικών Λεξιλογίων (Core Vocabularies) που έχει σχεδιάσει η Ευρωπαϊκή Ένωση βασίζεται στην λογική των Ανοικτών Συνδεδεμένων Δεδομένων. Με αυτό το μοντέλο γίνεται προσπάθεια να περιγραφούνε με όμοιο τρόπο όλες οι δημόσιες υπηρεσίες που προσφέρονται από οποιοδήποτε δημόσιο οργανισμό στην Ευρώπη. Η κοινή περιγραφή τους αποσκοπεί στην επίτευξη της διαλειτουργικότητας ανάμεσα στους δημόσιους οργανισμούς της ΕΕ.

Τέτοιες υπηρεσίες στο πλαίσιο παραδείγματος χάριν ενός πανεπιστημίου μπορούν να θεωρηθούν η «εγγραφή στο τμήμα», «η έκδοση φοιτητικής ταυτότητας», « η παραλαβή του πτυχίου» και προσφέρονται από όλα τα πανεπιστήμια στην Ευρώπη. Παρά ταύτα, αυτή τη στιγμή δεν έχει καθιερωθεί ένας κοινό τρόπος περιγραφής των γεγονότων αυτών για όλα τα πανεπιστήμια. Αυτό έχει ως απόρροια την δυσκολία εύρεσης της πληροφόρησης που χρειάζονται οι πολίτες και άλλοι οργανισμοί-επιχειρήσεις από τα πανεπιστήμια καθώς δυσχεραίνεται η επικοινωνία.

### 2.3.1 Η λύση του CPSV-AP

Αναγνωρίζοντας την έλλειψη αυτή ο ISA σχεδίασε το μοντέλο δεδομένων CPSV το 2013 για να μπορέσει να δώσει ένα κοινό τρόπο περιγραφής των Δημόσιων Υπηρεσιών όλης της Ευρώπης. Με την χρήση του CPSV-AP, οι Δημόσιοι Οργανισμοί μπορούν να:

1. Προσφέρουν πληροφορίες για τις υπηρεσίες τους με έναν φιλικό για τον χρήστη τρόπο, αφού θα υπάρχει μια κοινή λογική για την περιγραφή όλων των δημόσιων υπηρεσιών της Ευρώπης
2. Προσαρμόσουν τα διαφορετικά μοντέλα δεδομένων που χρησιμοποιούν σε ένα μόνο μοντέλο χρησιμοποιώντας μια περιγραφή, με αυτό τον τρόπο όλες οι Πύλες της ΕΕ θα μπορούν να διαμοιράζονται πληροφορίες
3. Δημοσιεύουν πληροφορίες σε Πύλες ηλεκτρονικής διακυβέρνησης με έναν πιο αποτελεσματικό και διαλειτουργικό τρόπο

Το 2018 κυκλοφόρησε η νεότερη έκδοση του CPSV-AP, η οποία εμπλουτίστηκε με βάση τα σχόλια των κρατών μελών της Ευρωπαϊκής Ένωσης.

### 2.3.2 Οι κλάσεις του CPSV-AP

Diagram, schematic

Description automatically generatedΣύμφωνα με την νέα έκδοση του CPSV-AP ορίζονται δεκαέξι κλάσεις εκ των οποίων οι δύο είναι υποχρεωτικές.

**Εικόνα 4** To πλήρες μοντέλο του CPSV-AP

**Η κλάση Public Organisation**

Η κλάση Public Organisation περιγράφει έναν Δημόσιο Οργανισμό.

**Η κλάση Public Service**

Η κλάση Public Service περιγράφει μια Δημόσια Υπηρεσία ενός Δημόσιου Οργανισμού.  
Οι παραπάνω δύο κλάσεις είναι υποχρεωτικές καθώς δίχως αυτές δεν μπορούν να οριστούν και δεν έχουν λόγο ύπαρξης οι υπόλοιπες.

**Οι κλάσεις Event**

Η κλάση Event περιγράφει ένα γεγονός/περιστατικό που σχετίζεται με μια ΔΥ.  
Η παραπάνω κλάση εξειδικεύεται περισσότερο μέσω των κλάσεων Business Event και Life Event. Η πρώτη περιγράφει γεγονότα που σχετίζονται με επιχειρήσεις (π.χ. έναρξη επιχείρησης) ενώ η δεύτερη με περιστατικά στη ζωή ενός ανθρώπου στο οποίο χρειάζεται να διεκπεραιωθεί μια ΔΥ (π.χ. εγγραφή σε ένα τμήμα πανεπιστημίου)

**Η κλάση Evidence**

Στην κλάση Evidence περιέχονται όλα τα στοιχεία που μπορούν να τεκμηριώσουν την συμμόρφωση με ένα κριτήριο μιας ΔΥ.

**Η κλάση Output**

Με αυτήν την κλάση περιγράφονται τα αποτελέσματα της ολοκλήρωσης μιας ΔΥ.

**Η κλάση Formal Framework**

Η κλάση αυτή περιγράφει όλα τα νομικά έγγραφα και κείμενα στρατηγικών στα οποία έχει βασιστεί η εκτέλεση μιας ΔΥ

**Η κλάση Agent**

Οποιοσδήποτε φορέας, πρόσωπο ή ομάδα έχουν κάποιο ρόλο στην πραγματοποίηση μιας ΔΥ, περιγράφονται μέσω αυτής της κλάσης.

**Η κλάση Criterion Requirement**

Με την κλάση αυτή περιγράφονται όλα τα κριτήρια που απαιτούνται για την εκκίνηση μιας ΔΥ.

**Η κλάση Participation**

Η κλάση αυτή περιγράφει αναλυτικά πως ένας Agent αλληλοεπιδρά με μια ΔΥ

**Η κλάση Cost**

Αυτή η κλάση προσδιορίζει το κόστος που καλείται να πληρώσει ο καταναλωτής μιας ΔΥ

**Η κλάση Channel**

Περιγράφει το κανάλι επικοινωνίας που χρησιμοποιείται για την παροχή μιας ΔΥ

**Η κλάση Rule**

Η κλάση αυτή αντιπροσωπεύει τα κείμενα που καθορίζουν τις διαδικασίες που πρέπει να εκτελεστούν ώστε να διεκπεραιωθεί μια ΔΥ.

**Η κλάση Contact Point**

Η κλάση αυτή περιγράφει το σημείο επικοινωνίας που σχετίζεται με μια ΔΥ.

**Η κλάση Opening Hours**

Με την opening hours περιγράφονται οι ώρες λειτουργίας ενός σημείου επικοινωνίας

Στον παρακάτω πίνακα φαίνονται όλες οι κλάσεις του CPSV-AP μαζί με τα αντίστοιχα URI όπως αυτά έχουν οριστεί από την ΕΕ.

|  |  |
| --- | --- |
| Κλάση | URI |
| Public Service | cpsv:PublicService |
| Public Organisation | cv:PublicOrganisation |
| Event | cv:Event |
| Business Event | cv:BusinessEvent |
| Life Event | cv:LifeEvent |
| Evidence | cv:Evidence |
| Output | cv:Ouput |
| Formal Framework | cpsv:FormalFramework |
| Agent | dct:Agent |
| Criterion Requirement | cv:CriterionRequirement |
| Participation | cv:Participation |
| Cost | cv:Cost |
| Channel | cv:Channel |
| Rule | cpsv:Rule |
| Contact Point | schema:ContactPoint |
| Opening Hours | schema:OpeningHoursSpecification |

**Πίνακας 1** Ονομασίες κλάσεων του CPSV-AP μαζί με τα URI τους

Η χρήση των URI γίνεται ώστε να είναι εύκολος ο εντοπισμός τους στον σημασιολογικό ιστό. Αυτή είναι μια πολύ σημαντική λειτουργία καθώς επιτρέπει την αξιοποίηση όλων των δεδομένων που ακολουθούν το πρότυπο CPSV-AP από άλλα υπολογιστικά συστήματα για την δημιουργία εφαρμογών τόσο από κυβερνητικούς φορείς όσο και από τρίτους. Οι πληροφορίες του κάθε δημόσιου οργανισμού αποτυπώνονται με τον ίδιο τρόπο, παρέχοντας μια κοινή μεθοδολογία για την μαζική αξιοποίησή τους. Χωρίς την χρήση των URIs θα ήταν αδύνατη η ύπαρξη των πληροφοριών αυτών σε μια μορφή αξιοποιήσιμη από υπολογιστικά συστήματα, αντιθέτως οι πληροφορίες θα ήταν αποκεντρωμένες και θα ήταν σχεδόν αδύνατη η περαιτέρω επεξεργασία και χρήση τους.

## 2.4 APIS

Η έννοια του Application Programming Interface χρησιμοποιήθηκε από τα πρώτα χρόνια της Επιστήμης των Υπολογιστών για να περιγράψει μια διεπαφή ενός προγράμματος που επιτρέπει τις κλήσεις προς αυτό για την ανταλλαγή δεδομένων. Ουσιαστικά η χρήση του API επιτρέπει σε ένα πρόγραμμα ή χρήστη να στείλει αιτήματα προς ένα άλλο πρόγραμμα και να λάβει απαντήσεις. Πριν την διάδοση των διαδικτυακών εφαρμογών ο όρος API χρησιμοποιούνταν κυρίως για να περιγράψει οποιαδήποτε «κλήση» προς ένα συγκεκριμένο σύστημα και την αναμενόμενη απάντησή του. Με αυτήν την λογική API μπορούσε να θεωρηθεί και μια βιβλιοθήκη ενός λογισμικού ή ένα framework .  
Η ραγδαία όμως εξέλιξη του διαδικτύου, οδήγησε και στην ανάπτυξη πολλών εφαρμογών αναπτυγμένων σε διαφορετικές αρχιτεκτονικές/τεχνολογίες που όμως έχουν την ανάγκη να επικοινωνήσουν μεταξύ τους και να ανταλλάξουν πληροφορίες. Αυτό το γεγονός δημιούργησε αρκετά προβλήματα καθώς η ετερογένεια των συστημάτων συχνά δεν μπορούσε να υποστηρίξει μια ομαλή, εύκολη, αξιόπιστη και επεκτάσιμη επικοινωνία. Οι εφαρμογές του διαδικτύου με άλλα λόγια δεν μπορούσαν να έρθουν σε επαφή με μεγάλο αριθμό άλλων εφαρμογών, περιορίζοντας με αυτόν τον τρόπο το «κοινό» που θα μπορούσε να τις αξιοποιήσει.  
Για να αντιμετωπιστεί αυτό το πρόβλημα δημιουργήθηκαν πολλές διαφορετικές λύσεις και πρωτόκολλα επικοινωνίας όπως το CORBA (Common Object Request Architecture) και το SOAP (Simple Object Access Protocol) τα οποία αν και έλυσαν διάφορα προβλήματα επικοινωνίας μεταξύ των εφαρμογών, είχαν κάποια μειονεκτήματα. Αυτά τα μειονεκτήματα προσπάθησε να ξεπεράσει η αρχιτεκτονική REST.

### 2.4.1 Η αρχιτεκτονική REST

Η αρχιτεκτονική REST (Representational state transfer) περιγράφει μια μορφή κανόνων και περιορισμών για τον τρόπο που θα πρέπει να συμπεριφέρονται τα κατανεμημένα συστήματα υπερμέσων όπως είναι το Διαδίκτυο. Ουσιαστικά το REST ορίζει πως μπορούν να μεταβληθούν οι πόροι ενός συστήματος. Αυτή η μεταβολή ορίζεται μέσω του πρωτοκόλλου HTTP.

Βασικά αξιώματα για την σχεδίαση συστημάτων REST είναι:

1. Οτιδήποτε μπορεί να θεωρηθεί πόρος
2. Κάθε πόρος είναι αναγνωρίσιμος από ένα μοναδικό URI
3. Γίνεται χρήση των βασικότερων μεθόδων HTTP (GET, PUT, DELETE)
4. Οι πόροι μπορεί να έχουν μια ποικιλία αναπαραστάσεων
5. Η επικοινωνία είναι «state less»

Με την χρήση της αρχιτεκτονικής REST στόχος είναι να υπάρχουν:

1. Διαχωρισμός ανάμεσα σε πόρους και αναπαραστάσεις
2. Ορατότητα στην επικοινωνία μεταξύ των services
3. Μια απλή διεπαφή
4. Φορητότητα
5. Επεκτασιμότητα
6. Αυξημένη Απόδοση
7. Αξιοπιστία

### 2.4.2 REST APIS

Graphical user interface, text

Description automatically generatedΈνας από τους δημοφιλέστερους και επικρατέστερους τύπους API είναι τα REST API. Με τον όρο REST API περιγράφουμε τα API που ακολουθούν την αρχιτεκτονική REST (Representational state transfer). Ουσιαστικά πρόκειται για εφαρμογές διαδικτύου που εκμεταλλεύονται την αρχιτεκτονική REST για να ανταλλάσσουν μηνύματα ανάμεσα σε διαφορετικά συστήματα ή ανάμεσα σε κάποιον χρήστη και ένα σύστημα. Τα REST APIs ορίζουν κάποιες λειτουργίες που επιθυμούν να παρέχουν πάνω σε έναν server, τις οποίες περιγράφουν αναλυτικά στο documentation τους.

**Εικόνα 5** Οι μέθοδοι REST όπως εμφανίζονται σε ένα REST API

Αυτές οι λειτουργίες στηρίζονται στις μεθόδους του πρωτοκόλλου HTTP για να

δημιουργήσουν τα endpoints τους.

Πιο αναλυτικά, έχουμε τις μεθόδους GET :

Get allItems: όπου μπορούμε να πάρουμε όλα στοιχεία που περιγράφει μια κλήση πχ Get allUsers ως απάντηση από τον server.  
Get Item{id}: όπου επιστρέφονται όλες πληροφορίες που σχετίζονται με ένα αντικείμενο που έχει το id , πχ GetUser{1092} εφόσον ο χρήστης υπάρχει, το σύστημα θα επιστρέψει τις πληροφορίες που σχετίζονται με αυτόν.  
Έχουμε την μέθοδο POST που προσθέτει ένα αντικείμενο στον server με τον οποίο επικοινωνεί το API, παραδείγματος χάριν POST User.   
Με την μέθοδο PUT ανανεώνουμε τις πληροφορίες που έχουμε για ένα αντικείμενο. Μια κλήση PUT User{1092} θα ανανεώσει τις πληροφορίες που σχετίζονται με τον user με id 1092.  
Τέλος, η μέθοδος DELETE διαγράφει ένα στοιχείο από τον server. Η κλήση DELETE User{1092}θα διαγράψει όλα τα στοιχεία που σχετίζονται με τον user με id 1092.

### 2.4.3 API ECONOMY

Αν και η αρχιτεκτονική των REST APIs υπάρχει από τις αρχές του 21ου αιώνα, μόλις τα τελευταία χρόνια φαίνεται πως έχει ξεκινήσει να γίνεται πιο έντονη η παρουσία τους. Η άφιξη των τεχνολογιών cloud και η εκθετική αύξηση των δεδομένων που παράγονται καθημερινά, ανέδειξε την ανάγκη για συστήματα που μπορούν να ανταλλάξουν πληροφορίες χωρίς προβλήματα επικοινωνίας. Η αξιοποίηση αυτών των δεδομένων για την εξαγωγή στατιστικών, την εκπαίδευση μοντέλων μηχανικής μάθησης και την προσφορά καλύτερης πληροφόρησης αποτελεί βασικό αίτημα των καιρών και τα REST APIs φαίνεται πως μπορούν να αποτελέσουν μέσο κάλυψης του αιτήματος αυτού.  
Ήδη οι μεγαλύτεροι τεχνολογικοί κολοσσοί και αρκετά ερευνητικά ιδρύματα επενδύουν στην ανάπτυξη APIs καθώς μπορούν να έχουν κέρδος προσφέροντας τους σε τρίτους, δημιουργώντας νέα επιχειρηματικά μοντέλα. Η πληροφόρηση πλέον φαίνεται πως ξεπερνάει τυχόν τεχνολογικά εμπόδια και είναι διαθέσιμη από ποικίλα μέσα (smartphones, personal computers, smart TVs).   
Αυτήν την ραγδαία ανάπτυξη των APIs έχουν στηρίξει και τα σύγχρονα εργαλεία προγραμματισμού που επιτρέπουν ακόμη και πιο αρχάριους προγραμματιστές να δημιουργήσουν δικά τους REST API χωρίς ιδιαίτερο κόπο. Βέβαια, αυτή η ραγδαία και χαοτική ανάπτυξη έχει επιφέρει και αρνητικά αποτελέσματα, καθώς δεν είναι λίγοι αυτοί που σχεδιάζουν APIs χωρίς την απαραίτητη ανάλυση και αρχιτεκτονική με αποτέλεσμα να είναι δύσχρηστα, δυσνόητα και να μην μπορούν εν τέλει να εξυπηρετήσουν τον σκοπό για τον οποίο δημιουργήθηκαν. Αυτό προκαλεί ζητήματα καθώς δημιουργείται σύγχυση τόσο στους δημιουργούς API όσο και στους χρήστες τους.  
Όπως και αν έχει όμως, οι δυνατότητες που προσφέρουν τα APIs φαίνεται πως δεν έχουν περάσει απαρατήρητες στον κόσμο της πληροφορικής αφού πλέον γίνεται λόγος για την «Οικονομία των APIs» δηλαδή την δημιουργία αυτού του ενεργού και ταχεία αναπτυσσόμενου οικοσυστήματος στο οποίο τα API κατέχουν κεντρικό ρόλο στην δημιουργία καινοτόμων εφαρμογών και τεχνολογικών λύσεων.

## 2.5 Συμπεράσματα

Συμπεραίνοντας, στον παρόν κεφάλαιο γίνεται έκδηλη η ύπαρξη των κατάλληλων συνθηκών για την δημιουργία εργαλείων που προάγουν την ηλεκτρονική διακυβέρνηση στην Ευρώπη. Τόσο οι ευρωπαϊκοί φορείς όσο και ο τεχνολογικός κόσμος συνεχώς εξελίσσονται και προσφέρουν πιο ανανεωμένα μέσα την ανάπτυξη ποικίλων πρωτοβουλιών. Παρά ταύτα φαίνεται πως υπάρχει η τάση ανάπτυξης των δύο αυτών κόσμων ξεχωριστά, με λίγα έργα να αξιοποιούν πλήρως τις δυνατότητες που παρέχονται και από τις δύο πλευρές.

# 3.Μεθοδολογία

Η ανάπτυξη ενός REST API μπορεί να γίνει με πολλά εργαλεία και μεθοδολογίες , καθώς οι περισσότερες νέες γλώσσες προγραμματισμού υποστηρίζουν την αρχιτεκτονική REST με δικές τους βιβλιοθήκες ή βιβλιοθήκες τρίτων. Η υλοποίηση του προϋποθέτει την εξοικείωση με βασικές έννοιες προγραμματισμού, διαδικτυακής επικοινωνίας και τον σχεδιασμό ουσιωδών και ευνόητων endpoints.  
Ουσιαστικά μπορούν να γίνει διάκριση σε δύο στάδια ανάπτυξης, το πρώτο αφορά την τεχνική υλοποίηση του API μέσω κάποιας γλώσσας προγραμματισμού και το δεύτερο αφορά την δημοσίευσή του στο διαδίκτυο.  
Στο πρώτο έμφαση δόθηκε στην επιλογή εργαλείων και τεχνικών που είναι αξιόπιστες και εύκολα επεκτάσιμες, ώστε το API να μπορέσει να χρησιμοποιηθεί μελλοντικά και από άλλους. Στο δεύτερο στάδιο τα κριτήρια επιλογής μεθόδου δημοσίευσης καθορίστηκαν από τους διαθέσιμους πόρους.

## 3.1 Τα εργαλεία ανάπτυξης του CPSV-AP REST API

Για την υλοποίηση του API επιλέχθηκε το .NET Core Framework βασισμένο στην C# της Microsoft ως μέσο ανάπτυξης. Η επιλογή του συγκεκριμένου Framework έγινε διότι περιέχει έτοιμο template ανάπτυξης για REST APIs και πλήρες documentation υποστηρίζοντας ακόμη και έναν πιο αρχάριο προγραμματιστή στην γρήγορη και αξιόπιστη υλοποίηση του. Επιπλέον, έχει επικρατήσει ιδιαίτερα στην αγορά και μπορεί να υποστηρίξει την αξιόπιστη ανάπτυξη, γρήγορων και επεκτάσιμων εφαρμογών.  
Γενικότερα, η Microsoft προσφέρει μια σειρά εργαλείων και βιβλιοθηκών που ακολουθούν τα πιο σύγχρονα πρότυπα προγραμματισμού και υποστηρίζεται από μια μεγάλη κοινότητα παγκοσμίως με πολλά forums, plugins, libraries και εκπαιδευτικό υλικό

Ένα από τα πιο δημοφιλή εργαλεία της είναι το Visual Studio, ένα πολύ δυνατό IDE που έχει επικρατήσει στην αγορά λόγω των πολλών δυνατοτήτων και διευκολύνσεων που προσφέρει στον προγραμματιστή. Για αυτό το λόγο επιλέχθηκε και ως το IDE ανάπτυξης του API. Επιπλέον χρησιμοποιήθηκαν βιβλιοθήκες τρίτων για να μπορέσει να γίνει η επικοινωνία με το RDF store του Πανεπιστημίου Μακεδονίας που περιέχει ανοικτά δεδομένα που ακολουθούν το CPSV-AP.

### C#

Η C# είναι μια γλώσσα προγραμματισμού που αναπτύχθηκε από την Microsoft ως εξέλιξη των γλωσσών που ανήκουν στην οικογένεια της C. Η πρώτη έκδοση της γλώσσας δημιουργήθηκε το 2000 από τον Anders Hejlsberg με σκοπό αυτή:

* 1. Να είναι απλή, μοντέρνα και να εξυπηρετεί πολλαπλούς σκοπούς ακολουθώντας αντικειμενοστραφή πρότυπα
  2. Να υποστηρίζει βασικές αρχές ανάπτυξης λογισμικού με στόχο την ευρωστία και την ανθεκτικότητα του λογισμικού
  3. Να υποστηρίζει ποικιλία διαφορετικών συστημάτων (λειτουργικά συστήματα, μικρές εφαρμογές, ενσωματωμένα συστήματα)
  4. Να μπορεί να δημιουργήσει λογισμικό κατάλληλο για την ανάπτυξη σε κατανεμημένα περιβάλλοντα
  5. Να είναι φορητή

H C# όπως αναφέρεται και στους στόχους της είναι όντως μια γλώσσα που μπορεί να χρησιμοποιηθεί για την ανάπτυξη τόσο απαιτητικών όσο και πιο απλών λογισμικών-προγραμμάτων. Στα βασικότερα πλεονεκτήματά της συμπεριλαμβάνονται:

1. Η αντικειμενοστρέφειά της
2. Η αυτόματη συλλογή απορριμμάτων
3. Είναι cross platform
4. Υποστηρίζει αντίστροφη συμβατότητα
5. Η ακεραιότητά και η διαλειτουργικότητα της

Αυτή τη στιγμή κυκλοφορεί η 9.0 έκδοση της γλώσσας και μαζί με τις υπόλοιπες γλώσσες της οικογένειας C, έχει αποτελέσει την βάση ανάπτυξη των πιο σύγχρονων εκδόσεων των Windows και πολλών εφαρμογών της Microsoft που έχουν κυριαρχήσει στην αγορά. Αξίζει να σημειωθεί ότι η συγκεκριμένη γλώσσα προτιμάται ιδιαιτέρως για την ανάπτυξη μεγάλων έργων λογισμικού όπως τα video games, web services, console applications.  
Στην C# έχει βασιστεί και το .NET Framework, μια ομάδα βιβλιοθηκών και εργαλείων που έχουν αναπτυχθεί από την Microsoft για την δημιουργία σύγχρονων, ευέλικτων, επεκτάσιμων και προγραμμάτων. Κάποια από τα εργαλεία που προσφέρει είναι και το .NET Core Rest API template που προσφέρει έναν σκελετό για την ανάπτυξη ενός REST API.

### SPARQL και Ανοικτά Συνδεδεμένα Δεδομένα

Η SPARQL είναι μια γλώσσα διαχείρισης ερωτημάτων για Ανοικτά Συνδεδεμένα Δεδομένα και τα RDF stores. Η SPARQL επιτρέπει στους προγραμματιστές να αξιοποιήσουν Linked Data στις εφαρμογές τους για να αντλήσουν την πληροφόρηση που χρειάζονται. Τα Ανοικτά Συνδεδεμένα Δεδομένα χρησιμοποιούν URIs για την περιγραφή των πόρων τους και ακολουθούν εντελώς διαφορετική λογικά από τα παραδοσιακά δεδομένα σε σχεσιακές βάσεις. Τα δεδομένα περιγράφονται με triplets που ορίζουν τις σχέσεις μεταξύ τους με καθορισμένα λεξικά. Η SPARQL δίνει την δυνατότητα πλήρους αξιοποίησης της παραπάνω αρχιτεκτονικής για την εξαγωγή απαντήσεων σε ερωτήματα που μπορεί να έχει ο χρήστης σε σχέση με τα data που του παρέχει μια βάση.

### 3.1.4 dotNetRDF

Το dotNetRDF είναι μια ανοικτού κώδικα βιβλιοθήκη του .ΝΕΤ που επιτρέπει σε εφαρμογές του να:

1. Διαβάζουν, διαχειρίζονται, να υποβάλλουν ερωτήματα και να γράφουν σε RDF βάσεις δεδομένων
2. Να συνεργάζονται με RDF triple stores όπως το Virtuoso

Επίσης προσφέρει μια σειρά GUI εργαλείων και εντολών για την υποστήριξη λειτουργιών σε RDF μέσω των Windows. Η συγκεκριμένη βιβλιοθήκη περιέχει μια μεγάλη ποικιλία εντολών που περιγράφονται αναλυτικά στο documentation της, δίνοντας με αυτόν τον τρόπο στον προγραμματιστή μεγάλη ελευθερία στην ανάπτυξη εφαρμογών που βασίζονται σε ανοικτά δεδομένα.   
Τέλος, προσφέρει αρκετή υποστήριξη στον προγραμματιστή καθώς υπάρχουν διαθέσιμοι και Οδηγοί Χρήσης, Οδηγοί για προγραμματιστές, documentation για το API της βιβλιοθήκης, ενεργή κοινότητα στο GitHub που βελτιώνει τον κώδικα όταν εντοπίζονται bugs και προσθέτει νέες λειτουργίες.

## 3.2 O σχεδιασμός ενός API

Η βασικότερη λειτουργία του API είναι τα endpoints του, δηλαδή ο κώδικας που αναλαμβάνει να εκτελέσει την επικοινωνία που αιτείται ο χρήστης. Χωρίς την χρήση των endpoints είναι αδύνατη η επικοινωνία του API με άλλα συστήματα, άρα δεν μπορεί να αξιοποιηθεί. Όπως αναφέρεται και στο Κεφάλαιο 2, αυτά τα endpoints ακολουθούν την αρχιτεκτονική REST, δηλαδή μπορούν να εκτελέσουν κλήσεις GET,PUT ,DELETE σε μια απομακρυσμένη τοποθεσία και να λάβουν μια προκαθορισμένη απάντηση. Για να σχεδιαστούν όμως αυτά τα endpoints θα πρέπει να ακολουθηθεί ένα μοντέλο ανάπτυξης που θα μπορεί εύκολα να κατανοηθεί και να επεκταθεί από τρίτους.

### 3.2.1 Το μοντέλο ανάπτυξης MVC

Το Models – Views – Controllers είναι ένα από τα πιο δημοφιλή μοντέλα ανάπτυξης web εφαρμογών. Σύμφωνα με αυτό το μοντέλο η αλληλεπίδραση του χρήστη με μια εφαρμογή μπορεί να διαιρεθεί σε τρία διασυνδεδεμένα μέρη ώστε να είναι πιο ξεκάθαρος ο διαχωρισμός της πληροφόρησης που χρειάζεται ο χρήστης από αυτήν που χρησιμοποιείται από το σύστημα.   
Η εφαρμογή ουσιαστικά χτίζεται σε τρία επίπεδα αυτό του controller

1. Controller, που είναι υπεύθυνος για την επικοινωνία με το/α μοντέλο/α και ενημερώνει τις πληροφορίες τους
2. Model, που ορίζει την μορφή που θα έχει η πληροφορία όταν έρχεται από μια βάση
3. View, που παρουσιάζει με γραφικό τρόπο τις πληροφορίες των μοντέλων

Σε ένα API μπορεί να γίνει χρήση του μοντέλου αυτού για την επικοινωνία του API με τον server και τον χρήστη. Βέβαια η έννοια του view είναι περιττή σε αυτή την περίπτωση όποτε δημιουργείται ένα πιο απλό μοντέλο.

Diagram

Description automatically generated

**Εικόνα 6** Το μοντέλο MVC όπως το προτείνει το REST API Template της Microsoft

Όπως φαίνεται και στο διάγραμμα, ο Controller αναλαμβάνει να δεχτεί το αίτημα που στέλνει ο χρήστης προς το API , να συλλέξει την πληροφορία που χρειάζεται και να προωθήσει στον χρήστη την κατάλληλη απάντηση. Τα Models αναλαμβάνουν να την εννοιολογική χαρτογράφηση του αιτήματος του χρήστη και την μορφοποίηση της πληροφορίας βάση αυτού.  
Επιπλέον για λόγους επεκτασιμότητας και απλοποίησης του κώδικα μπορούν προστεθούν Repositories, κλάσεις δηλαδή που επικοινωνούν με τον server που περιέχει τα δεδομένα που χρειάζονται οι χρήστες. Οι κλάσεις Repository αναλαμβάνουν να στείλουν αιτήματα στον server και να προωθήσουν τις απαντήσεις του στο υπόλοιπο πρόγραμμα. Με αυτόν τον τρόπο διαχωρίζονται λογικά οι λειτουργίες του API και γίνεται πιο εύκολη η επέκτασή του και η κατανόηση του από τρίτους.

## Συμπεράσματα

Κλείνοντας το κεφάλαιο αυτό, γίνεται αντιληπτή η πληθώρα τον εργαλείων και τεχνικών που είναι διαθέσιμες σε ένα προγραμματιστή για την υλοποίηση ενός έργου λογισμικού. Η επιλογή των κατάλληλων εργαλείων είναι πολύ σημαντική απόφαση καθώς καθορίζει πολλές φορές τόσο την ποιότητα, την επεκτασιμότητα και την αξιοπιστία ενός προγράμματος όσο και την αποδοχή και χρήση του από άλλους. Επιπλέον, τα κριτήρια επιλογής μια μεθόδου επίσης ποικίλουν καθώς βασίζονται στην προσωπική εμπειρία του προγραμματιστή, στις τάσεις της αγοράς καθώς και στην καθοδήγηση ή οδηγία που έχει λάβει.

# 4.CPSV-AP API

Το Πανεπιστήμιο Μακεδονίας ως μέρος του δημόσιου τομέα στην Ελλάδα, προσφέρει διάφορες υπηρεσίες στους πολίτες και σε τρίτους οργανισμούς και αντιμετωπίζει τις ίδιες δυσκολίες στον διαμοιρασμό και στην αξιοποίηση των δεδομένων που συλλέγει ή παράγει. Αυτή την στιγμή, εάν κάποιος χρειάζεται να ενημερωθεί για μια υπηρεσία του Πανεπιστημίου, θα πρέπει είτε να επισκεφτεί την ιστοσελίδα του ([www.uom.gr](http://www.uom.gr)) και να αναζητήσει σχετικές αναρτήσεις είτε να επικοινωνήσει με το προσωπικό του Πανεπιστημίου είτε να επισκεφτεί το ίδιο το πανεπιστήμιο. Αυτές οι διαδικασίες όμως μπορούν να αποδειχθούν αρκετά κοστοβόρες ή και αδύνατες.

Η επίσκεψη στην ιστοσελίδα προαπαιτεί από την πλευρά του χρήστη πως θα γνωρίζει ποια υπηρεσία έχει ανάγκη και πως να καθοδηγηθεί εντός της ιστοσελίδας για να βρει αυτό που αναζητεί. Ενώ παράλληλα στην τωρινή έκδοση η ανάγνωση του περιεχομένου της απαιτεί την γνώση ελληνικών ή αγγλικών (με το περιεχόμενο στα αγγλικά να παρουσιάζει σημαντικές ελλείψεις σε σχέση με το αντίστοιχο ελληνικό). Παραδείγματος χάριν τα link στην σελίδα που αφορούν τα «Φοιτητικά Ζητήματα» λειτουργούν μόνο στην ελληνική γλώσσα. Αυτό ενδέχεται να δυσκολέψει αρκετά πολίτες του εξωτερικού που θέλουν να ενημερωθούν για κάποια υπηρεσία του Πανεπιστημίου.

Η επικοινωνία με το προσωπικό απαιτεί την εύρεση κάποιου μέσου επικοινωνίας (email, τηλέφωνο) με αυτό και πως ο πολίτης γνωρίζει από πριν σε ποια υπηρεσία είναι αρμόδια για το αίτημά του. Για παράδειγμα ένας τελειόφοιτος φοιτητής θα χρειαστεί να υποβάλει κάποια δικαιολογητικά για την λήψη του πτυχίου του και άρα να επικοινωνήσει με την γραμματεία του τμήματός του για να ενημερωθεί πιο αναλυτικά στέλνοντας ένα email ή μέσω τηλεφωνικής κλήσης.

Αυτά αποτελούν πολύ απλά παραδείγματα στα οποία γίνεται έκδηλη η ανάγκη τυποποίησης και αυτοματοποίησης της ανταλλαγής πληροφοριών ανάμεσα στο πανεπιστήμιο και σε τρίτους και το πως ένα API βασισμένο σε δεδομένα με την μορφή του CPSV-AP μπορεί να διευκολύνει τόσο τους πολίτες όσο και το προσωπικό ενός δημόσιου οργανισμού.

## 4.1 Η δομή του project CPSV-AP API

Το project του CPSV-API έχει βασιστεί στο .NET REST API Template του .ΝΕΤ Core. Σε αυτό συμπεριλαμβάνονται τα βασικότερα για την ανάπτυξη του project αρχεία:

1. Program.cs
2. Startup.cs

Και οι φάκελοι

1. Controllers
2. Models
3. Repository
4. Properties

Επιπλέον υπάρχει ο φάκελος Dependencies όπου γίνεται η διαχείριση όλων των βιβλιοθηκών που προστίθενται και χρησιμοποιούνται στο Project για να εμπλουτιστούν οι λειτουργίες που προσφέρει το .NET Framework.

Οι κλάσεις Program.cs και Startup.cs περιέχουν τον κώδικα που ορίζει το template που χρησιμοποιήθηκε για να μπορέσει να γίνει εκκίνηση της εφαρμογής.

Σημαντική προσθήκη είναι αυτή των εντολών public void ConfigureServices(IServiceCollection services)

{ services.AddControllers().AddJsonOptions(options =>

{options.JsonSerializerOptions.IgnoreNullValues = true });} όπου το ΑPI απορρίπτει null πεδία στις απαντήσεις που στέλνει στον χρήστη.

**Program.cs**

|  |
| --- |
| namespace CPSV\_API  {  public class Program  {  public static void Main(string[] args)  {  CreateHostBuilder(args).Build().Run();  }  public static IHostBuilder CreateHostBuilder(string[] args) =>  Host.CreateDefaultBuilder(args)  .ConfigureWebHostDefaults(webBuilder =>  {    webBuilder.UseStartup<Startup>();  });  }  } |

**Startup.cs**

|  |
| --- |
| namespace CPSV\_API  {  public class Startup  {  public Startup(IConfiguration configuration)  {  Configuration = configuration;  }  public IConfiguration Configuration { get; }  public void ConfigureServices(IServiceCollection services)  {  services.AddControllers().AddJsonOptions(options =>  {  options.JsonSerializerOptions.IgnoreNullValues = true;  });  }  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  {  if (env.IsDevelopment())  {  app.UseDeveloperExceptionPage();    app.UseHttpsRedirection();  app.UseRouting();  app.UseAuthorization();  app.UseEndpoints(endpoints =>  {  endpoints.MapControllers();  });  }  }  }  } |

**Repositories**

Όσο αναφορά τα Repositories έχει υλοποιηθεί μόνο μια κλάση που αναλαμβάνει να παραμετροποιήσει τις συναρτήσεις που χρησιμοποιούνται από το dotNetRDF για να μπορέσουν να εκτελεστούν τα απαραίτητα SPARQL ερωτήματα. Ως είσοδο, δέχεται ένα string το οποίο προσδιορίζει το SPARQL ερώτημα που τεθεί στην RDF βάση και επιστρέφει τα αποτελέσματα που έχει επιστρέψει το query. Στον κώδικα αναγράφονται τα endpoints και namespaces που απαιτούνται για την εκτέλεση του SPARQL ερωτήματος.

**PublicOrganisationData.cs**

|  |
| --- |
| namespace CPSV\_API.Repositories  {  public static class PublicOrganisationData  {    public static SparqlResultSet fetchData(SparqlParameterizedString queryString)  {  Options.HttpDebugging = true;  SparqlRemoteEndpoint endpoint = new SparqlRemoteEndpoint(new Uri("http://data.dai.uom.gr:8890/sparql"), "http://data.dai.uom.gr:8890/CPSV-AP");  queryString.Namespaces.AddNamespace("cpsv", new Uri("http://purl.org/vocab/cpsv#"));  queryString.Namespaces.AddNamespace("dcterms", new Uri("http://purl.org/dc/terms/"));  queryString.Namespaces.AddNamespace("dct", new Uri("http://purl.org/dc/terms/"));  queryString.Namespaces.AddNamespace("cv", new Uri("http://data.europa.eu/m8g/"));  Console.WriteLine(queryString.ToString());  RemoteQueryProcessor processor = new RemoteQueryProcessor(endpoint);  SparqlQueryParser parser = new SparqlQueryParser();  SparqlQuery query = parser.ParseFromString(queryString);  SparqlResultSet results = (SparqlResultSet)query.Process(processor);  return results;  }  }  } |

**Controllers**

Έχει υλοποιηθεί ένας controller που περιέχει εφτά κλήσεις GET, αυτές οι κλήσεις καθορίστηκαν με βάση τα διαθέσιμα δεδομένα της βάσης του πανεπιστημίου και την πληροφόρηση που δυνητικά θα χρειαζότανε ένας χρήστης.  
Πιο αναλυτικά, οι μέθοδοι του CPSVController είναι:

1. GET allPublicServices: όπου επιστρέφονται όλα τα URI και ονόματα των ΔΥ που είναι διαθέσιμα. Αυτή είναι μια πολύ χρήσιμη κλήση, καθώς ενημερώνει τον χρήστη για όλες τις διαθέσιμες υπηρεσίες ενός δημόσιου οργανισμού.

2. GET PublicService/{id}: όπου επιστρέφονται όλα τα στοιχεία μιας ΔΥ με βάση το CPSV-AP. Με αυτή την κλήση ο χρήστης μπορεί να μάθει λεπτομέρειες για την ΔΥ που τον ενδιαφέρει, αρκεί να γνωρίζει το id της

3. GET allEvidences: όπου επιστρέφονται όλα τα URI και ονόματα των δικαιολογητικών. Με αυτήν την κλήση ο χρήστης μπορεί πολύ εύκολα να ενημερωθεί για τα δικαιολογητικά που ίσως χρειαστεί σε μελλοντικές συναλλαγές του με έναν δημόσιο οργανισμό

4. GET allFormalFrameworks: όπου επιστρέφονται όλα τα URI και ονόματα των νομικών εγγράφων, ενημερώνοντας των χρήστη για όλα τα σχετικά επίσημα έγγραφα που αφορούν τις ΔΥ ενός δημόσιου οργανισμού

5. GET PublicServiceswithFormalFramework/{id}/{id2}

Με αυτή την κλήση επιστρέφονται τα URI και ονόματα όλων των ΔΥ που χρειάζονται ένα νομικό έγγραφο χ. Με την χρήση της κλήσης αυτής ένας χρήστης μπορεί πολύ εύκολα να μάθει που θα του χρειαστεί ένα συγκεκριμένο δικαιολογητικό

6. GET PublicOrganisation   
Με αυτήν την κλήση επιστρέφονται τα διαθέσιμα URI των Δημόσιων Οργανισμών με τις ονομασίες τους. Με αυτήν την κλήση ο χρήστης ενημερώνεται για

7. GET PublicServiceswithEvidence/{id}

Με την κλήση αυτή επιστρέφονται τα URI και ονόματα των ΔΥ που χρειάζονται ένα συγκεκριμένο δικαιολογητικό, ακολουθώντας παρόμοια λογική με την κλήση 5.

**CPSVController.cs**

Οι μέθοδοι που έχουν αναπτυχθεί στον παρακάτω κώδικα ακολουθούν την ίδια λογική ανάπτυξης. Στην αρχή ορίζεται η ονοματολογία της κλήσης και το είδος της απάντησης που θα επιστρέψει το σύστημα δηλαδή αν θα είναι PublicService, FormalFramework κτλ Στην συνέχεια καθορίζεται το SPARQL ερώτημα που υλοποιεί την επιθυμητή απάντηση που περιμένει ο χρήστης από την κάθε κλήση που χρησιμοποιεί. Μετά καλείται κλάση του Repository για να επιστραφούν τα απαραίτητα δεδομένα. Τέλος δημιουργείται ένα στιγμιότυπο του κατάλληλου μοντέλου που μετατρέπει τα SPARQL results σε οντότητες του CPSV-AP για να μπορέσουν να σταλούν πίσω στον χρήστη.

|  |
| --- |
| namespace CPSV\_API.Controllers  {  [Route("api/[controller]")]  [ApiController]  public class CPSVController : ControllerBase  {    [HttpGet("allPublicServices")]  public ActionResult<PublicService> getPublicServices()  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = "SELECT DISTINCT ?PS\_uri ?name {?PS\_uri a cpsv:PublicService. ?PS\_uri dct:title ?name.}";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  List<PublicService> publicServices = new List<PublicService>();  foreach (var result in results)  {  PublicService ps = new PublicService();  ps.createPublicService(result);    publicServices.Add(ps);  }  return Ok(publicServices);  }    [HttpGet("PublicService/{id}")]  public ActionResult<List<string>> getPublicService(string id)  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = "SELECT DISTINCT ?field ?description WHERE {<http://data.dai.uom.gr:8890/PublicServices/id/ps/" + id + "> ?field ?description}";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  PublicService ps = new PublicService();    ps.createPublicService(results, id);  return Ok(ps);  }    [HttpGet("allEvidences")]  public ActionResult<Evidence> getEvidences()  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = "SELECT DISTINCT ?Evidence\_name ?URI {?URI a cv:Evidence. ?URI dct:title ?Evidence\_name.}";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  List<Evidence> Evidences = new List<Evidence>();  foreach (var result in results)  {  Evidence ev = new Evidence();  ev.createEvidence(result);  Evidences.Add(ev);  }  return Ok(Evidences);  }    [HttpGet("allFormalFrameworks")]  public ActionResult<Evidence> getFormalFrameworks()  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = "SELECT DISTINCT ?o ?name {?o a cpsv:FormalFramework. ?o dct:title ?name. }";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  List<FormalFramework> formalFrameworks = new List<FormalFramework>();  foreach (var result in results)  {  FormalFramework legalResource = new FormalFramework();  legalResource.createFormalFramework(result);  formalFrameworks.Add(legalResource);  }    return Ok(formalFrameworks);  }  [HttpGet("PublicServiceswithFormalFramework/{id}/{id2}")]  public ActionResult<PublicService> getPublicServiceswithFormalFrameworks(string id, string id2)  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = " SELECT DISTINCT ?PS\_uri ?name WHERE {?PS\_uri cv:hasFormalFramework <http://data.dai.uom.gr:8890/LegalFramework/id/"+id+"/"+ id2 + ">. ?PS\_uri dct:title ?name.}";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  List<PublicService> publicServices = new List<PublicService>();  foreach (var result in results)  {  PublicService ps = new PublicService();  ps.createPublicService(result);  publicServices.Add(ps);  }  return Ok(publicServices);  }  [HttpGet("PublicOrganisation")]  public ActionResult<PublicOrganisation> getPublicOrganisation()  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  //SPARQ Command that retrieved the specified data  queryString.CommandText = "SELECT DISTINCT ?PO\_uri ?name WHERE {?PO\_uri a cv:PublicOrganisation. ?PO\_uri dct:title ?name}";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  List<PublicOrganisation> publicOrganisations = new List<PublicOrganisation>();    foreach (var result in results )  {  PublicOrganisation po = new PublicOrganisation();  po.createPublicOrganisation(result);  publicOrganisations.Add(po);  }  return Ok(publicOrganisations);  }  [HttpGet("PublicServiceswithEvidence/{id}")]  public ActionResult<PublicService> getPublicServiceswithEvidence(string id)  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = "SELECT DISTINCT ?PS\_uri ?name WHERE {?PS\_uri cpsv:hasInput <http://data.dai.uom.gr:8890/PublicServices/id/doc/" +id+">. ?PS\_uri dct:title ?name.} ";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  List<PublicService> publicServices = new List<PublicService>();  foreach (var result in results)  {  PublicService ps = new PublicService();  ps.createPublicService(result);  publicServices.Add(ps);  }  return Ok(publicServices);  }  [HttpGet("TotalPublicServices")]  public ActionResult<string> getTotal()  {  SparqlParameterizedString queryString = new SparqlParameterizedString();  queryString.CommandText = "Select (count(?ps) as ?Number\_of\_PS) where { ?ps a cpsv:PublicService.} ";  SparqlResultSet results = PublicOrganisationData.fetchData(queryString);  string total = results.ToString();  return Ok(total);  ;  }    }  } |

**Models**

Συνολικά αναπτύχθηκαν τέσσερα μοντέλα βασισμένα στο CPSV-AP. Η υλοποίησή τους ακολούθησε τις ανάγκες και τις προδιαγραφές του CPSV-AP όπως αυτές ορίστηκαν στο Κεφάλαιο 2. Τα μοντέλα που υλοποιήθηκαν είναι τα:

1. PublicService: υλοποιεί μια ΔΥ
2. PublicOrganisation: υλοποιεί έναν δημόσιο οργανισμό
3. Evidence: υλοποιεί τα δικαιολογητικά
4. FormalFramework: υλοποιεί τα νομικά έγγραφα

Στην αρχή κάθε μοντέλου δηλώνονται οι απαραίτητες για αυτό μεταβλητές και στην συνέχεια μια συνάρτηση που αναλαμβάνει την δημιουργία ενός στιγμιότυπου της κλάσης με τις πληροφορίες που θα λάβει από τον controller.  
Για την δημιουργία του στιγμιότυπου απαιτείται η πληροφορία που έρχεται από τα SPARQLResults να διαχωριστεί ανάλογα με το τι αντιπροσωπεύει ώστε να καταχωρηθεί στην αντίστοιχη μεταβλητή της κλάσης του μοντέλου. Αυτή η λειτουργία υλοποιείται κάπως απλοϊκά με την χρήση λογικών που αποδέχονται ή καταρρίπτουν συγκεκριμένα strings όταν θέτουν τιμές σε μια μεταβλητή. Αυτές οι λογικές εφαρμόζονται αν περίπτωση σε κάθε κλάση ανάλογα το μοντέλο και τα δεδομένα μου καλείται να επεξεργαστεί.

Η κλάση PublicService

|  |
| --- |
| public class PublicService  {  public string Uri { get; set; }  public string Name { get; set; }  public string title { get; set; }  public string description { get; set; }  public List<string> Keywords { get; set; }  public List<string> hasInput { get; set; }  public string ProcessingTime { get; set; }    public void createPublicService(SparqlResult result)  {  string temp =result.ToString();  temp = temp.Replace("?PS\_uri =", "");  temp = temp.Replace("?name =", "");  string[] temp2 = temp.Split(",");    if (temp2[0].Contains("PublicServices/id/ps/")) { this.Uri = temp2[0]; this.Name = temp2[1]; } else { this.Uri = temp2[1]; this.Name = temp2[0]; }    }  public void createPublicService(SparqlResultSet results, string id)  {  this.Uri = "http://data.dai.uom.gr:8890/PublicServices/id/ps/" + id;  List<string> inputList = new List<string>();  List<string> keywords = new List<string>();  foreach (var item in results)  {    string temp = item.ToString();  temp = temp.Replace("?field =", "");  temp = temp.Replace("?description =", "");  string[] temp2 = temp.Split(",");  if (temp2[0].Contains("hasInput")) { inputList.Add(temp2[1]); }  if (temp2[0].Contains("dcat#keyword")) { keywords.Add(temp2[1]); }  if (temp2[0].Contains("processingTime")) { this.ProcessingTime = temp2[1]; }  if (temp2[0].Contains("/terms/title")) { this.title = temp2[1]; }  if (temp2[0].Contains("/terms/description")) { this.description = temp2[1]; }  }  if (inputList.Count == 0) { this.hasInput.Add("no values"); } else { this.hasInput = inputList; }  if (keywords.Count == 0) { this.Keywords.Add("no values"); } else { this.Keywords = keywords; }  }  } |

**Η κλάση PublicOrganisation**

|  |
| --- |
| public class PublicOrganisation  {  public string Uri { get; set; }  public string Name { get; set; }  public void createPublicOrganisation(SparqlResult result)  {  string temp = result.ToString();  temp = temp.Replace("?PO\_uri =", "");  temp = temp.Replace("?name =", "");  string[] temp2 = temp.Split(",");  if (temp2[0].Contains("PublicOrganization/")) { this.Uri = temp2[0]; this.Name = temp2[1]; } else { this.Uri = temp2[1]; this.Name = temp2[0]; }  }  } |

**H κλάση FormalFramework**

|  |
| --- |
| public class FormalFramework  {  public string Uri { get; set; }  public string Name { get; set; }  public void createFormalFramework(SparqlResult result)  {  string temp = result.ToString();  temp = temp.Replace("?PS\_uri =", "");  temp = temp.Replace("?name =", "");  string[] temp2 = temp.Split(",");    if (temp2[0].Contains("/LegalFramework/id/")) { this.Uri = temp2[0]; this.Name = temp2[1]; } else { this.Uri = temp2[1]; this.Name = temp2[0]; }  }    } |

**Η κλάση Evidence**

|  |
| --- |
| public class Evidence  {  public string Name { get; set; }  public string Uri { get; set; }  public string Description { get; set; }  public string Language { get; set; }  public string Type { get; set; }  public string RelatedDocumentation { get; set; }  public void createEvidence(SparqlResult results)  {  List<string> inputList = new List<string>();  List<string> keywords = new List<string>();  foreach (var item in results)  {  string temp = item.ToString();  temp = temp.Replace("?field =", "");  temp = temp.Replace("?description =", "");  string[] temp2 = temp.Split(",");  //below depending if the item has a keyword or not the code assigns the relative information to each corresponding field  if (temp2[0].Contains("name")) {this.Name = temp2[1]; }  if (temp2[1].Contains("PublicServices/id/doc/")) { this.Uri = temp2[1]; }  }  }  } |

## Δημοσιεύοντας το CPSV-API

Υπάρχουν πολλοί τρόποι και μέσα για να μπορέσει να δημοσιευθεί μια web εφαρμογή. Αφού ολοκληρωθεί η ανάπτυξη του κώδικα, μια εφαρμογή μπορεί να εγκατασταθεί ή σε ένα ιδιωτικό server ή σε έναν τρίτο πάροδο μέσω τεχνολογιών cloud. Επίσης σε μεγάλες εφαρμογές υπάρχει και επιλογή υβριδικού hosting, όπου ένα μέρος της εφαρμογής φυλάσσεται σε έναν απομακρυσμένο server και ένα άλλο σε έναν ιδιωτικό. Για το CPSV-API επιλέχθηκε το hosting στο cloud μέσω της πλατφόρμας Heroku ώστε να μπορέσουν τα endpoints του να είναι διαθέσιμα προς όλους για χρήση. Η επιλογή ενός παρόχου cloud έχει πολλά οφέλη καθώς ο δημιουργός της εφαρμογής δεν επωμίζεται το κόστος διαχείρισης του server, της εγκατάστασης της εφαρμογής και της εξασφάλισης της ομαλής λειτουργίας της. Επιπροσθέτως, με την χρήση των εργαλείων του κάθε παρόχου μπορεί εύκολα να έχει πλήρη εικόνα για την κατάσταση της εφαρμογής του, το ιστορικό αλλαγών, την δυνατότητα πλήρης διαχείρισης των εκδόσεων της εφαρμογής και κάποια εργαλεία analytics.   
Γνωστότεροι πάροχοι τέτοιων υπηρεσιών είναι η Microsoft με τα προϊόντα Azure, η Amazon με τα Amazon Web Services, Oracle και η Google. Οι συγκεκριμένες εταιρείες έχουν κυριαρχήσει στην αγορά καθώς προσφέρουν μεγάλοι ποικιλία υπηρεσιών καλύπτοντας σχεδόν όλες τις ανάγκες ενός οργανισμού, μιας εταιρείας ή ενός ιδιώτη. Βέβαια για την παροχή αυτών των υπηρεσιών υπάρχει και η αντίστοιχη τιμολόγηση για αυτό και δεν επιλέχθηκαν ως πάροχοι για την εγκατάσταση του CPSV-API.

## 4.2.1 HEROKU

To Ηeroku είναι μια cloud πλατφόρμα που ακολουθεί το μοντέλο PaaS (Platform as a Service) υποστηρίζοντας διάφορες γλώσσες προγραμματισμού. Ιδρύθηκε το 2007 και αγοράστηκε από την Salesforce το 2012. Το Heroku ουσιαστικά παρέχει ένα domain σε ένα application στο οποίο μπορούν να γίνουν αιτήματα HTTP. Η πλατφόρμα υποστηρίζει επίσημα τις εξής γλώσσες προγραμματισμού που ονομάζει buildpacks:

1. Java
2. Node.js
3. Scala
4. Clojure
5. Python
6. PHP
7. Go

Αλλά επίσης μπορεί να επεκταθεί χρησιμοποιώντας πακέτα τρίτων από το Heroku Elements.

**Εγκατάσταση ενός App**

Για να μπορέσει να γίνει deploy σε ένα app θα πρέπει αρχικά να δημιουργηθεί ένας λογαριασμός στο heroku.com, αφού γίνει αυτό, o χρήστης οδηγείται στο Dashboard του Heroku και από εκεί μπορεί να διαχειριστεί τις εφαρμογές του.

Graphical user interface, text, application, Word

Description automatically generated

**Εικόνα 7** Heroku Dashboard

Πατώντας στο New -> Create New App, ξεκινάει η διαδικασία εγκατάστασης μιας εφαρμογής

Στην οθόνη που εμφανίζεται ο χρήστης επιλέγει το όνομα της εφαρμογής του, το οποίο καθορίζει και το όνομα του domain που θα λάβει, καθώς και την περιοχή του. Αν το όνομα είναι διαθέσιμο ο χρήστης μπορεί να συνεχίσει την δημιουργία της εφαρμογής του.

Graphical user interface, text, application, email

Description automatically generated

**Εικόνα 8** Heroku Create New App First Screen

Στην συνέχεια εμφανίζεται το βασικό μενού διαχείρισης της εφαρμογής, όπου αρκετές επιλογές δεν είναι διαθέσιμες στην δωρεάν έκδοση.

Graphical user interface, text, application, email, Teams

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Εικόνα 9** Οθόνη διαχείρισης μιας εφαρμογής στο Heroku

Όπως φαίνεται και στην εικόνα, δίνονται τρεις επιλογές deploy μέσω του

1. Ηeroku Giτ
2. Github
3. . Container Registry

Για το CPSV-API επιλέχθηκε η μέθοδος του GitHub. Για να συνεχιστεί η διαδικασία, θα πρέπει ο χρήστης να έχει έναν λογαριασμό στο Github όπου θα βρίσκεται το αντίστοιχο repository με τον κώδικα που θέλει να ανέβει στο Heroku. Αφού γίνει αυτό, o χρήστης πρέπει να αναζητήσει το repository του, ώστε να μπορέσει το Heroku να το αναγνωρίσει:

Graphical user interface, text, application

Description automatically generated

**Εικόνα 10** Προσθήκη κώδικα από το Github

Αφού βρεθεί, η μόνη κίνηση που απομένει είναι να γίνει Connect.

Graphical user interface, application

Description automatically generated

**Εικόνα 11** Αποτελέσματα αναζήτησης repository

Αυτόματα, αφού πατηθεί το κουμπί Connect το Heroku «στήνει» σε έναν server του. Αν βρεθεί κάποιο σφάλμα στην διαδικασία, εμφανίζει και το αντίστοιχο μήνυμα που ενημερώνει τον χρήστη.  
Αν μια εφαρμογή, όμως δεν είναι υλοποιημένη σε μια από τις γλώσσες που υποστηρίζει η πλατφόρμα, ο προγραμματιστής μπορεί να αξιοποιήσει το Heroku Elements για να βρει τους πόρους που χρειάζεται.  
To Heroku .NET Core Buildback είναι ένα third party buildback που προσφέρεται από τον jincod και επιτρέπει την υλοποίηση .NET εφαρμογών στον Heroku. Για την εγκατάστασή του ο χρήστης θα πρέπει να τρέξει την   
εντολή μέσω command line αφού εγκαταστήσει τον client του Heroku στον υπολογιστή του.

*heroku buildpacks:set jincod/dotnetcore*

Αφού γίνει επιτυχώς η εγκατάσταση του buildback, θα πρέπει να προστεθεί στις ρυθμίσεις του app

Graphical user interface, application

Description automatically generated

**Εικόνα 12** Προσθήκη επιπλέον buildpacks σε μια εφαρμογή

Η προσθήκη γίνεται βάζοντας το link του buildback



και πατώντας το κουμπί Add buildpack.

Τώρα, μπορεί να γίνει η εγκατάσταση της εφαρμογής χωρίς προβλήματα. Αφού εγκατασταθεί, το CPSV-AP είναι έτοιμο να χρησιμοποιηθεί.

**Πρόσβαση**

Ο έλεγχος πρόσβασης γίνεται από την επιλογή Access όπου ο ιδιοκτήτης της εφαρμογής μπορεί να προσθέσει collaborators

**Graphical user interface, text, application, email

Description automatically generated**

**Εικόνα 13** Διαχείριση πρόσβασης στην εφαρμογή

## 4.2.2 GitHub

Όπως είδαμε στην προηγούμενη ενότητα προϋπόθεση για να μπορέσει να γίνει deploy ο κώδικας είναι η δημιουργία ενός repository στο GitHub. Το GitHub είναι μια από τις πιο γνωστές και δημοφιλής πλατφόρμες διαχείρισης κώδικα και versioning.   
Με την εγγραφή στην ιστοσελίδα ο χρήστης μπορεί να αρχίσει να ανεβάζει τα προγραμματιστικά του project στην σελίδα και να συνεργαστεί με τρίτους για την βελτίωση τους. Τα περισσότερα έργα λογισμικού, ιδίως αυτά που αναπτύσσονται στο πλαίσιο μιας ομάδας, το χρησιμοποιούν για να μπορέσουν να κάνουν πλήρη έλεγχο εκδόσεων και προωθήσουν τον κώδικά τους σε τρίτους. Επιπλέον όλα τα σύγχρονα IDEs έχουν plugins που υποστηρίζουν την ενσωμάτωση των projects τους αυτοματοποιημένα σε ένα Github Repository.

To GitHub Repository της παρούσας εργασίας βρίσκεται εδώ:  
[electramakridou/CPSV-API (github.com)](https://github.com/electramakridou/CPSV-API)

.

## 4.3 Δουλεύοντας με το CPSV-API

Από την στιγμή του το API έχει δημοσιευθεί μπορεί πλέον να χρησιμοποιηθεί από τρίτους. Για να αξιοποιήσει κανείς τα endpoint του μπορεί να βρει πολλά εργαλεία, ένα από τα πιο γνωστά, λειτουργικά και φιλικά για τον χρήστη είναι το POSTMAN. Μέσω του Postman μπορεί κανείς χρησιμοποιήσει τα URI του API και να λάβει τις απαντήσεις που επιθυμεί.

Graphical user interface, text, application

Description automatically generatedΓια την χρήση του Postman προ απαιτείται η εγκατάστασή του σε έναν υπολογιστή. Αφού γίνει η εγκατάσταση ο χρήστης μπορεί να εκτελέσει άμεσα κλήσεις σε ένα API χρησιμοποιώντας κλήσεις GET,POST,DELETE.

**Εικόνα 14** Το βασικό παράθυρο εργασίας στο Postman

Αφού διαλέξει την κλήση που επιθυμεί να χρησιμοποιήσει, έστω την

[**https://cpsv-api.herokuapp.com/api/CPSV/allFormalFrameworks**](https://cpsv-api.herokuapp.com/api/CPSV/allFormalFrameworks)

πατώντας Send, επικοινωνεί με το CPSV-API και λαμβάνει την αναμενόμενη απάντηση.

Graphical user interface, text, application

Description automatically generated

**Εικόνα 15** Αποτελέσματα κλήσης στο Postman

Στην οποία ο Postman δίνει την δυνατότητα επιλογής μορφοποίησης ανάλογα με τις προτιμήσεις του χρήστη.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Εικόνα 16** Επιλογές μορφοποίησης αποτελεσμάτων

Αν θα θέλαμε να δούμε τα ίδια data, θα έπρεπε να επισκεφτούμε το SPARQL endpoint του ΠΑΜΑΚ ([Virtuoso SPARQL Query Editor (uom.gr)](http://data.dai.uom.gr:8890/sparql)) και να γράψουμε το αντίστοιχο query σε SPARQL.

Graphical user interface, text, application, email

Description automatically generated

**Εικόνα 17** Το περιβάλλον εργασίας του Virtuoso

Table

Description automatically generated

**Εικόνα 18** Αποτελέσματα ενός query στο Virtuoso

Με την χρήση του API αυτή η διαδικασία απλοποιείται, καθώς ο χρήστης απλά θα πρέπει να γνωρίζεις τις κλήσεις του API, χωρίς να υπάρχει ανάγκη για περισσότερες τεχνικές γνώσεις. Σε αντίθεση η χρήση του Virtuoso προϋποθέτει μια βασική εξοικείωση με την SPARQL για την δημιουργία απλών ερωτημάτων, την γνώση του μοντέλου CPSV-AP ώστε να γίνει σωστή χρήση των prefixes και των identifiers, επίσης είναι δύσκολη η περαιτέρω αξιοποίηση των αποτελεσμάτων που εμφανίζονται στον χρήστη αφού δεν μπορούν να γίνουν export σε machine readable format.

## Συμπεράσματα

Η υλοποίηση ενός έργου λογισμικού αποδεικνύεται πως αν και έχει απλοποιηθεί ως διαδικασία, παραμένει απαιτητική, καθώς ο προγραμματιστής καλείται να βρει λύσεις σε αρκετά ζητήματα που προκύπτουν κατά την ανάπτυξη του έργου, αναζητώντας διεξοδικά στους πόρους που του είναι διαθέσιμοι. Επίσης καλείται να βρει τις βέλτιστες τακτικές υλοποίησης των ιδεών ώστε να παράγει ένα αξιόλογο αποτέλεσμα. Όσο αναφορά την παρούσα εργασία, ό,τι ζητήματα προέκυψαν μπόρεσαν να αντιμετωπιστούν επιτυχώς ώστε να δημιουργηθεί ένα πλήρως λειτουργικό, πειραματικό API με κάποιους βέβαια περιορισμούς.

# 5. Συμπεράσματα

Έπειτα από την μελέτη που έγινε για την εκπόνηση αυτής της εργασία, μπορούν να αντληθούν αρκετά συμπεράσματα, τόσο για τον τρόπο που διαχειρίζεται η ΕΕ τα προγράμματα ενίσχυσης της ηλεκτρονικής διακυβέρνησης, όσο και για τους περιορισμού που υπάρχουν για την ην περαιτέρω ανάπτυξη του CPSV-API και την υιοθέτησή του από τρίτους

## Δράσεις της Ευρωπαϊκής Ένωσης

Όσο αναφορά τα προγράμματα και τις δράσεις της ΕΕ, φαίνεται πως έχουν γίνει σημαντικά βήματα στον εκσυγχρονισμό των δημοσίων διοικήσεων και στην ψηφιοποίηση των υπηρεσιών τους, παρά ταύτα τα κράτη μέλη της κινούνται με διαφορετικές ταχύτητες όσο αναφορά την υιοθέτηση των νέων αυτών στρατηγικών. Αυτό γίνεται φανερό από το γεγονός πως από την αρχή υλοποίησης του έως και σήμερα το CPSV-AP υιοθετήθηκε και αξιοποιήθηκε από μικρό αριθμό δημόσιων οργανισμών στην Ευρώπη. Αυτό δημιουργεί ζητήματα, καθώς άμεση απόρροια αυτού του γεγονότος είναι η περιορισμένη ανατροφοδότηση που θα δεχτεί η ΕΕ για την βελτίωσή του. Παράλληλα, φαίνεται πως οι δράσεις της ΕΕ απασχολούν κυρίως δημόσιους και κυβερνητικούς φορείς, με τους επαγγελματίες ιδιώτες του κλάδου να ασχολούνται με άλλα project. Αυτό είναι σημαντικό ζήτημα καθότι τα έργα που καλείται να υλοποιήσει η ΕΕ για να πετύχει την διαλειτουργικότητα ανάμεσα στις δημόσιες υπηρεσίες της, θα χρειαστεί να χρησιμοποιήσουν εργαλεία από την αγορά των ΤΠΕ στα οποία εξειδικεύονται αρκετοί επαγγελματίες στην Ευρώπη. Αυτή η εξειδίκευση μπορεί να βοηθήσει σημαντικά στην υλοποίηση αξιόπιστων και χρήσιμων εφαρμογών στην Ευρώπη.

## Περιορισμοί στο API

Το API που αναπτύχθηκε στα πλαίσια αυτής της εργασίας αν και λειτουργικό ως έναν βαθμό, έχει κάποιος σημαντικούς περιορισμούς. Αρχικά οι κλήσεις που υλοποιήθηκαν είναι μόνο της μορφής GET, καθώς η οποιαδήποτε τροποποίηση του RDF store του Πανεπιστημίου Μακεδονίας προϋποθέτει μεγαλύτερη εξειδίκευση τόσο στην SPARQL όσο και στην βιβλιοθήκη που χρησιμοποιήθηκε dotNetRDF, αυξάνοντας την πολυπλοκότητα υλοποίησης του έργου, χωρίς την παροχή κάποιας άμεσης αξίας. Επιπλέον, τα δεδομένα που έχει αυτή τη στιγμή στην διάθεση του το RDF του πανεπιστημίου είναι ενδεικτικά και χρησιμοποιούνται για πειραματικούς σκοπούς, αυτό σημαίνει πως είναι αρκετά πιο δύσκολη η ανάπτυξη μιας πλήρους εφαρμογής αφού ίσως να υπάρχουν ελλείψεις σε δεδομένα για να υλοποιηθούν τα απαραίτητα test. Επιπροσθέτως, δεν υπάρχουν γενικότερα πολλά rdf με δεδομένα που να ακολουθούν το μοντέλο CPSV-AP με αποτέλεσμα να είναι πιο δύσκολη η πλήρης κατανόηση του τρόπου που έχει δομηθεί το παρόν μοντέλο

## Βελτιώσεις στο API

Tο παρόν API έχει σχεδιαστεί ώστε να μπορεί να κατανοηθεί εύκολα από τρίτους. Ο κώδικας είναι όσο το δυνατόν πιο περιεκτικός, ακολουθώντας σύγχρονα πρότυπα ανάπτυξης ώστε να μπορεί να επεκταθεί χωρίς ιδιαίτερα προβλήματα από τρίτους. Ήδη υπάρχουν κάποιες τεχνικές βελτιώσεις που μπορούν να γίνουν όπως η προσθήκη error handling σε μη υποστηριζόμενα αιτήματα, η προσθήκη logging για καταγραφή του τρόπου που συμπεριφέρεται η εφαρμογή και η δημιουργία ενός documentation στηριζόμενο στον κώδικα που υλοποιήθηκε (πέραν του σχολιασμού του στην εργασία). Τέλος προς το παρόν στο API έχουν προστεθεί μόνο οι κλάσεις του CPSV-AP που χρησιμοποιούνται από τις κλήσεις που έχουν υλοποιηθεί, αυτό σημαίνει πως μελλοντικά θα πρέπει να προστεθούν και άλλα μοντέλα (εκτός από κλήσεις) ώστε το API να εξυπηρετεί περισσότερες ανάγκες.

## Προτάσεις

Κλείνοντας την παρούσα εργασία, γίνεται ακόμη πιο φανερή η ανάγκη για ενημέρωση και δραστηριοποίηση από τους πολίτες όσο αναφορά τις δράσεις της Ευρωπαϊκής Ένωσης και την δημιουργία οργανωμένων ομάδων για την υλοποίηση συγκεκριμένων έργων με την υποστήριξη των τοπικών κυβερνητικών φορέων. Η δημιουργία του παρόντος API αποδεικνύει πως η ευαισθητοποίηση των πολιτών και της επιστημονικής κοινότητας σχετικά με τις δράσεις της Ευρωπαϊκής Ένωσης μπορεί να οδηγήσει στην ανάπτυξη αρκετά χρήσιμων εργαλείων που θα μπορούν να διευκολύνουν την καθημερινότητα των πολιτών. Όπως και με την παρούσα εργασία, έτσι και σε μελλοντικά έργα δεν αποκλείεται και η επέκτασή τους από τρίτους που θα

# 6. ΒΙΒΛΙΟΓΡΑΦΙΑ

## 6.1 Επίσημα κείμενα στο διαδίκτυο

ISA2 programme Interoperability Solutions for European Public Administrations, Businesses and Citizens. Core Vocabularies Simplified, reusable and extensible data models Available at:  
[Core Vocabularies – Simplified, reusable and extensible data models (europa.eu)](https://ec.europa.eu/isa2/sites/default/files/docs/publications/2017_core_vocabularies_leaflet_lr_v11.pdf)

Publications Office of the European Union. (2015) e-Government Core Vocabularies handbook Available at:  
[e-government\_core\_vocabularies\_handbook.pdf (europa.eu)](https://ec.europa.eu/isa2/sites/default/files/e-government_core_vocabularies_handbook.pdf)

European Commission. (2010). The European eGovernment Action Plan 2011-2015 Harnessing ICT to promote smart, sustainable & innovative Government Available at:   
[EN (europa.eu)](https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0743:FIN:EN:PDF)

European Commission. (2017). European Interoperability Framework – Implementation Strategy Available at:   
[resource.html (europa.eu)](https://eur-lex.europa.eu/resource.html?uri=cellar:2c2f2554-0faf-11e7-8a35-01aa75ed71a1.0017.02/DOC_1&format=PDF)

Deloitte. (2015). API economy: Tech Trends 2015: The fusion of business and IT A public sector perspective Available at:  
[API economy | Deloitte | Public Sector](https://www2.deloitte.com/gr/en/pages/public-sector/articles/gx-public-sector-tech-trends-api-economy-report.html)

## Άρθρα

Ed Downey, Matthew A. Jones, Kathryn Kloby .(2012) From E-Government to E-Governance. doi: [10.4018/978-1-4666-0071-3.ch015](http://dx.doi.org/10.4018/978-1-4666-0071-3.ch015)

Hans Solli-Sæther, Leif S Flak. (2012). Interoperability, maturity and benefits in e-government

Åke Grönlund, Thomas A Horan. (2004) Introducing e-Gov: History, Definitions, and Issues

## 6.3 Δημοσιευμένες Εργασίες

Ζαρκάδα Ειρήνη (2015).“Οι Δημόσιες Υπηρεσίες μέσα από τα Περιστατικά Ζωής” Πτυχιακή Εργασία. Πανεπιστήμιο Μακεδονίας  
Σωτηρίου Αγγελή (2016). «Σχεδίαση και υλοποίηση εργαλείου αυτόματης ανάπτυξης προσαρμόσιμων διεπαφών χρήστη για RESTful web APIs» Διπλωματική Εργασία. Αριστοτέλιο Πανεπιστήμιο Θεσσαλονίκης  
Κωνσταντίνος Κοροβέσης, Χρυσόστομος Μανώλης (2016) «Σχεδιασμός και Ανάπτυξη Διαδικτυακών Υπηρεσιών Αρχιτεκτονικής REST» Πανεπιστήμιο Δυτικής Αττική

Fielding, Roy Thomas (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine.

## 6.4 Βιβλία

Leonard Richardson ,Mike Amundsen ,Sam Ruby. (2013). RESTful Web APIs: Services for a Changing World 1st Edition. O’Reilly Media.  
Mark Masse. (2011). REST API Design Rulebook. O’Reilly Media.  
Drayton, Peter; Albahari, Ben; Neward, Ted (2002). C# Language Pocket Reference. O'Reilly.

Καλογήρου, Γ., Παναγιωτόπουλος, Π. 2016. Ηλεκτρονική Διακυβέρνηση. [Κεφάλαιο 5]. Στο Καλογήρου, Γ., Παναγιωτόπουλος, Π., Τσακανίκας, Ά., Σιώκας, Ε., Καρούνος, Θ., Μάγκλαρης, Β., Τρούλος, Κ., Καλογεράς, Δ., Τσιαβός, Π., Κανέλλος, Ν., Μερεκούλιας, Β. 2016. *Κοινωνία της πληροφορίας και οικονομία της γνώσης*. [ηλεκτρ. βιβλ.] Αθήνα:Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών Διαθέσιμο στο: <http://hdl.handle.net/11419/6210>

## 6.5 Ιστοσελίδες

[ASP.NET Web APIs | Rest APIs with .NET and C# (microsoft.com)](https://dotnet.microsoft.com/apps/aspnet/apis)

[.NET | Free. Cross-platform. Open Source. (microsoft.com)](https://dotnet.microsoft.com/)

[ASP.NET MVC Pattern | .NET (microsoft.com)](https://dotnet.microsoft.com/apps/aspnet/mvc)

[C# docs - get started, tutorials, reference. | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/csharp/)

[dotNetRDF](https://dotnetrdf.org/)

[GitHub](https://github.com/)

[Heroku | Login](https://id.heroku.com/login)

[SPARQL Query Language for RDF (w3.org)](https://www.w3.org/TR/rdf-sparql-query/)

# ΠΑΡΑΡΤΗΜΑΤΑ

## Α. Χρήσιμα links

Οδηγός ενσωμάτωσης VS Studio projects στο Github:

[Get Started with Git and GitHub in Visual Studio - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/modules/visual-studio-github-push/)  
Online SPARQL editor της Ευρωπαϊκής Ένωσης  
[Advanced SPARQL Query Editor - Publications Office of the EU (europa.eu)](https://op.europa.eu/en/advanced-sparql-query-editor)  
Οδηγός διαχείρισης repositories στο Github  
[Hello World · GitHub Guides](https://guides.github.com/activities/hello-world/)

SPARQL endpoint του Πανεπιστημίου Μακεδονίας  
[Virtuoso SPARQL Query Editor (uom.gr)](http://data.dai.uom.gr:8890/sparql)