Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά Συστήματα

Υλοποίηση Υπηρεσιών Ιστού για σκοπούς ηλεκτρονικής μάθησης

Σοφία Κουτεντάκη

Επιβλέπων Καθηγητής
Θανάσης Χατζηλάκος

Αύγουστος 2013

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Υλοποίηση Υπηρεσιών Ιστού για σκοπούς ηλεκτρονικής μάθησης

Σοφία Κουτεντάκη

Επιβλέπων Καθηγητής
Θανάσης Χατζηλάκος

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Αύγουστος 2013

Περίληψη

Στην παρούσα διπλωματική εργασία προτείνουμε μια service-oriented αρχιτεκτονική τύπου REST για πρόσβαση σε υπηρεσίες μάθησης και των συστατικών των πόρων τους από Web-based περιβάλλοντα. Στόχος της εν λόγω αρχιτεκτονικής είναι η αύξηση της διαλειτουργικότητας, και η επικοινωνία των εμπλεκόμενων πλατφορμών ηλεκτρονικής μάθησης με δυνατότητα ασφαλούς διασύνδεσης και ανταλλαγής δεδομένων και πληροφοριών. Η διαδικτυακή υπηρεσία υλοποιείται σαν μια διαδικτυακή εφαρμογή εύκολα προσβάσιμη από κάθε ενδιαφερόμενο και επιπλέον προσφέρεται και σαν μια διαδικτυακή υπηρεσία τύπου REST για να μπορεί εύκολα να χρησιμοποιηθεί από διαφορετικές εφαρμογές πελάτη. Η υλοποίηση της συγκεκριμένης υπηρεσίας έγινε με τη χρήση της PHP γλώσσας προγραμματισμού διαδικτύου.

Για την εφαρμογή αυτής της υπηρεσίας ιστού, χρησιμοποιήθηκαν ελεύθερα λογισμικά διαχείρισης εκπαιδευτικού περιεχομένου όπως είναι τα συστήματα διανομής της εκπαίδευσης (Moodle) και εργαλεία συνεργατικής συγγραφής (MediaWiki) καθώς και εκείνα που χρησιμοποιούνται ως ιστολόγια (WordPress). Επίσης, περιγράφηκε η προτεινόμενη μεθοδολογία που αφορά τη σχεδίαση και ανάπτυξη μιας υπηρεσίας ασφαλούς διασύνδεσης και ανταλλαγής δεδομένων. Η μεθοδολογία αυτή, εφαρμόζεται σε ενοποιημένες λειτουργίες μιας υπηρεσίας Moodle από τη διεπαφή του χρήστη της μαθησιακής δραστηριότητας και επικυρώνεται σε πραγματικές υπηρεσίες του MediaWiki και WordPress που βασίζονται σε διεπαφές που χρησιμοποιούν REST πρωτόκολλα.

Λέξεις κλειδιά :

Παγκόσμιος Ιστός (Web 2.0), Υπηρεσίες Ιστού, REST, μάθηση με τη βοήθεια υπολογιστή, προσαρμογή, πιστοποίηση, εξουσιοδότηση, ασφάλεια, πρωτόκολλο OAuth 2.0, σχεδίαση προσανατολισμένη σε υπηρεσίες, Moodle, MediaWiki, WordPress.

Summary

In this paper we propose a service-oriented architecture type REST for accessing in services of learning components and their resources from Web-based environments. The recommended architecture aims to increase the interoperability and communication of involved e-learning platform, enabling secure connectivity and data exchange and information. The web service is implemented as a web application easily accessible from all concerned and also offered as a web service in REST type that can easily be used by different client applications. The service was implemented in PHP.

To implement this web service, used freely content management software such as educational content distribution systems of education (Moodle) and collaborative authoring tools (MediaWiki) and those used as blogs (WordPress). A methodology for the design and development of a service for secured interface and data exchange is also described. The proposed methodology is applied to integrating operations of a Moodle service in the client interface of a learning activity and validated on actual MediaWiki and WordPress services providing REST-based programming interfaces.

Keywords:

World Wide Web (Web 2.0), Web Services, REST, computer-assisted instruction, adaptation, authentication, authorization, secure, OAuth 2.0 protocol, service orientation, Moodle, MediaWiki, WordPress.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Θανάση Χατζηλάκο, για το ενδιαφέρον που έδειξε και αξιολόγησε την προσπάθεια μου, καθώς και την υποψήφια διδάκτορα κα. Άννα Μαυρουδή για τις χρήσιμες συμβουλές και την υποστήριξη της καθ' όλη τη διάρκεια υλοποίησης αυτής της μεταπτυχιακής διατριβής.

Θα ήθελα επίσης να ευχαριστήσω τους ερευνητές Juan Manuel Dodero και Ernie Ghiglione για τις κατευθυντήριες οδηγίες που μου πρόσφεραν στην σχεδίαση και ανάπτυξη της εν λόγω υπηρεσίας, στηριζόμενη στο δημοσιευμένο paper τους με τίτλο: «ReST-Based Web Access to Learning Design Services. IEEE Transactions On Learning Technologies [1]».

Τέλος, ένα μεγάλο ευχαριστώ χρωστώ στην οικογένεια μου για τη στήριξη καθ' όλη τη διάρκεια των σπουδών μου δείχνοντας κατανόηση και υπομονή. Επίσης, ευχαριστώ θερμά τα αγαπημένα μου πρόσωπα για την ηθική συμπαράσταση που μου πρόσφεραν.

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

Τα τελευταία χρόνια παρουσιάζεται έντονο ενδιαφέρον και δραστηριότητα στην ανάδειξη μιας νέας γενιάς του Παγκόσμιου Ιστού που βασίζεται σε συσχετιζόμενες τεχνολογίες και πρότυπα του Διαδικτύου. Αυτή η νέα γενιά ονομάζεται Web 2.0 και στόχο έχει την ανάπτυξη του Web ως πλατφόρμα. Στο πλαίσιο αυτής της κατεύθυνσης, οι υπηρεσίες «τρέχουν» στο παράθυρο ενός φυλλομετρητή (browser) και η επικοινωνία επιτυγχάνεται μέσω διαδικτύου και απομακρυσμένων εξυπηρετητών [2]. Στο Web 2.0, το βασικό στοιχείο είναι ο χρήστης και κύριος στόχος του είναι η παροχή φιλικών προς το χρήστη διεπαφών (User Interfaces). Οι τεχνολογίες και πρότυπα που χρησιμοποιούνται προορίζονται προς αυτή τη κατεύθυνση, προσφέροντας στους χρήστες τη δυνατότητα να μοιράζονται πληροφορίες και να συνεργάζονται online.

Μία από τις κύριες συνέπειες του Web ως πλατφόρμα, είναι ότι υπάρχει λιγότερη έμφαση στο λογισμικό, και πολύ περισσότερη σε μια εφαρμογή που παρέχει μια υπηρεσία. Σημαντικό ρόλο στο τομέα της ανάπτυξης των υπηρεσιών, διαδραματίζει η ανάπτυξη της εφαρμογής μιας service-oriented αρχιτεκτονικής, δηλαδή μιας αρχιτεκτονικής προσανατολιζόμενης σε υπηρεσίες. Ουσιαστικά, η αρχιτεκτονική αυτή, είναι μια συλλογή από υπηρεσίες, οι οποίες προσαρμόζονται ευκολότερα στις ανάγκες των χρηστών και παράλληλα προσφέρουν ευελιξία στους φορείς που παρέχουν αυτές τις υπηρεσίες [2, 3].

Στις μέρες μας, ολοένα και περισσότερες επιχειρήσεις προσανατολίζονται προς αυτή τη κατεύθυνση με σκοπό τη δημιουργία υπηρεσιών Ιστού για την παροχή των προσφερόμενων υπηρεσιών τους. Εκτός όμως από τις επιχειρήσεις, οι υπηρεσίες Ιστού έχουν εισχωρήσει και στο χώρο της εκπαίδευσης. Πολλά από τα υπάρχοντα περιβάλλοντα ηλεκτονικής μάθησης (Moodle, e-class[1]κ.α.) παρέχουν αναπόσπαστο κομμάτι των υπηρεσιών που προσφέρουν τα Πανεπιστήμια, γιατί παρέχουν πληθώρα εργαλείων και εφαρμογών για κοινή χρήση μαθησιακού περιεχομένου και εκτεταμένη αλληλεπίδραση συμμετεχόντων. Όμως, παρουσιάζουν μειονεκτήματα όσον αφορά την επεκτασιμότητα και την ανταλλαγή περιεχομένου. Τη λύση στο πρόβλημα αυτό, έρχεται να δώσουν οι υπηρεσίες Ιστού, λόγω της δυνατότητας που έχουν στη παροχή διαλειτουργικότητας διευκολύνοντας την αλληλεπίδραση μεταξύ διαφορετικών εφαρμογών [3].

Έτσι, σε αυτήν την εργασία, ως βασικός πυρήνας για την εφαρμογή των υπηρεσιών Ιστού, θα χρησιμοποιηθεί ένα παραδοσιακό σύστημα διαχείρισης μάθησης όπως ειναι το Moodle. Η επιλογή αυτού του συστήματος έγινε λόγω του ότι, ναι μεν χρησιμοποιείται από μεγάλους οργανισμούς σε Ελλάδα και εξωτερικό όπως είναι το MIT, το Yale και πολλά Πανεπιστήμια του εσωτερικού και εξωτερικού, αλλά και λόγω της συμβατότητας που προσφέρει με διαφορετικές πλατφόρμες. Επίσης, θα χρησιμοποιηθεί η πλατφόρμα συνεργατικής συγγραφής MediaWiki γιατί προσφέρει συνεργασία, ενισχύει τις άτυπες συζητήσεις μεταξύ των μελών και είναι εύκολο και διαδραστικό. Τέλος, θα χρησιμοποιηθεί η πλατφόρμα ενός ιστολογίου όπως είναι το WordPress, γιατί είναι ένα ευρέως αποδεκτό ανοικτού κώδικα σύστημα διαχείρισης περιεχομένου που προσφέρει λειτουργικότητα και δυνατότητες τροποποίησης.

[1] http://www.eclass.gunet.gr

1.1 Κίνητρο

Η ανάγκη για διαλειτουργικότητα των διαφόρων συστημάτων διανομής της εκπαίδευσης μέσω Διαδικτύου όπως είναι το Moodle, που αναπτύχθηκαν τελευταία και εφαρμόζονται στο χώρο της εκπαίδευσης, οδήγησε στην απαίτηση της συμβατότητας και της συνεργασίας με ετερογενής διαδικτυακά συστήματα. Λόγω των πολλών δυνατοτήτων που προσφέρουν στην εκπαιδευτική κοινότητα έχουν καθιερωθεί ως απαραίτητα εργαλεία μάθησης διευκολύνωντας τη συνεργασία, και τον προγραμματισμό. Η χρησιμοποίηση της τεχνολογίας των υπηρεσιών Ιστού με απλό, σαφή και εύχρηστο τρόπο, παρέχει το κλειδί για μια κατανεμημένη προσέγγιση για την ενσωμάτωση τέτοιων υπηρεσιών σε ετερογενή διαδικτυακά συστήματα.

1.2 Σκοπός και Στόχοι

Σκοπός αυτής της μεταπτυχιακής διατριβής, είναι η ανάπτυξη μιας υπηρεσίας Ιστού βασισμένη σε σχεδίαση προσανατολισμένη σε υπηρεσίες, για την διευκόλυνση εκπαιδευτικών δραστηριοτήτων ηλεκτρονικής μάθησης. Για την υλοποίηση των υπηρεσιών Ιστού χρησιμοποιούνται κατά βάση δύο ειδών τεχνολογίες. Η πρώτη βασίζεται στο πρωτόκολλο SOAP (Simple Object Access Protocol) και η δεύτερη στο πρωτόκολλο REST (Representative State Transfer). Το SOAP αποτελεί την παραδοσιακή μέθοδο στην υλοποίηση υπηρεσιών Ιστού σχεδιασμένο για κατανεμημένα συστήματα, ενώ το REST για διαδικτυακά περιβάλλοντα προσφέροντας ευελιξία και προσαρμοστικότητα. Δεν απαιτεί την ύπαρξη πολύπλοκων προτύπων σε σχέση με το SOAP, με αποτέλεσμα να προσφέρει πλεονεκτήματα στην απόδοση των υπηρεσιών Ιστού. Εξαιτίας αυτών των πλεονεκτημάτων, στην εν λόγω εργασία, προτείνεται μια αρχιτεκτονική βασισμένη στο πρωτόκολλο REST ως πιο σύγχρονο και πιο απλό πρωτόκολλο για επιτυχή πρόσβαση σε υπηρεσίες ηλεκτρονικής μάθησης από εφαρμογές βασισμένες σε διαδικτυακά περιβάλλοντα.

Θα προσπαθήσουμε να σχεδιάσμουμε και να υλοποιήσουμε μια υπηρεσία Ιστού τύπου REST σε επίπεδο πελάτη/εξυπηρετητή (client/server) η οποία θα παρέχει μία απλή διεπαφή επικοινωνίας. Μέσω αυτής της υπηρεσίας θα επιτευχθεί η διασύνδεση της πλατφόρμας Moodle με τις πλατφόρμες MediaWiki και WordPress με σκοπό την ανταλλαγή δεδομένων μεταξύ τους, αφού πρώτα προηγηθεί η πιστοποίηση των χρηστών του Moodle στις άλλες πλατφόρμες.

Επιπλέον, θα προσπαθήσουμε να απαντήσουμε στα παρακάτω ερωτήματα ερευνητικού ενδιαφέροντος :

- Πως μπορεί να επιτευχθεί η διασύνδεση ετερογενών διαδικτυακών πλατφορμών με σκοπό την ανταλλαγή δεδομένων;
- Μπορεί ένας χρήστης μιας διαδικτυακής πλατφόρμας μάθησης να έχει πρόσβαση σε άλλες ανοικτού κώδικα πλατφόρμες όπως αυτής του Ιστολογίου και αυτής της συνεργαστικής μάθησης, χωρίς την απαίτηση δημιουργίας του χρήστη σε αυτές;

1.3 Δομή της Μεταπτυχιακής Διατριβής

Στο **Κεφάλαιο 1** αναλύονται το αντικείμενο της έρευνας, τα κίνητρα, οι στόχοι και ο σκοπός της μεταπτυχιακής διατριβής.

Στο **Κεφάλαιο 2** παρουσιάζονται οι βασικές τεχνολογίες/πρότυπα των υπηρεσιών Ιστού που χρησιμοποιούνται για την ανάπτυξη και υλοποίηση μιας υπηρεσίας Ιστού.

Στο **Κεφάλαιο 3** γίνεται μια ανασκόπηση της βιβλιογραφίας όσον αφορά τις υπηρεσίες Ιστού (Web Services), τα βασικά χαρακτηριστικά τους και τις λειτουργίες που προσφέρουν. Επίσης γίνεται αναφορά στο μοντέλο και στην αρχιτεκτονική στοίβα των υπηρεσιών Διαδικτύου.

Στο **Κεφάλαιο 4** προσδιορίζεται η μεθοδολογία που ακολουθήθηκε σε αυτή την εργασία καθώς επίσης, γίνεται αναφορά σε ορισμούς που περιγράφουν την σημασία του πρωτοκόλλου OAuth 2.0 και των web APIs γενικά, αλλά και το ρόλο τους στην αρχιτεκτονική σχεδίαση και υλοποίηση της εφαρμογής.

Στο **Κεφάλαιο 5** περιγράφονται τα σενάρια διασύνδεσης που πρέπει να ακολουθηθούν για την επιτυχή αλληλεπίδραση των εμπλεκόμενων πλατφορμών και την ανταλλαγή των δεδομένων μεταξύ τους. Σε αυτά, καθορίζονται και ρυθμίσεις των RESTful υπηρεσιών (δηλαδή υπηρεσιών που βασίζονται στο αρχιτεκτονικό στυλ REST το οποίο καθορίζει μια ενιαία διεπαφή με σκοπό την καλύτερη λειτουργία τους στο Διαδίκτυο).

Στο **Κεφάλαιο 6** παρουσιάζονται τα συμπεράσματα που εξάγαμε από την σχεδίαση και χρήση της διαδικτυακή υπηρεσίας καθώς και η μελλοντική εργασία που απαιτείται για την διεύρυνση αυτής.

Στο **Παράρτημα Α** παρουσιάζονται ενδεικτικά παραδείγματα μεγάλης έκτασης που βοήθησαν στην κατανόηση κάποιων τεχνολογιών.

Στο **Παράρτημα Β** παρουσιάζονται στιγμιότυπα οθονών από την εγκατάσταση των διαδικτυακών πλατφορμών Moodle, MediaWiki και WordPress που έλαβαν μέρος στην υλοποίησης της εφαρμογής.

Κεφάλαιο 2
Τεχνολογίες υλοποίησης των Web Services

Σε αυτό το κεφάλαιο περιγράφονται με αναλυτικό τρόπο οι τεχνολογίες που χρησιμοποιούνται για την υλοποίηση των υπηρεσιών Ιστού.

2.1 XML (Extensible Markup Language)

Η XML είναι μια απλή και παγκοσμίως αποδεκτή γλώσσα σήμανσης, που επιτρέπει την δομημένη αναπαράσταση των δεδομένων του Παγκόσμιου Ιστού, χρησιμοποιώντας XML έγγραφα. Παρόλου που αρχικά σχεδιάστηκε για να ανταποκριθεί στις απαιτήσεις των μεγάλων παρόχων περιεχομένου στο Διαδίκτυο, διαδραματίζει ολοένα και σημαντικότερο ρόλο στην ανταλλαγή των δεδομένων στον Παγκόσμιο Ιστό. Αναπτύχθηκε από το W3C's XML Working Group το 1996 ως επέκταση της SGML (Standard Generalized Markup Language) γλώσσας σήμανσης, η οποία ορίζει απλούς συντακτικούς κανόνες για την περιγραφή των δεδομένων της.

Ως γλώσσα σήμανσης (markup), η XML χρησιμοποιεί ετικέτες (tags) "<></>" για την περιγραφή των δεδομένων της. Όπως δηλώνει και ο ορισμός της (extensible), η XML είναι μια επεκτάσιμη γλώσσα, γιατί επιτρέπει στον χρήστη να ορίσει τα δικά του στοιχεία (elements) και γνωρίσματα (attributes) για την περιγραφή οποιουδήποτε τύπου δεδομένων (data). Κάθε στοιχείο μπορεί να αποτελείται απο ένα ή περισσότερα γνωρίσματα όπου σαν σκοπό έχουν να δώσουν περισσότερες λεπτομέρειες στην περιγραφή των δεδομένων. Δεν εξαρτάται από καμία συγκεκριμένη τεχνολογία και χρησιμοποιεί συνήθως το πρωτόκολλο HTTP[2] για την επικοινωνία των XML μηνυμάτων που ανταλλάσσονται μέσω του Διαδικτύου [4].

Ένα XML έγγραφο είναι σωστό, εάν είναι καλά σχηματισμένο (well formed) και έγκυρο (valid).

Για να ελέγξουμε εάν ένα XML έγγραφο είναι καλά σχηματισμένο, η XML χρησιμοποιεί έναν αναλυτή (XML parser), μια εφαρμογή δηλαδή, η οποία έχει πρόσβαση στη δομή και το περιεχόμενο ενός εγγράφου. Ένα καλά σχηματισμένο XML έγγραφο είναι αυτό που ακολουθεί την XML σύνταξη, και μπορεί να είναι εντελώς επεξεργάσιμο από ένα XML parser. Εάν βρεθούν σφάλματα σύνταξης του εγγράφου, τότε δεν θεωρείται XML έγγραφο και ο XML parser απορρίπτει ολόκληρο το έγγραφο.

Για την εγκυρότητα ενός εγγράφου, η XML χρησιμοποιεί ένα DTD (Document Type Definition), έγγραφο το οποίο περιγράφει τη δομή ενός έγκυρου XML εγγράφου. Δηλαδή, το DTD έγγραφο λειτουργεί ως πρότυπο XML έγγραφο, με σκοπό να συγκριθεί με άλλα XML έγγραφα για να επιβεβαιώσει την εγκυρότητα των στοιχείων του εγγράφου. Να ελέγξει με άλλα λόγια, εάν το έγγραφο ακολουθεί τη σωστή δομή (ετικέτες) ενός XML εγγράφου. [4]

Ακολουθεί ένα **παράδειγμα** ενός **XML εγγράφου** που περιγράφει τα στοιχεία ενός πελάτη μιας παραγγελίας :

```
<? xml version="1.0" encoding="UTF-8"?>
<Order>
      <Customer>
      <name>George Samir</name>
      <street>1111 AnyStreet</street>
      <city>AnyTown</city>
      <state>GA</state>
      <zip>10000</zip>
      </Customer>
</Order>
```

[2] Μπορεί να χρησιμοποιεί και άλλα πρωτόκολλα όπως TCP/IP, SMTP.

Όπως παρατηρούμε, η πρώτη γραμμή ορίζει ότι η κωδικοποίηση των χαρακτήρων του XML εγγράφου, ακολουθεί τη UTF-8 κωδικοποίηση και βασίζεται στην προδιαγραφή XML 1.0. Κάθε στοιχείο περικλείεται από μια ετικέτα αρχής <> και μια ετικέτα τέλους </> για να περιγράψει το περιεχόμενο, δηλαδή τα δεδομένα που περιλαμβάνονται σε κάθε ένα από αυτά. Κάθε στοιχείο μπορεί να περιλαμβάνει περισσότερα από ένα στοιχεία-παιδιά. Στο παράδειγμα μας, διακρίνουμε τα στοιχεία **`Order`**, **`Customer`**, **`name`**, **`street`**, **`city`**, **`state`** και **`zip`**, όπου τα τέσσερα τελευταία είναι στοιχεία-παιδιά του στοιχείου **`Customer`**, το οποίο είναι παιδί του στοιχείου ρίζας **`Order`**.

2.1.1 Πλεονεκτήματα της γλώσσας XML

Η XML αποτελεί σήμερα το πρότυπο για την αποθήκευση δεδομένων που ανταλλάσσονται μεταξύ των εφαρμογών. Μερικοί από τους παράγοντες που επηρέασαν την ευρεία αποδοχή της είναι οι εξής :

- **Αποδοχή μεταφοράς και ανεξαρτησίας δεδομένων**

Η XML είναι ένας τυπικός τρόπος για να τεθούν οι πληροφορίες σε μορφή τέτοια που να μπορούν να επεξεργαστούν και να ανταλλάσσονται μεταξύ διαφορετικών, λειτουργικών συστημάτων, εφαρμογών λογισμικού και Διαδικτύου.

- **Ομοιομορφία και Συμμόρφωση**

Η XML ορίζει μια κοινή μορφοποίηση που τη καθιστά ευρέως αποδεκτή.

- **Απλότητα και Διαφάνεια**

Εάν είναι απαραίτητο, τα XML έγγραφα μπορούν να αναγνωστούν από ανθρώπους, παρόλο που δεν προορίζονται για αυτό το σκοπό.

- **Ευέλικτη και Επεκτάσιμη**

Εξαιτίας της δυνατότητας χρησιμοποίησης απεριόριστου πλήθους ετικετών σε ένα XML έγγραφο, όταν μια νέα πληροφορία απαιτείται, απλώς μια ετικέτα προστίθεται στη ήδη υπάρχουσα δομή. Δεν δημιουργεί καμία εξάρτηση για τη θέση που πρέπει να βρίσκονται οι πληροφορίες στη δομή.

➲ **Διαχωρισμός των δεδομένων**

Η αναπαράσταση των δεδομένων διαχωρίζεται από την παρουσίαση και τη μορφοποίηση των δεδομένων όταν εμφανίζονται σε ένα πρόγραμμα περιήγησης.

➲ **Βιομηχανική αποδοχή**

Εξαιτίας των πολλών δυνατοτήτων που προσφέρει, η XML έχει γίνει ευρέως αποδεκτή από την τεχνολογία των πληροφοριών και τη βιομηχανία υπολογιστών [4,5].

2.1.2 XML Schema

Το πρότυπο DTD αποδείχθηκε ότι δεν ήταν αρκετό για τον ορισμό της δομής των XML εγγράφων γιατί δεν μπορεί να υποστηρίξει πολλούς τύπους δεδομένων όπως, μορφές ημερομηνίας, αριθμούς ή άλλους κοινούς τύπους. Λόγω αυτών των περιορισμών, η Κοινοπραξία του Παγκοσμίου Ιστού (W3C[3]) όρισε ένα νέο πρότυπο για τον καθορισμό XML εγγράφων που ονομάζεται XML Schema.

Το XML Schema περιλαμβάνει τα ακόλουθα πλεονεκτήματα σε σχέση με το DTD:

- ✓ Επιτρέπει τον ορισμό απλών και σύνθετων τύπων στοιχείων (complex types) και γνωρισμάτων
- ✓ Παρέχει ένα τυποποιημένο τρόπο για να αναπαριστά ακέραιους, ημερομηνίες, ώρα, null τιμές και πολλές άλλες μορφές δεδομένων
- ✓ Ορίζεται ως έγγραφα XML και έτσι εξαλείφεται η ανάγκη ύπαρξης ειδικών parsers που να τα διαβάζουν [5].

Ένα **παράδειγμα** χρήσης ενός **XML Schema** για τον καθορισμό της δομής ενός XML εγγράφου στο οποίο θα τηρούμε πληροφορίες για ένα βιβλιοπωλείο (BookStore) που περιέχει πολλά βιβλία (Book) είναι το παρακάτω [6]:

[3] Διεθνής Οργανισμός Τυποποίησης του World Wide Web - http://www.w3.org/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    targetNamespace="http://www.books.org"
                    xmlns=http://www.books.org>
<!-- Ορισμός του σύνθετου στοιχείου BookStore -->
<xsd:element name="BookStore">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- Ορισμός του σύνθετου στοιχείου Book -->
<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- Ορισμός των τύπων για τα απλά στοιχεία του Book -->
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Η 1η γραμμή, ορίζει ότι η κωδικοποίηση των χαρακτήρων του XML εγγράφου, ακολουθεί τη UTF-8 κωδικοποίηση και βασίζεται στην προδιαγραφή XML 1.0. Η 2η γραμμή ορίζει το XMLSchema που καθορίζεται από τη διεύθυνση http://www.w3.org/2001/XMLSchema καθώς και το namespace στο οποίο θα ανήκουν οι νέοι καθορισμένοι τύποι των δεδομένων. Το namespace που χρησιμοποιείται εδώ είναι το http://www.books.org. Στη συνέχεια δηλώνονται τα σύνθετα στοιχεία που αποτελούν το XMLSchema.

Συγκεκριμένα, στη 6η γραμμή ορίζεται το πρώτο σύνθετο στοιχείο με όνομα **"BookStore"** και στη 14η γραμμή το δεύτερο με όνομα **"Book". Γ**ια να καθορίσουν τους σύνθετους τύπους των δεδομένων που θα περιέχουν ορίζεται η ετικέτα **<complexType>.** Επιπλέον, παρατηρούμε ότι και στα δύο σύνθετα στοιχεία ορίζεται η ετικέτα **<sequence>** για να καθορίσει μια ακολουθία στοιχείων-παιδιών που περιέχονται σε καθένα απο αυτά. Για κάθε στοιχείο-παιδί, καθορίζεται η ελάχιστη **(minOccurs)** και η μέγιστη **(minOccurs)** συχνότητα εμφάνισης τους.

Στις γραμμές 26-30 ορίζονται οι τύποι των απλών στοιχείων (Title, Author, Date, ISBN, Publisher) που περιέχονται στο στοιχείο Book.

Το **XML έγγραφο** που ακολουθεί το παραπάνω XML Schema περιγράφεται παρακάτω[6]:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <BookStore xmlns ="http://www.books.org"
3              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4              xsi:schemaLocation=http://www.books.org/BookStore.xsd">
5       <Book>
6            <Title>Web Services Security</Title>
7            <Author>Ravi Trivedi</Author>
8            <Date>Dec, 2002</Date>
9            <ISBN>1861007655</ISBN>
10           <Publisher>Wrox Publishing</Publisher>
11      </Book>
12  </BookStore>
```

Παρατηρούμε λοιπόν στην 1η γραμμή ότι, το XML έγγραφο ακολουθεί την ίδια κωδικοποίηση για την κωδικοποίηση των χαρακτήρων του όπως δηλώθηκε στο XMLSchema. Στην 2η γραμμή, ορίζεται η ιδιότητα xmlns για να δηλώσει το προεπιλεγμένο namespace του XML εγγράφου, δηλώντας με αυτό τον τρόπο στον ελεγκτή του schema ότι όλα τα στοιχεία που χρησιμοποιούνται σε αυτό το XML έγγραφο προέρχονται από το namespace http://www.books.org.

Η 3η γραμμή δηλώνει το namespace του XMLShcema που καθορίζεται από τη διεύθυνση http://www.w3.org/2001/XMLSchema-instance.

Η 4η γραμμή ορίζει την ιδιότητα chemaLocation για να δηλώσει στον ελεγκτή του XML Schema ότι το namespace http://www.books.org είναι ορισμένο στο BookStore.xsd. Οπότε, ο ελεγκτής του XML Schema θα αναζητήσει το συγκεκριμένο αρχείο για να επικυρώσει την ορθότητα του XML εγγράφου.

Στη 5η γραμμή, δηλώνεται το σύνθετο στοιχείο **<Book>** που περιέχει τις λεπτομέρειες ενός βιβλίου, που περιγράφονται από τα στοιχεία-παιδιά που ορίστηκαν στο σύνθετο στοιχείο Book του XML Schema. Όπως παρατηρούμε, η σειρά εμφάνισης των στοιχείων-παιδιών είναι ακριβώς η ίδια, όπως δηλώθηκε στο XML Schema.

2.2 SOAP (Simple Object Access Protocol)

Το SOAP είναι μια προδιαγραφή XML πρωτοκόλλου για την ανταλλαγή μηνυμάτων μεταξύ του αιτούντος και του παρόχου μιας υπηρεσίας Ιστού.

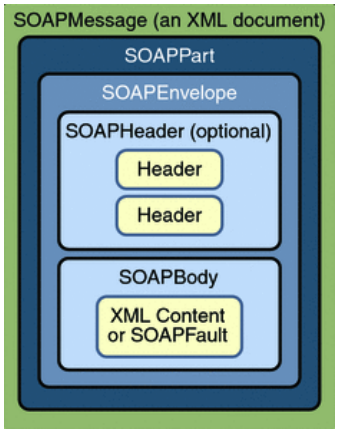
Το πρωτόκολλο SOAP έχει τα ακόλουθα χαρακτηριστικά:

- Έχει σχεδιαστεί ώστε να είναι απλό και επεκτάσιμο.
- Παρέχει ένα πλαίσιο για να περιγράψει το περιεχόμενο των μηνυμάτων και τις οδηγίες για τη διαδικασία ανταλλαγής αυτών, καθώς και ένα προαιρετικό σύνολο κανόνων κωδικοποίησης για την αναπαράσταση καθορισμένων τύπων δεδομένων. Είναι στην ουσία ένας φάκελος (envelope) στον οποίο περικλείεται οποιαδήποτε πληροφορία πρέπει να σταλεί.
- Όλα τα SOAP μηνύματα έχουν τυποποιημένη δομή που βασίζεται στην XML.

- Το SOAP είναι ένα ανεξάρτητο πρωτόκολλο μεταφοράς. Βασίζεται στο πρωτόκολλο HTTP για την διαπραγμάτευση και τη μεταφορά των μηνυμάτων. Ως εκ τούτου, το SOAP μπορεί να τρέξει πάνω στην υπάρχουσα υποδομή του Διαδικτύου.
- Υποστηρίζει χαλαρά συνδεδεμένες εφαρμογές που αλληλεπιδρούν ανταλλάσοντας ασύγχρονα μηνύματα.

Λόγω αυτών των χαρακτηριστικών, δεν έχει σημασία ποια τεχνολογία χρησιμοποιείται για την υλοποίηση της υπηρεσίας ούτε από τη μεριά του αιτούντα της υπηρεσίας (Client), ούτε και από τη μεριά του παρόχου των υπηρεσιών (Service Provider) εφ' όσον μπορούν να εκδίδουν και να επεξεργάζονται αντίστοιχα μηνύματα XML.

Όπως δείχνει το Σχήμα 2.1, ένα απλό μήνυμα SOAP αποτελείται από ένα φάκελο (envelope) που ορίζει ένα πλαίσιο για την περιγραφή του μηνύματος, και περιέχει τρία μέρη: τη **κεφαλίδα** (header), το **σώμα** (body) και το **σφάλμα** (fault). Η κεφαλίδα είναι προαιρετική, και περιέχει πληροφορίες χρήσιμες για τους ενδιάμεσους και τελικούς κόμβους παρέχοντας σε αυτούς τη δυνατότητα επεξεργασίας των μηνυμάτων τους χρησιμοποιώντας ένα σύνολο κανόνων κωδικοποίησης για να εκφραστούν καθορισμένοι τύποι δεδομένων. Το σώμα, είναι υποχρεωτικό και καθορίζει μια σύμβαση για την σωστή αναπαράσταση των κλήσεων απομακρυσμένης διαδικασίας (RPCs[4]) και απαντήσεων σε αυτές. Τέλος, το σφάλμα, είναι ένα προαιρετικό στοιχείο που παρέχει πληροφορίες σχετικά με σφάλματα που παρουσιάστηκαν κατά την επεξεργασία των μηνυμάτων [5].



Σχήμα 2.1 Η δομή ενός SOAP μηνύματος.

[4] Remote Procedure Calls: Είναι μια άλλη τεχνική πρόσβασης σε υπηρεσίες ιστού που χρησιμοποιείται σε client/server εφαρμογές. Ο client επικαλείται μια εξ' αποστάσεως διαδικασία που βρίσκεται σε κάποιον server και έπειτα λαμβάνει την αντίστοιχη απάντηση από αυτόν. Επιπλέον, η τεχνική αυτή αποκρύπτει τις λεπτομέρειες της επικοινωνίας του δικτύου.

Η βασική σύνταξη ενός SOAP μηνύματος περιλαμβάνει τα εξής στοιχεία [7]:

```
<?xml version="1.0 encoding="utf-8""?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header> ... </soap:Header>
       <soap:Body>
        ...
              <soap:Fault>
                ...
              </soap:Fault>
       </soap:Body>
</soap:Envelope>
```

Για να γίνει περισσότερο αντιληπτό, ακολουθεί ένα παράδειγμα ενός SOAP μηνύματος [8]:

SOAP Μήνυμα αίτησης

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetEndorsingBoarder xmlns:m="http://namespaces.snowboard-info.com">
      <manufacturer>K2</manufacturer>
      <model>Fatbob</model>
    </m:GetEndorsingBoarder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Όπως παρατηρούμε, στην κορυφή του μηνύματος υπάρχει η ετικέτα `<?xml version="1.0" encoding="UTF-8"?>` για να καθορίσει την έκδοση της XML γλώσσας που χρησιμοποιείται καθώς και το χαρακτήρα κωδικοποίησης του μηνύματος. Ακολουθεί η ετικέτα `<SOAP-ENV:Envelope>`, για να καθορίσει ότι η κωδικοποίηση του SOAP μηνύματος ακολουθεί το schema που ορίζεται στο `http://schemas.xmlsoap.org/soap/encoding`. Έπειτα, ακολουθεί η ετικέτα `<SOAP-ENV:Body>`, η οποία περιλαμβάνει την κύρια πληροφορία του μηνύματος. Παρατηρούμε λοιπόν ότι, το μήνυμα αποστέλλεται σε μια υπηρεσία Ιστού, την `namespaces.snowboard-info.com`, η οποία για ένα συγκεκριμένο κωδικό κατασκευαστή (`manufacturer`) και ένα συγκεκριμένο μοντέλο (`model`), ζητάει να τις επιστραφεί το τρέχων όνομα του κατασκευαστή. Οι παράμετροι που απαιτούνται για την επίτευξη αυτής της λειτουργίας δίδονται στο κυρίως σώμα του μηνύματος.

SOAP Μήνυμα απάντησης

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetEndorsingBoarderResponse
       xmlns:m="http://namespaces.snowboard-info.com">
      <endorsingBoarder>Chris Englesmann</endorsingBoarder>
    </m:GetEndorsingBoarderResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ο παραλήπτης του παραπάνω μηνύματος απαντάει με ένα άλλο SOAP μήνυμα, περιλαμβάνοντας τις αντίστοιχες ετικέτες που περιγράφηκαν προηγουμένως. Η αλλαγή που παρατηρείται είναι στο σώμα του μηνύματος, το οποίο τώρα περιέχει την πληροφορία που ζητήθηκε, δηλαδή το όνομα του κατασκευαστή (`endorsingBoarder`).

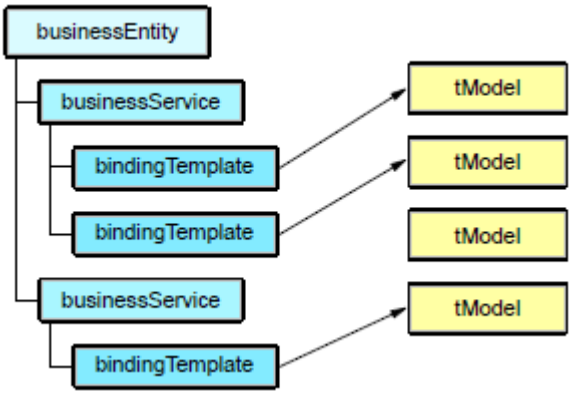
2.3 UDDI (Universal Description, Discovery and Integration)

Το UDDI παρέχει ένα τυποποιημένο πρότυπο για τη δημοσίευση, την ανακάλυψη και την ενοποίηση των υπηρεσιών Διαδικτύου. Είναι μια πρωτοβουλία της βιομηχανίας και για αυτό συνήθως αναφέρεται και ως μητρώο επιχειρήσεων UDDI (UDDI Business Registry). Ορίζει τον τρόπο καταχώρησης των υπηρεσιών διαδικτύου σε ένα ενιαίο μητρώο προσπαθώντας έτσι, να δημιουργήσει μια ανεξάρτητη πλατφόρμα για την περιγραφή υπηρεσιών επιτρέποντας στις επιχειρήσεις να ανακαλύπτουν γρήγορα και εύκολα τις μεταξύ τους υπηρεσίες.

Το UDDI ορίζει δυο κατηγορίες προγραμματιστικών διεπαφών APIs (Application Programming Interface) για την προσπέλαση των υπηρεσιών. Τη διεπαφή δημοσίευσης (Publishing API), που χρησιμοποιούν οι επιχειρήσεις για την καταχώρηση υπηρεσιών και δημοσίευσης των περιγραφών των υπηρεσιών τους και τη διεπαφή ερωτήσεων (Inquiry API) που χρησιμοποιούν οι πελάτες, για την εύρεση υπηρεσιών ενός συγκεκριμένου είδους επιχείρησης. Για την περιγραφή των υπηρεσιών χρησιμοποιούνται WSDL έγγραφα τα οποία δημοσιεύονται από τους διαχειριστές του μητρώου UDDI χρησιμοποιώντας XML μορφοποίηση. Οι προγραμματιστικές

διεπαφές, παρέχουν ένα σύστημα αλληλεπίδρασης μεταξύ του διαχειριστή του μητρώου και του χρήστη για την καταχώρηση, διαχείριση, αναζήτηση εταιριών και υπηρεσιών μέσα στον μητρώο [9].

Υπάρχουν τέσσερις θεμελιώδεις τύποι δεδομένων σε μια καταχώρηση UDDI, όπως αναπαριστάται στο Σχήμα 2.2 [5].

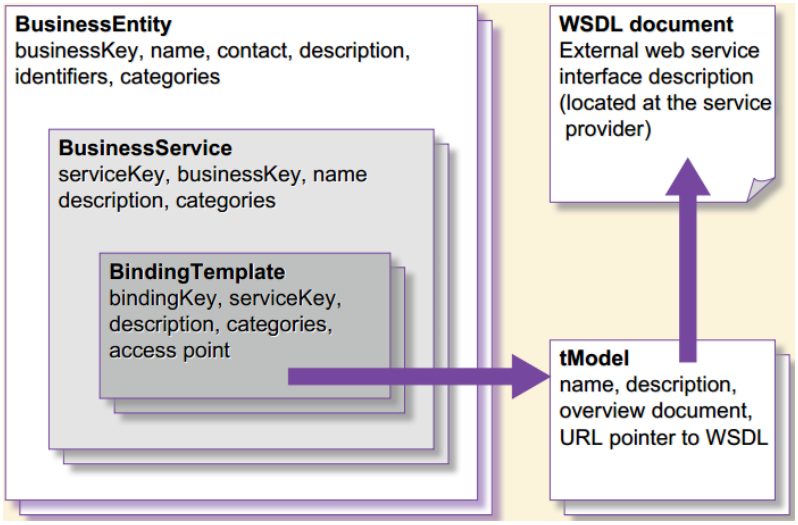


Σχήμα 2.2 Βασικοί UDDI τύποι δεδομένων.

Η UDDI καταχώρηση ξεκινάει με το **businessEntity**. Το στοιχείο businessEntity περιγράφει την επιχείρηση που παρέχει την υπηρεσία, περιλαμβάνοντας τις βασικές πληροφορίες της επιχείρησης (π.χ όνομα επιχείρησης και στοιχεία επικοινωνίας), τις πληροφορίες κατηγοριοποίησης (π.χ. είδος επιχείρησης), καθώς και πληροφορίες αναγνώρισης (π.χ. αριθμός επιχείρησης). Το businessEntity περιλαμβάνει μια συλλογή από **businessService** στοιχεία, ένα για κάθε υπηρεσία διαδικτύου που η επιχείρηση επιθυμεί να δημοσιεύσει. Κάθε στοιχείο businessService περιλαμβάνει τις περιγραφικές πληροφορίες σχετικά με την υπηρεσία διαδικτύου που προσφέρεται από το στοιχείο businessEntity. Το businessService περιλαμβάνει μια συλλογή από bindingTemplates στοιχεία. Τα **bindingTemplates** στοιχεία περιγράφουν τις απαραίτητες τεχνικές πληροφορίες που απαιτούνται για την πρόσβαση σε μια υπηρεσία (π.χ. URL). Τα bindingTemplates στοιχεία μπορούν να συσχετιστούν με ένα αριθμό στοιχείων tModel και να διαμορφώσουν το τεχνικό μοντέλο που προκύπτει.

Ένα **tModel** είναι ένα γενικό μοντέλο που χρησιμοποιείται για την αποθήκευση τεχνικών πληροφοριών σχετικές με τον τρόπο χρήσης μιας υπηρεσίας, τα πρωτόκολλα, του όρους που πρέπει να ακολουθούνται κ.α. [9, 10].

Ένα αντιπροσωπευτικό σχήμα αναπαράστασης των παραπάνω πληροφορίων παρουσιάζεται παρακάτω (Σχήμα 2.3).



Σχήμα 2.3 Το UDDI μοντέλο δεδομένων.

Υπάρχουν τρεις τύποι υπηρεσιών που προσφέρει μια υπηρεσία καταγραφής registry UDDI. Αυτές είναι :

- **White pages** (Λευκές σελίδες)

Παρέχουν τις βασικές πληροφορίες αναγνώρισης των επιχειρήσεων, όπως όνομα, τηλ, διεύθυνση, καθώς και μοναδικά αναγνωριστικά, που επιτρέπουν σε χρήστες να ανακαλύψουν την διαδικτυακή υπηρεσία που προσφέρει η επιχείρηση.

- **Yellow pages** (Κίτρινες σελίδες)

Παρέχουν πληροφορίες σχετικές την κατηγοριοποίηση των υπηρεσιών ή επιχειρήσεων σύμφωνα με επίσημα πρότυπα ταξινόμησης. Επιτρέπουν δηλαδή, σε χρήστες να ανακαλύψουν την διαδικτυακή υπηρεσία που προσφέρει η επιχείρηση σύμφωνα με τη κατηγοριοποίηση της στο μητρώο (π.χ. είδος επιχείρησης, κωδ. προϊόντος).

- **Green pages** (Πράσινες σελίδες)

Περιέχουν τις τεχνικές πληροφορίες που περιγράφουν τον τρόπο πρόσβασης για κάθε μια από τις προσφερόμενες υπηρεσίες. Τέτοιες πληροφορίες μπορεί να είναι η διεύθυνση μιας υπηρεσίας και γενικά οποιαδήποτε πληροφορία απαραίτητη για τον εντοπισμό της [9, 10].

2.4 WSDL (Web Service Description Language)

Η WSDL - Γλώσσα Περιγραφής Υπηρεσιών Ιστού, είναι μια γλώσσα που βασίζεται στην XML και παρέχει ένα πρότυπο για την περιγραφή των υπηρεσιών διαδικτύου προκειμένου τα προγράμματα/πελάτες να έχουν πρόσβαση σε αυτές. Αναπτύχθηκε αρχικά από τη Microsoft και την IBM και βρίσκεται υπό διαδικασία τυποποίησης από το W3C από τις αρχές του 2002. Οι κύριες εκδόσεις της είναι η WSDL 1.1 και η WSDL 2.0 η οποία προτείνεται από το W3C. Παρόλο που η WSDL, δεν έχει ακόμα την τελική έγκριση κάποιου Οργανισμού, έχει ήδη ευρέως υιοθετηθεί ως η «de facto» περιγραφική γλώσσα υπηρεσιών διαδικτύου με πιο διαδεδομένη τη WSDL 1.1 γατί υποστηρίζεται από πληθώρα εφαρμογών υλοποίησης υπηρεσιών Ιστού.

Χρησιμοποιώντας WSDL, ένας πάροχος υπηρεσιών μπορεί να περιγράψει τη λειτουργικότητα, την ποιότητα των υπηρεσιών, και άλλα χαρακτηριστικά μιας υπηρεσίας Ιστού, έτσι ώστε ο αιτών μιας υπηρεσίας να μπορεί να καταλάβει πώς να αλληλεπιδράσει σωστά με την υπηρεσία. Στον κόσμο των χαλαρά συνδεδεμένων υπηρεσιών Ιστού, η WSDL διαδραματίζει κεντρικό ρόλο για τη διαλειτουργικότητα μεταξύ των υπηρεσιών που εφαρμόζονται σε διάφορες πλατφόρμες [11].

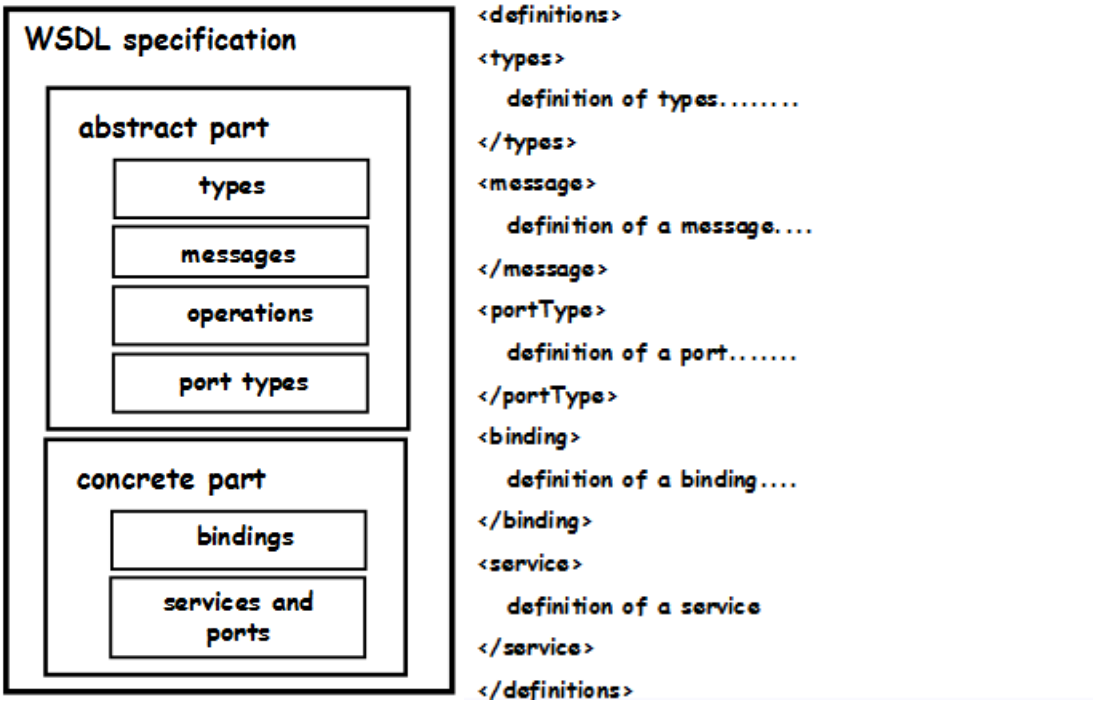
Η WSDL επιτρέπει τον διαχωρισμό της περιγραφής της αφηρημένης (abstract) λειτουργικότητας που προσφέρει μια υπηρεσία από συγκεκριμένες λεπτομέρειες (concrete) της περιγραφής υπηρεσιών, όπως το «πώς» και το «πού» αυτή η λειτουργία προσφέρεται. Αυτή η προδιαγραφή ορίζει μια γλώσσα για την περιγραφή της αφηρημένης λειτουργικότητας μιας υπηρεσίας αναφέροντας τα μηνύματα που ανταλλάσσονται, καθώς και ένα πλαίσιο για την περιγραφή των συγκεκριμένων στοιχείων της περιγραφόμενης υπηρεσίας. Ορίζει επίσης τα κριτήρια συμμόρφωσης για τα έγγραφα WSDL σε αυτή τη γλώσσα [12].

Συγκεκριμένα, η περιγραφή μιας υπηρεσίας ιστού αποτελείται όπως δείχνει και το Σχήμα 2.4 από δύο μέρη:

- Το **αφηρημένο μέρος (abstract part)**: στο οποίο περιγράφονται οι διαθέσιμες λειτουργίες μια υπηρεσίας διαδικτύου καθώς και η βασική σύνταξη που πρέπει να έχουν τα μηνύματα ανταλλαγής για την επιτυχή υλοποίηση της κλήσης μιας υπηρεσίας.
- Το **συγκεκριμένο μέρος (concrete part)**: στο οποίο περιγράφονται πληροφορίες που αφορούν στην τοποθεσία μιας υπηρεσίας αναφέροντας την ακριβή διεύθυνση της (URI), καθώς και στην υλοποίηση της. Δηλαδή, στην περίπτωση που τα μηνύματα SOAP

ανταλλάσσονται μέσω HTTP, η σύνδεση καθορίζει ποια μέρη του μηνύματος εισόδου πάνε στην επικεφαλίδα SOAP και ποια μέρη πηγαίνουν στο σώμα SOAP. [11].

Το Σχήμα 2.4 δείχνει τη δομή ενός πρότυπου εγγράφου WSDL 1.1 που ακολουθείται για την περιγραφή των υπηρεσιών Ιστού [13].



Σχήμα 2.4 Συντακτική δομή ενός WSDL εγγράφου για την περιγραφή μιας υπηρεσίας ιστού.

Στα WSDL έγγραφα, οι υπηρεσίες ορίζονται σύμφωνα με 6 βασικά στοιχεία:

- **definitions,** το οποίο ορίζει το όνομα μιας υπηρεσίας διαδικτύου και καθορίζει όλα τα namespaces[5] του WSDL εγγράφου.
- **types**, το οποίο παρέχει τους τύπους όλων των δεδομένων που χρησιμοποιούνται από την υπηρεσία διαδικτύου για τα μηνύματα που ανταλλάσσονται μεταξύ πελάτη/εξυπηρετητή . Το εξ' ορισμού πρότυπο που χρησιμοποιείται στην WSDL είναι το XML Schema.
- **message**, στο οποίο καθορίζεται η μορφή των μηνυμάτων που ανταλλάσσονται καθώς και το σύνολο των παραμέτρων που σχετίζονται με τις μεθόδους της υπηρεσίας.

[5] Χρησιμοποιείται για να δηλώσει όλες τις εξωτρικές προδιαγραφές/πρότυπα που χρησιμοποιούνται σε ένα WSDL έγγραφο. Π.χ. WSDL, XMLSchema, SOAP.

- **portType**, το οποίο είναι ένα σύνολο αφηρημένων λειτουργιών που παρέχει η υπηρεσία. Κάθε λειτουργία μπορεί να αναφέρεται τόσο σε μηνύματα αίτησης όσο και σε μηνύματα απόκρισης.
- **binding**, το οποίο καθορίζει το πρωτόκολλο και τις προδιαγραφές μορφοποίησης των δεδομένων που πρέπει να έχουν οι λειτουργίες και τα μηνύματα, σε ένα συγκεκριμένο portType. Με άλλα λόγια, περιγράφει πως μπορεί να υλοποιηθεί μια υπηρεσία.
- **service**, το οποίο χρησιμοποιείται για να ορίσει τη διεύθυνση (URI) πρόσβασης στην υπηρεσία. Ένα service μπορεί να είναι μια συλλογή απο ports [12, 13].

Προκειμένου να κατανοήσουμε περισσότερο τα στοιχεία που αποτελούν ένα WSDL έγγραφο, θα χρησιμοποιήσουμε το παράδειγμα [14] που απεικονίζεται στο Παράρτημα αυτής της διατριβής (βλέπε σελ. 106).

Όπως παρατηρούμε, στη 1η γραμμή του αρχείου υπάρχει η ετικέτα `<?xml version="1.0" encoding="UTF-8"?>` για να καθορίσει την έκδοση της XML γλώσσας που χρησιμοποιείται καθώς και το χαρακτήρα κωδικοποίησης του μηνύματος.

Η 2η γραμμή περιλαμβάνει το στοιχείο ρίζα `<definitions>`, για να καθορίσει το όνομα της υπηρεσίας, εδώ "`FooSample`". Επιπλέον, το στοιχείο `<definitions>` περιέχει ένα σύνολο από `Namespaces`, τόσο στο στοιχείο ρίζα όσο και στο `<schema>` που αποτελεί μέρος του στοιχείου `<types>`. Τα `Namespaces` που χρησιμοποιούνται εδώ, είναι τα WSDL, SOAP και XMLSchema.

Το στοιχείο `<types>` ορίζεται στην 10η γραμμή του αρχείου και δηλώνει ότι ο τύπος δεδομένων που χρησιμοποιείται σε αυτό το έγγραφο είναι το `XMLSchema` .

Οι γραμμές 18-23 περιγράφουν τα μηνύματα αίτησης και απόκρισης που ανταλλάσσονται. Δηλαδή, στη 18η γραμμή περιλαμβάνεται το πρώτο στοιχείο `<message>` για να περιγράψει το μήνυμα αίτησης "`Simple.foo`" ενώ στη 21η γραμμή υπάρχει το δεύτερο στοιχείο `<message>` για να περιγράψει το μήνυμα απόκρισης "`Simple.fooResponse`" το οποίο θα περιέχει έναν ακέραιο.

Στη 24η γραμμή περιλαμβάνεται το στοιχείο `<portType>` για να περιγράψει το σύνολο των λειτουργιών που μπορεί να δεχτεί η συγκεκριμένη υπηρεσία. Εδώ, το στοιχείο `<portType>` της υπηρεσίας "`FooSample`", ορίζει τη λειτουργία με όνομα "`Foo`", που αναπαριστάται από το

στοιχείο <operation>. Το στοιχείο <operation> αποτελείται από το μήνυμα εισόδου "Simple.foo" και το μήνυμα εξόδου "Simple.fooResponse" που ορίζονται από τα στοιχεία <input message> και <output message> αντίστοιχα.

Στη 30η γραμμή, περιλαμβάνεται το στοιχείο <binding>, με όνομα "SimpleBinding" για να ορίσει το πρωτόκολλο που θα χρησιμοποιηθεί. Εδώ, το στοιχείο <soap:binding>, ορίζει ότι θα χρησιμοποιεί το πρωτόκολλο SOAP και η μετάδοση των SOAP μηνυμάτων θα γίνει πάνω από το HTTP. Επιπλέον, παρατηρούμε ότι το <binding> στοιχείο περιλαμβάνει τη λειτουργία "foo" καθώς και δεδομένα εισόδου και εξόδου παρόμοια με αυτά του <portType>.

Τέλος, στη 49η γραμμή του αρχείου, περιλαμβάνεται το στοιχείο <service> με όνομα "FOOSAMPLEService" για να ορίσει την URI διεύθυνση και την θύρα (port) για πρόσβαση στην υπηρεσία "FooSample". Η URI διεύθυνση ορίζεται από το στοιχείο <soap:address> εδώ, "http://carlos:8080/FooSample/FooSample.asp" ενώ η θύρα ορίζεται από το στοιχείο <port> με όνομα "SimplePort".

Κεφάλαιο 3
Υπηρεσίες Διαδικτύου (Web Services)

3.1 Βασικές Έννοιες

Τα τελευταία χρόνια, ο Παγκόσμιος Ιστός (World Wide Web) έχει επιφέρει σημαντικές αλλαγές στον τρόπο με τον οποίο οι χρήστες του Διαδικτύου επικοινωνούν μεταξύ τους. Έμφαση έχει δοθεί στη προσπάθεια ανάπτυξης υπηρεσιών οι οποίες θα δώσουν επιπρόσθετες δυνατότητες τόσο στους χρήστες αλλά όσο και στις επιχειρήσεις που τις χρησιμοποιούν ως μέσο προβολής και δημοσιοποίησης των δραστηριοτήτων τους.

Μια νέα τεχνολογία που έχει ήδη θεσπιστεί από το W3C με το όνομα Υπηρεσία Διαδικτύου (Web Service) ήρθε για να δώσει τη λύση στο πρόβλημα της συμβατότητας και διαλειτουργικότητας των υπηρεσιών Ιστού που υπέφεραν οι παλαιότερες τεχνολογίες DCOM[6](Distributed

[6] Είναι η κατανεμημένη επέκταση του COM (Component Object Model), το οποίο παρέχει ένα σύνολο από διεπαφές που επιτρέπουν στους πελάτες (client) και τους διακομιστές (servers) να επικοινωνούν εντός ίδιου δικτύου για την ανταλλαγή των υπηρεσιών τους.

Component Object Model) και CORBA[7](Common Object Request Broker Architecture) εξαιτίας των διαφορών στις γλώσσες προγραμματισμού και του υλικολογισμικού (middleware) που χρησιμοποιούνταν.

Η άνθηση των υπηρεσιών διαδικτύου δεν είναι μια επανάσταση στα κατανεμημένα συστήματα. Αντίθετα, είναι μια φυσική εξέλιξη στην αρχιτεκτονική των υπηρεσιών διαδικτύου καθιστώντας δυνατή τη δημιουργία και παροχή ηλεκτρονικών υπηρεσιών με αρκετά απλό και δυναμικό τρόπο, χρησιμοποιώντας ανοιχτά πρότυπα και προδιαγραφές,

Είναι εντελώς ανεξάρτητες από τη γλώσσα προγραμματισμού, το λειτουργικό σύστημα, και το υλικό για την προώθηση χαλαρής σύνδεσης μεταξύ του αιτούντα της υπηρεσίας και του παρόχου υπηρεσιών [5]. Λέγοντας «χαλαρή σύνδεση» εννοούμε το βαθμό ανεξαρτησίας που πρέπει να υπάρχει μεταξύ δύο συστημάτων. Ως εκ τούτου, οι υπηρεσίες διαδικτύου συνδέονται και αλληλεπιδρούν περισσότερο ελεύθερα στο Διαδίκτυο, αγνοώντας την συμπεριφορά και τον τρόπο υλοποίησης των εφαρμογών που τις χρησιμοποιούν.

Ένας κοινά αποδεκτός ορισμός για τα web services δεν έχει ακόμη επικρατήσει, ωστόσο αν θέλουμε να προσδιορίσουμε πιο σωστά την έννοια τους, θα μπορούσαμε να χρησιμοποιήσουμε τον ορισμό που έρχεται από την IBM[8] :

"Τα web services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service αποτελεί μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μια λειτουργία προς εκτέλεση καθώς και τα δεδομένα που ανταλλάσσονται με κάποια άλλη εφαρμογή. Μια web service μπορεί να αποτελείται από μια ομάδα από web services, που όταν αλληλεπιδρούν μεταξύ τους καθορίζουν μια συγκεκριμένη web service εφαρμογή σε μια service-oriented αρχιτεκτονική." [15]

Οι υπηρεσίες διαδικτύου συνδυάζουν τη δύναμη δύο τεχνολογιών: την XML, την παγκόσμια γλώσσα περιγραφής δεδομένων, και το HTTP πρωτόκολλο μεταφοράς που ευρέως υποστηρίζεται από τα προγράμματα περιήγησης (browsers) και τους Web Servers.

Web Services = XML + πρωτόκολλο μεταφοράς (όπως το HTTP).

[7] Είναι μια αρχιτεκτονική για τη δημιουργία, τη διανομή και τη διαχείριση κατανεμημένων αντικειμένων μέσα σε ένα δίκτυο, επιτρέποντας σε προγράμματα να επικοινωνούν μέσω μιας διεπαφής «broker». Προτάθηκε από τη κοινοπραξία OMG (Object Management Group), αλλά αναπτύχθηκε από τη Sun.

[8] http://www.ibm.com

3.2 Βασικά χαρακτηριστικά

Μερικά από τα βασικά χαρακτηριστικά των υπηρεσιών διαδικτύου είναι τα εξής:

- **Οι υπηρεσίες διαδικτύου μπορούν να ενεργοποιηθούν, να δημοσιευθούν και να εντοπιστούν σε όλο το Διαδίκτυο**

Τα **πρότυπα** που απαιτούνται για να το πράξουν αυτό είναι τα εξής:

– Simple Object Access Protocol (**SOAP**), γνωστό ως πρωτόκολλο Απλής Προσπέλασης Αντικειμένου, βασισμένο σε XML μηνύματα πρωτοκόλλου

– Web Service Description Language (**WSDL**), η γλώσσα περιγραφής των υπηρεσιών του Παγκόσμιου Ιστού

– Universal Description, Discovery, and Integration (**UDDI**), ένας κατάλογος μητρώου που μπορεί να χρησιμοποιηθεί για την αναζήτηση των περιγραφών των υπηρεσιών

- **Οι υπηρεσίες διαδικτύου είναι αυτοπεριγραφικές**

Όταν δημοσιεύεται μια υπηρεσία Ιστού, δημοσιεύεται και μια διεπαφή (interface) αυτής ή αλλιώς μια περιγραφή αυτής. Η περιγραφή περιλαμβάνει όλες τις απαραίτητες πληροφορίες που απαιτούνται προκειμένου να αναγνωριστεί και κατά συνέπεια να κληθεί από τον ενδιαφερόμενο που ζητάει την υπηρεσία.

- **Οι υπηρεσίες διαδικτύου είναι ανεξάρτητες και διαλειτουργικές**

Μια υπηρεσία διαδικτύου είναι εντελώς ανεξάρτητη από την πλατφόρμα στην οποία υλοποιείται και τη γλώσσα προγραμματισμού στην οποία η υπηρεσία είναι γραμμένη.

- **Οι υπηρεσίες διαδικτύου είναι ανοιχτά και καθορισμένα πρότυπα**

Η γλώσσα XML και το ευρέως διαδεδομένο πρωτόκολλο HTTP αποτελούν τη τεχνική βάση για τις διαδικτυακές υπηρεσίες. Ένα μεγάλο μέρος της τεχνολογίας των υπηρεσιών διαδικτύου έχει κατασκευαστεί με ανοικτού κώδικα εργαλεία. Ως εκ τούτου, η ανεξαρτησία και η διαλειτουργικότητα είναι ρεαλιστικοί στόχοι.

- **Οι υπηρεσίες διαδικτύου είναι δυναμικές**

Εξαιτίας της δημοσιοποίησης της περιγραφής των υπηρεσιών Ιστού χρησιμοποιώντας UDDI και WSDL πρότυπα, η διαδικασία της εύρεσης τους γίνεται με αυτοματοποιημένο και δυναμικό τρόπο.

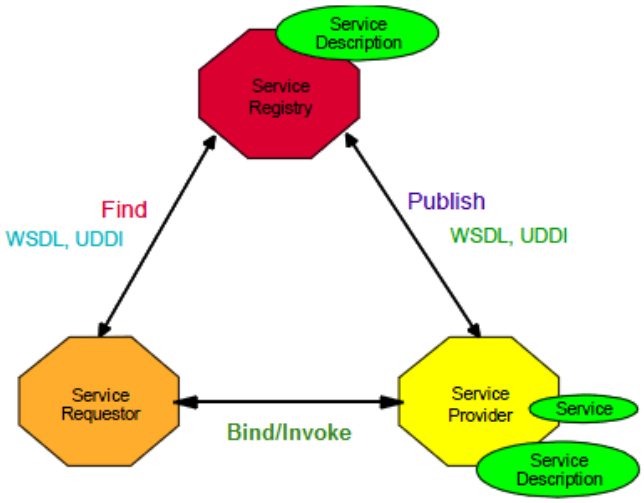
- **Οι υπηρεσίες διαδικτύου είναι επαναχρησιμοποιήσιμες**

Οι υπηρεσίες διαδικτύου επιτρέπουν στους προγραμματιστές να επαναχρησιμοποιήσουν την ήδη υπάρχουσα τεχνολογία μιας εφαρμογής και να τη συνενώσουν με άλλες εφαρμογές [5].

3.3 Το μοντέλο των Web Services

Κάθε υπηρεσία διαδικτύου για να μπορεί να λειτουργήσει αποδοτικά, αλλά ταυτόχρονα και να εξελιχθεί πρέπει να βασίζεται σε κάποια συγκεκριμένη δομή. Να ακολουθεί δηλαδή, ένα συγκεκριμένο μοντέλο πάνω στο οποίο θα αναπτύσσονται όλες οι λειτουργίες και τα πρωτόκολλα της αντίστοιχης υπηρεσίας. Το μοντέλο που χρησιμοποιείται στις μέρες μας για τις υπηρεσίες διαδικτύου είναι το λεγόμενο SOA (Service Oriented Architecture). Το SOA αντιπροσωπεύει μια αρχιτεκτονική προσανατολισμένη σε υπηρεσίες. Σε αυτό το μοντέλο συναντάμε ξεχωριστές οντότητες, οι οποίες μπορούν να διανεμηθούν πάνω από ένα δίκτυο, να συνδυαστούν και να ξαναχρησιμοποιηθούν έτσι ώστε να δημιουργήσουν επιχειρησιακές εφαρμογές.

Το μοντέλο αυτό, που χρησιμοποιείται για την περιγραφή των υπηρεσιών διαδικτύου απεικονίζεται στο παρακάτω Σχήμα 3.1 [16].



Σχήμα 3.1 Το μοντέλο των Υπηρεσιών Διαδικτύου.

Σύμφωνα με το παραπάνω σχήμα, διακρίνουμε τις οντότητες που αποτελούν την αρχιτεκτονική των υπηρεσιών διαδικτύου. Αυτές είναι: η οντότητα που ζητάει την υπηρεσία (Service Requestor), η οντότητα που παρέχει την υπηρεσία (Service Provider) και τέλος η οντότητα του μητρώου υπηρεσιών (Service Registry).

Ακολουθεί ανάλυση για κάθε μια από τις οντότητες που χρησιμοποιούνται στη συγκεκριμένη αρχιτεκτονική με έμφαση στις λειτουργίες που εκτελεί η καθεμία είτε ανεξάρτητα είτε σε συνδυασμό με κάποια άλλη.

• **Service provider** - **η οντότητα η οποία παρέχει την υπηρεσία**. Από αρχιτεκτονικής πλευράς είναι η πλατφόρμα που αναλαμβάνει την υλοποίηση της υπηρεσίας. Δηλαδή, δημοσιοποιεί την περιγραφή της υπηρεσίας ιστού στην οντότητα του καταλόγου υπηρεσιών (Service Registry) και ουσιαστικά παρέχει την υπηρεσία στο διαδίκτυο. Επιπλέον, της ανατίθεται ο ρόλος της λήψης των μηνυμάτων κλήσης για την υπηρεσία από έναν ή περισσότερους αιτούμενους και της παροχής των απαραίτητων αποτελεσμάτων.

• **Service requestor - η οντότητα η οποία ζητάει την υπηρεσία.** Από αρχιτεκτονικής πλευράς είναι η εφαρμογή που αναζητά και προκαλεί την εκτέλεση ενός web service. Με άλλα λόγια, είναι ο «αιτούμενος» (Client) της υπηρεσίας, ο οποίος πυροδοτεί την έναρξη της όλης διαδικασίας. Δηλαδή, ο αιτών αναζητά την κατάλληλη περιγραφή μιας υπηρεσίας στις καταχωρήσεις του μητρώου υπηρεσιών. Στη συνέχεια, και ενώ έχει εντοπίσει τον επιθυμητό πάροχο, δημιουργεί σύνδεση με αυτόν, καλεί την υπηρεσία και λαμβάνει τα αποτελέσματα.

• **Service registry - η οντότητα του καταλόγου υπηρεσιών.** Περιέχει καταχωρήσεις με δυνατότητες αναζήτησης, όπου οι πάροχοι υπηρεσιών δημοσιεύουν τις περιγραφές των υπηρεσιών τους και οι αιτούντες εντοπίζουν τις υπηρεσίες και τις σχετικές πληροφορίες δέσμευσης (binding) με αυτές [16].

Για να επωφεληθεί μια εφαρμογή από την χρησιμοποίηση των υπηρεσιών διαδικτύου, τρεις λειτουργίες πρέπει να πραγματοποιούνται: δημοσίευση της περιγραφής των υπηρεσιών, αναζήτηση ή εύρεση της περιγραφής των υπηρεσιών, και δέσμευση ή ενεργοποίηση των υπηρεσιών που βασίζονται στην περιγραφή των υπηρεσιών. Αυτές οι λειτουργίες μπορούν να συμβούν είτε μεμονωμένα είτε επαναληπτικά.

Πιο αναλυτικά, οι λειτουργίες αυτές είναι:

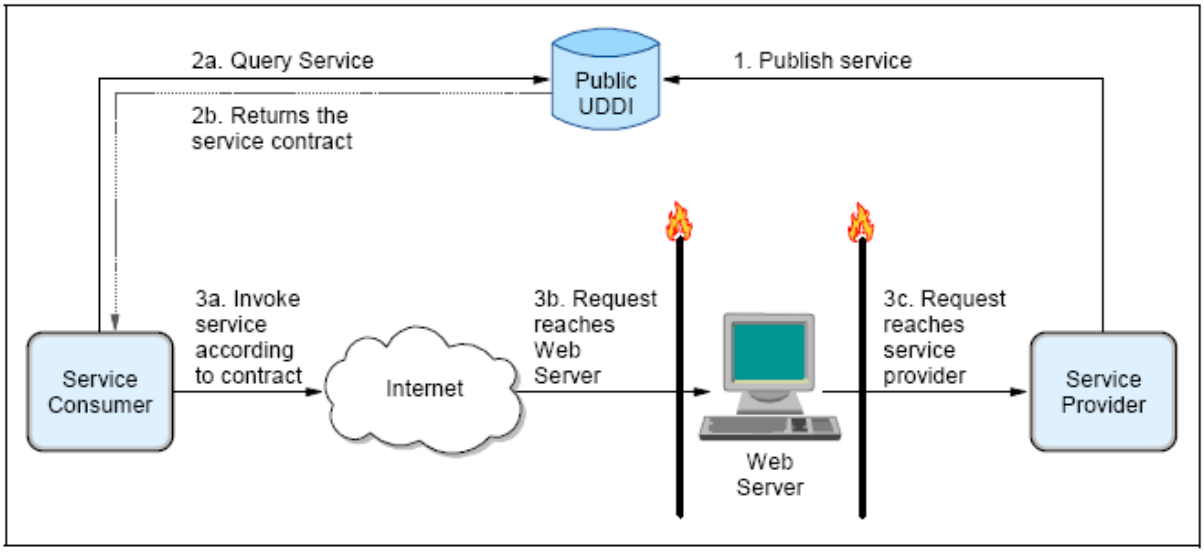
• **Publish** (Δημοσιοποίηση). Για να είναι προσβάσιμες, οι περιγραφές των υπηρεσιών θα πρέπει να δημοσιεύονται, έτσι ώστε ο αιτών της υπηρεσίας να μπορεί να τις βρει. Πού έχουν δημοσιευθεί μπορεί να ποικίλει ανάλογα με τις απαιτήσεις της εφαρμογής. Η καταχώρηση των απαραίτητων πληροφοριών για τη περιγραφή μιας υπηρεσίας επιτυγχάνεται μέσω της χρήσης της γλώσσας περιγραφής WSDL.

• **Find** (Εύρεση). Στη λειτουργία εύρεσης, ο αιτών της υπηρεσίας ανακτά μια περιγραφή της υπηρεσίας άμεσα ή μέσω ερωτημάτων στο μητρώο υπηρεσιών για τον τύπο της απαιτούμενης υπηρεσίας. Η λειτουργία εύρεσης μπορεί να συμμετέχει σε δύο διαφορετικές φάσεις του κύκλου ζωής για τον αιτούντα υπηρεσιών: 1) κατά το χρόνο σχεδίασης, για να ανακτήσει τη περιγραφή της διεπαφής της υπηρεσίας για την ανάπτυξη της εφαρμογής και 2) κατά το χρόνο εκτέλεσης, για να ανακτήσει τη θέση που βρίσκεται η περιγραφή της υπηρεσίας έτσι ώστε να την ενεργοποιήσει. Η επικρατούσα τεχνολογία καταλόγου ονομάζεται UDDI.

• **Bind/Invoke** (Δέσμευση/Ενεργοποίηση). Στη λειτουργία ενεργοποίησης, έχουμε την εγκαθίδρυση σύνδεσης μεταξύ αιτούμενου και παρόχου για την κλήση της κατάλληλης υπηρεσίας και αποστολή των αποτελεσμάτων [16].

Συνεπώς, οι ήδη υπάρχουσες εφαρμογές που χρησιμοποιούν τέτοιου είδους αρχιτεκτονική, μπορούν εύκολα να μετασχηματισθούν σε υπηρεσίες που να εξυπηρετούν άλλες υπάρχουσες ή και νέες εφαρμογές. Ακόμα, πολλές υπηρεσίες μπορούν να συνδυαστούν και να συνεργασθούν για την παραγωγή ενός αποτελέσματος με τρόπο διαφανή προς τον τελικό χρήστη, στον οποίο δίνεται η αίσθηση ότι κλήθηκε μία και μόνο υπηρεσία.

Για να γίνει περισσότερο αντιληπτό, ακολουθεί ένα τυπικό παράδειγμα αλληλεπίδρασης μιας υπηρεσίας διαδικτύου η οποία βασίζεται στο παραπάνω μοντέλο (Σχήμα 3.2).



Σχήμα 3.2 Αλληλεπίδραση μιας υπηρεσίας διαδικτύου.

Σύμφωνα με το παραπάνω σχήμα, ο πάροχος υπηρεσιών φιλοξενεί μια υπηρεσία διαδικτύου και ορίζει την περιγραφή της, έτσι ώστε να μπορεί να τη δημοσιεύει σε έναν αιτούντα υπηρεσίας ή στο μητρώο της υπηρεσίας. Ο αιτών της υπηρεσίας χρησιμοποιεί μια λειτουργία εύρεσης για να

ανακτήσει την περιγραφή της υπηρεσίας σε τοπικό επίπεδο ή από το μητρώο υπηρεσιών και χρησιμοποιεί την περιγραφή της υπηρεσίας για να συνδεθεί με τον πάροχο υπηρεσιών που προσφέρει την υπηρεσία και να αλληλεπιδράσει με την εφαρμογή.

3.4 Πρότυπα των Υπηρεσιών Διαδικτύου

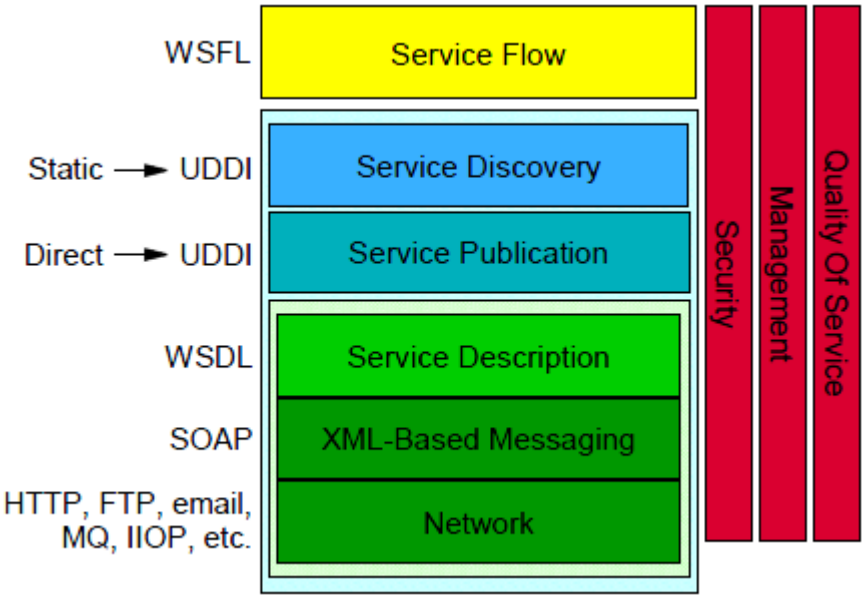
Οι υπηρεσίες διαδικτύου είναι μια συλλογή από τεχνολογικά πρότυπα και πρωτόκολλα. Πιο συγκεκριμένα, είναι XML αναπαραστάσεις προγραμμάτων, αντικειμένων ή κειμένων που είναι προσπελάσιμα μέσω Διαδικτύου για απ' ευθείας αλληλεπίδραση μεταξύ των εφαρμογών. Οι υπηρεσίες Ιστού μπορούν να προσπελαστούν με χρήση φυλλομετρητών (browsers) και παρέχουν ένα ανεξάρτητο από δεδομένα μηχανισμό παρουσίασης των υπηρεσιών με χρήση XML πρωτοκόλλων. Οι βασικές τεχνολογίες που χρησιμοποιούνται είναι :

- Extensible Markup Language (**XML**)– που περιλαμβάνει τη βασική XML και XML schemas
- Simple Object Access Protocol (**SOAP**) – που αποτελεί το πρωτόκολλο επικοινωνίας εφαρμογών βασισμένο σε XML
- Universal Description, Discovery and Integration (**UDDI**) – που είναι ο χώρος αποθήκευσης για την καταχώρηση και αναζήτηση των περιγραφών των υπηρεσιών διαδικτύου
- Web Services Description Language (**WSDL**) – που είναι ένα XML schema για την περιγραφή των μηνυμάτων, των λειτουργιών και των αντιστοιχήσεων των πρωτοκόλλων των υπηρεσιών διαδικτύου

Η χρήση αυτών των ανοικτών προτύπων, παρέχει ευρεία διαλειτουργικότητα μεταξύ διαφορετικών εφαρμογών λογισμικού, ανεξάρτητα από την πλατφόρμα και το framework στα οποία έχουν υλοποιηθεί, και χωρίς να απαιτούν καμία αλλαγή στον μηχανισμό του συστήματος. Οι πληροφορίες που αφορούν τις διαδικτυακές υπηρεσίες δημοσιεύονται, οπότε η εύρεση και η χρήση τους γίνεται ταχύτατα, ενώ μπορούν να δημιουργηθούν και να ανανεωθούν με εύκολο και οικονομικό τρόπο [5].

3.5 Η αρχιτεκτονική Στοίβα των Web Services

Για την εκτέλεση των τριών λειτουργιών (δημοσίευσης, εύρεσης, ενεργοποίησης) των υπηρεσιών διαδικτύου, με σκοπό την διάδοση τους μέσω διαδικτύου, χρησιμοποιείται μια αρχιτεκτονική στοίβα. Η στοίβα αυτή αποτελείται από μια συλλογή τεχνολογίων των Web Services δομημένη σε επίπεδα, όπου σε κάθε ένα επίπεδο ενσωματώνει τα αντίστοιχα πρότυπα τεχνολογίων. Το Σχήμα 3.3 παρουσιάζει μια εννοιολογική σχεδίαση της στοίβας των Web Services. Τα ανώτερα επίπεδα χτίζονται πάνω στις ικανότητες που παρέχονται από τα χαμηλότερα επίπεδα. Οι κάθετες στήλες αντιπροσωπεύουν τις απαιτήσεις που πρέπει να αντιμετωπιστούν σε κάθε επίπεδο της στοίβας. Το κείμενο στα αριστερά απεικονίζει το πρότυπο των τεχνολογιών που εφαρμόζεται σε κάθε στρώμα της στοίβας.



Σχήμα 3.3 Η αρχιτεκτονική Στοίβα των Web Services.

Έτσι, το πρώτο επίπεδο της στοίβας είναι το επίπεδο δικτύου (**Network**) που θεωρείται υπεύθυνο για την ανταλλαγή των μηνυμάτων ανάμεσα στις εφαρμογές. Οι διαδικτυακές υπηρεσίες όπως αναφέρθηκε παραπάνω, για να μπορέσουν να κληθούν από τον αιτούντα της υπηρεσίας πρέπει να βρίσκονται σε προσβάσιμο δίκτυο. Λόγω της πανταχού παρουσίας του, το HTTP (Hyper Text Transfer Protocol) είναι το στάνταρτ πρωτόκολλο δικτύου που χρησιμοποείται συνήθως για τη μεταφορά πληροφοριών και βρίσκεται στο χαμηλότερο επίπεδο της στοίβας. Άλλα δημοφιλή πρωτόκολλα διαδικτύου που μπορούν να υποστηριχθούν, είναι το SMTP (Simple Mail Transport Protocol) ή το FTP (File Transfer Protocol).

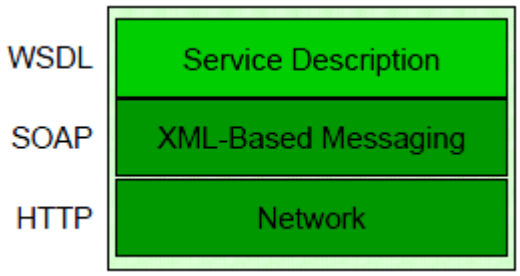
Το επόμενο στρώμα – **XML-Based Messaging**, βασίζεται σε XML μηνύματα και αντιπροσωπεύει την χρήση της XML ως βάση για το πρωτόκολλο ανταλλαγής μηνυμάτων. Το SOAP είναι το επιλεγμένο XMLπρωτόκολλο ανταλλαγής μηνυμάτων για τους εξής λόγους:

- Είναι ένας τυποποιημένος μηχανισμός ανταλλαγής μηνυμάτων και κλήσεων απομακρυσμένων διαδικασιών που χρησιμοποιούν την γλώσσα XML.
- Είναι απλό, και είναι βασικά μια μέθοδος HTTP POST για την αποστολή δεδομένων περιλαμβάνοντας ένα XML φάκελο που περικλείει όλες τις απαραίτητες πληροφορίες που πρέπει να σταλούν.
- Προτιμάται σε σχέση με την απλή μέθοδο HTTP POST της XML, επειδή προσδιορίζει ένα πρότυπο μηχανισμό για την ενσωμάτωση επεκτάσεων στο μήνυμα χρησιμοποιώντας κεφαλίδες SOAP και ένα πρότυπο κωδικοποίησης της λειτουργίας.
- Τα SOAP μηνύματα υποστηρίζουν τις λειτουργίες της δημοσίευσης, εύρεσης και ενεργοποίησης της αρχιτεκτονικής των διαδικτυακών υπηρεσιών.

Πάνω από το επίπεδο XML μηνυμάτων βρίσκεται το επίπεδο περιγραφής των υπηρεσιών διαδικτύου – **Service Description**. Το επίπεδο αυτό, βασίζεται στην προδιαγραφή WSDL, η οποία περιγράφει σε μορφή XML εγγράφων τις διαθέσιμες υπηρεσίες διαδικτύου. Με άλλα λόγια, η WSDL ορίζει τη διεπαφή και τους μηχανισμούς αλληλεπίδρασης των υπηρεσιών διαδικτύου.

Αυτά είναι τα **τρία** υποχρεωτικά επίπεδα για την δημιουργία διαλειτουργικών υπηρεσιών διαδικτύου και τη δημοσίευση τους στο Διαδίκτυο.

Έτσι, η απλούστερη στοίβα αποτελείται από το πρωτόκολλο HTTP για το στρώμα του δικτύου, το πρωτόκολλο SOAP για την ανταλλαγή των XML μηνυμάτων και τη WSDL για το στρώμα της περιγραφής των υπηρεσιών, όπως φαίνεται στο παρακάτω Σχήμα 3.4.



Σχήμα 3.4 Διαλειτουργικότητα της στοίβας των Web Services.

Η στοίβα που απεικονίζεται στο παραπάνω σχήμα, εξασφαλίζει τη διαλειτουργικότητα και επιτρέπει στις υπηρεσίες Ιστού να επικρατούν στην υπάρχουσα υποδομή του Διαδικτύου. Η ευελιξία δεν τίθεται σε κίνδυνο από την απαίτηση διαλειτουργικότητας, επειδή η πρόσθετη στήριξη μπορεί να παρέχεται από εναλλακτικές και επιπρόσθετες σημαντικές τεχνολογίες.

Ενώ τα κάτω τρία στρώματα της στοίβας επισημαίνουν τις ανάλογες τεχνολογίες που πρέπει να εφαρμόζονται σε κάθε επίπεδο της με σκοπό την επίτευξη της διαλειτουργικότητας, τα επόμενα τρία επίπεδα – **Service Publication, Service Discovery** και **Service Flow,** είναι προαιρετικά και χρησιμοποιούνται ανάλογα με τις επιχειρησιακές ανάγκες, όταν απαιτείται.

Στο επίπεδο της δημοσίευσης - **Service Publication** περιλαμβάνεται η ενέργεια που εκτελεί ο πάροχος μιας υπηρεσίας (service provider) έτσι ώστε ένα WSDL έγγραφο να γίνει διαθέσιμο στον υποψήφιο αιτούντα της υπηρεσίας (service requestor), σε κάθε στάδιο του κύκλου ζωής της υπηρεσίας του αιτούντα.

Το απλούστερο παράδειγμα σε αυτό το επίπεδο της δημοσίευσης των υπηρεσιών διαδικτύου, είναι όταν ο πάροχος υπηρεσιών αποστέλλει ένα WSDL έγγραφο απευθείας σε έναν αιτούντα της υπηρεσίας. Αυτό ονομάζεται άμεση δημοσίευση. Για παράδειγμα, η αποστολή ενός WSDL εγγράφου μέσω ενός μηνύματος ηλεκτρονικού ταχυδρομείου, είναι ένα παράδειγμα άμεσης δημοσίευσης. Δηλαδή, ένας αιτών μιας υπηρεσίας αφού εντοπίσει το υποψήφιο πάροχο της περιγραφόμενης υπηρεσίας συνδέεται απευθείας με αυτόν, καλεί την υποψήφια υπηρεσία και στη συνέχεια λαμβάνει το αντίστοιχο WSDL έγγραφο από τον πάροχο που το παρέχει μέσω Email. Η άμεση δημοσίευση είναι χρήσιμη για στατικές εφαρμογές. Εναλλακτικά, ο πάροχος υπηρεσιών μπορεί να δημοσιεύσει το WSDL έγγραφο που περιγράφει την υπηρεσία σε ένα φιλοξενούμενο τοπικό WSDL μητρώο, όπως είναι το ιδιωτικό μητρώο UDDI.

Επειδή μια υπηρεσία διαδικτύου δεν μπορεί να ανακαλυφθεί εάν δεν έχει πρώτα δημοσιευθεί, η ανακάλυψη υπηρεσιών εξαρτάται από τη δημοσίευσή τους. Έτσι, στο αμέσως επόμενο επίπεδο, βρίσκεται το επίπεδο της ανακάλυψης μιας υπηρεσίας - **Service Discovery.** Η ποικιλία των μηχανισμών ανακάλυψης σε αυτό το επίπεδο είναι παράλληλη με το σύνολο των μηχανισμών δημοσίευσης. Κάθε μηχανισμός που επιτρέπει στον αιτούντα της υπηρεσίας να αποκτήσει πρόσβαση στην περιγραφή των υπηρεσιών και τον θέτει στη διάθεση της εφαρμογής λογισμικού κατά το χρόνο εκτέλεσης της, μπορεί να θεωρηθεί ως ανακάλυψη υπηρεσιών. Το απλούστερο παράδειγμα της ανακάλυψης είναι η στατική ανακάλυψη, όπου ο αιτών της υπηρεσίας ανακτά ένα WSDL έγγραφο από ένα τοπικό αρχείο. Αυτό είναι συνήθως το WSDL έγγραφο που λαμβάνεται μέσω μιας άμεσης δημοσίευσης ή μέσω αποτελεσμάτων

προηγούμενης λειτουργίας αναζήτησης. Εναλλακτικά, η υπηρεσία μπορεί να ανακαλυφθεί κατά το χρόνο σχεδίασης ή της εκτέλεσης χρησιμοποιώντας ένα τοπικό WSDL μητρώο, ένα ιδιωτικό μητρώο UDDI.

Στο ανώτερο επίπεδο, βρίσκεται το επίπεδο της ροής υπηρεσιών - **Service Flow**, το οποίο παρέχει τη σύνθεση των υπηρεσιών σε ροές εργασίας (workflows) με σκοπό την αναπαράστασης τους ως μια υψηλότερου επιπέδου υπηρεσία διαδικτύου. Μια σύνθεση των υπηρεσιών του Παγκοσμίου Ιστού θα μπορούσε να παίξει ένα ή περισσότερους ρόλους. Οι υπηρεσίες διαδικτύου που χρησιμοποιούν οι επιχειρήσεις θα μπορούσαν να συνεργαστούν για να παρουσιαστεί μια ενιαία διεπαφή των υπηρεσιών διαδικτύου για το κοινό, ή οι υπηρεσίες που παρέχονται από διάφορες επιχειρήσεις θα μπορούσαν να συνεργαστούν για την εκτέλεση business-to-business συναλλαγών. Δηλαδή, μια ροή εργασίας μπορεί να καλέσει οποιαδήποτε διαδικτυακή υπηρεσία, αρκεί να συμμετέχει σε μια επιχειρηματική διαδικασία. Με άλλα λόγια, το επίπεδο της ροής υπηρεσιών, περιγράφει πως μια υπηρεσία επικοινωνεί, συνεργάζεται με μια άλλη υπηρεσία καθώς και ποιες ροές εκτελούνται. Η WSFL[9] (Web Services Flow Language) χρησιμοποιείται για να περιγράψει αυτές τις αλληλεπιδράσεις.

Τέλος, αξίζει να επισημάνουμε ότι, σε όλα τα παραπάνω επίπεδα της στοίβας, οι εφαρμογές που χρησιμοποιούν διαδικτυακές υπηρεσίες, πρέπει να παρέχουν ασφάλεια (**security**), διαχείριση (**management**) και ποιότητα των παρεχόμενων υπηρεσιών (**quality of service**). Αυτές οι απαιτήσεις πρέπει να αντιμετωπίζονται σε κάθε επίπεδο της στοίβας. Τα αποτελέσματα κάθε επιπέδου μπορούν να είναι ανεξάρτητα από τα άλλα [16].

Πιο αναλυτικά, η ασφάλεια είναι ένα από τα σημαντικότερα ζητήματα των υπηρεσιών διαδικτύου. Σε πιο παραδοσιακές εφαρμογές τύπου client/server, ο έλεγχος για την ασφάλεια της αναταλλασσόμενης πληροφορίας μπορούσε να γίνει πιο εύκολα. Όμως, στις μέρες μας, όπου η πληροφορία εκτίθεται σε λιγότερο προστατευμένα περιβάλλοντα όπως αυτά των web services, κρίνεται απαραίτητο η πρόληψη αυξημένων μέτρων ασφαλείας για την προστασία των δεδομένων που αναταλλάσσονται μεταξύ διαδικτυακών εφαρμογών.

Επειδή οι υπηρεσίες διαδικτύου αφορούν συνήθως μεγάλης κλίμακας κατανεμημένες εφαρμογές, η ευελιξία στην διαχείριση τους κρίνεται απαραίτητη. Και αυτό, γιατί όσο πιο εύκολα και αυτόματα μπορεί μια διαδικτυακή υπηρεσία να προσαρμοστεί σε αναμενόμενες αλλαγές μιας εφαρμογής, τόσο πιο ποιοτική και αξιόπιστη θεωρείται. Με αυτό το τρόπο, οι υπηρεσίες που

[9] Είναι μια XML γλώσσα για την περιγραφή της λειτουργικότητας που παρέχεται από μια συλλογή υπηρεσιών διαδικτύου για την επίτευξη συγκεκριμένων επιχειρηματικών αναγκών.

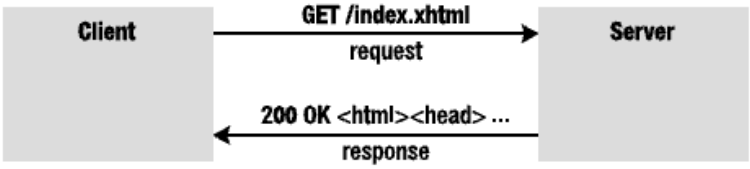
εξελίσσονται με αποδοτικό τρόπο μπορούν εύκολα να χειριστούν πλήθος αλληλεπιδράσεων με το χρήστη.

3.6 RESTful Web Services

Για να κατανοήσουμε τι ακριβώς είναι οι RESTful υπηρεσίες Ιστού, θα αναφερθούμε αρχικά, στο τι είναι το REST. Το REST (Representational State Transfer) είναι ένα αρχιτεκτονικό στυλ για κατανεμημένα συστήματα υπερμέσων όπως ο Παγκόσμιος Ιστός. Παρουσιάστηκε για πρώτη φορά το 2000 στη διδακτορική διατριβή του Roy Thomas Fielding και αναπτύχθηκε από την ομάδα του W3C. Παρέχει ένα σύνολο αρχιτεκτονικών περιορισμών που όταν εφαρμόζονται σε μια υπηρεσία Ιστού, τονίζουν την επεκτασιμότητα των αλληλεπιδράσεων των συστατικών, τη γενικότητα των διεπαφών, την ανεξάρτητη ανάπτυξη συστατικών μερών, και των συστατικών του διαμεσολαβητή για μείωση του χρόνου αλληλεπίδρασης και την επιβολή της ασφάλειας [17].

Στο REST, κάθε κομμάτι πληροφορίας στο Διαδίκτυο είναι ένας πόρος (resource), ο οποίος αναπαριστάται ως ένα URI. Αυτοί οι πόροι ενεργοποιούνται χρησιμοποιώντας καλά ορισμένες HTTP μεθόδους (POST, GET, PUT, DELETE). Έτσι, κάθε υπηρεσία Ιστού που υπακούει στο αρχιτεκτονικό στύλ REST, είναι μια RESTful υπηρεσία Ιστού. Όπως κάθε υπηρεσία Ιστού, οι RESTful υπηρεσίες λαμβάνουν μια αίτηση η οποία ορίζει τη διαδικασία που πρέπει να εκτελεστεί, και στη συνέχεια επιστρέφει μια απάντηση που περιέχει το αποτέλεσμα της εκτέλεσης της αίτησης.

Η αρχιτεκτονική REST αποτελείται από εξυπηρετητές (servers) και πελάτες (clients). Οι πελάτες είναι αυτοί που εκκινούν την αποστολή των αιτημάτων τους στους εξυπηρετητές, οι οποίοι στη συνέχεια επεξεργάζονται αυτά τα αιτήματα έτσι ώστε να επιστρέψουν και τις ανάλογες απαντήσεις. Κάθε αίτημα που πραγματοποιείται, αποστέλλεται σε ένα συγκεκριμένο πόρο. Κάθε πόρος έχει μοναδική διεύθυνση στην οποία αντιστοιχίζεται το κάθε αίτημα. Όπως φαίνεται και στο Σχήμα 3.5, για τη δεδομένη χρονική στιγμή πραγματοποίησης ενός αιτήματος, οι απαντήσεις που αποστέλλονται από τους εξυπηρετητές στους πελάτες εμφανίζουν αναπαραστάσεις συγκεκριμένων πόρων. Αυτό σημαίνει ότι, με κάθε νέο αίτημα επιστρέφεται και μία ανάλογη απόκριση στον πελάτη με τη μορφή αναπαράστασης προκειμένου ο ίδιος ο πελάτης να μεταβεί σε μία νέα κατάσταση [17, 18].



Σχήμα 3.5 HTTP μήνυμα αίτησης και απόκρισης

Η προσέγγιση των RESTful υπηρεσιών διαδικτύου αποτελείται από τις εξής έννοιες :

- Το **πόρο** (resource)

Το κλειδί στις RESTful αρχιτεκτονικές, είναι οι πόροι. Ένας πόρος μπορεί να είναι οτιδήποτε στο οποίο ένας πελάτης μπορεί να αναφερθεί ή να αλληλεπιδράσει με αυτό και γενικά κάθε είδους πληροφορία προσβάσιμη μέσω διευθύνσεων του διαδικτύου (URL). Π.χ. ένας πόρος μπορεί να είναι ένα αρχείο, μια εικόνα, ένας διαδικτυακός τόπος κ.α. Μια διαδικτυακή εφαρμογή μπορεί να αποτελείται από μια συλλογή τέτοιων πόρων. Επιπλέον, ορισμένες υπηρεσίες ασχολούνται με περισσότερους από έναν τύπο πόρων.

- την **αναπαράσταση** (representation)

Η αναπαράσταση είναι κάθε χρήσιμη πληροφορία σχετικά με την κατάσταση ενός πόρου. Τεχνικά, μια αναπαράσταση είναι ένα serialization πόρων σε μια συγκεκριμένη μορφή, όπως XML, XHTML (Extensible Hypertext Markup Language), JSON[10](JavaScript Object Notation), RDF[11] κ.α. Οι γλώσσες που βασίζονται στην XML είναι ιδιαίτερα σημαντικές στο σημασιολογικό πλαίσιο, επειδή επιτρέπουν επεκτάσιμο σχολιασμό δεδομένων.

- το **ενιαίο αναγνωριστικό** (uniform identifier)

Κάθε πόρος συνδέεται με ένα τουλάχιστον URI (Uniform Resource Identifier), που λειτουργεί ταυτόχρονα ως το όνομά του και ως αναγνωριστικό. Τα αναγνωριστικά θα έπρεπε να είναι περιγραφικά, και σύμφωνα με προκαθορισμένους περιορισμούς που περιλαμβάνουν τη δομή, την ιεραρχία και τον τρόπο γραφής τους.

- την **ενιαία διεπαφή** (uniform interface)

Κάθε ενέργεια που εκτελείται ορίζεται από το πρωτόκολλο HTTP. Το πρωτόκολλο REST, όπως αναφέραμε και προηγουμένως προσφέρει πέντε κύριες HTTP μεθόδους: GET, HEAD, POST, PUT και DELETE. Αυτοί οι μέθοδοι εφαρμόζονται σε πόρους κατά τη διάρκεια εκτέλεσης HTTP

[10] Προέρχεται από τη γλώσσα Javascript και είναι ένα ανοικτό πρότυπο σχεδιασμένο για την αναπαράσταση απλών δομών δεδομένων που ονομάζονται αντικείμενα.

[11] Είναι το επίσημο πρότυπο του W3C για την κωδικοποίηση μεταδεδομένων και άλλων μοντέλων δεδομένων στο Σημασιολογικό Ιστό.

αιτημάτων. Πιο συγκεκριμένα, μια HTTP μέθοδος εκτελείται από ένα συγκεκριμένο URI. Παρά την απλότητά του, αυτό το χαρακτηριστικό είναι εξαιρετικά ισχυρό, διότι ορίζει μια ενιαία διεπαφή για όλες τις πιθανές υπηρεσίες. Εάν μια εφαρμογή πελάτης γνωρίζει τους πόρους που προσφέρονται από μια συγκεκριμένη υπηρεσία, αυτόματα θα ξέρει πώς να ανακτήσει, να δημιουργήσει, να ενημερώσει και να διαγράψει αυτούς τους πόρους.

Οι συνηθέστεροι HTTP μέθοδοι που χρησιμοποιούν οι RESTful εφαρμογές στις κλήσεις των μηνυμάτων αλληλεπίδρασης για τη δημοσίευση (δημιουργία/ενημέρωση), ανάγνωση και διαγραφή των δεδομένων τους μέσω URIs είναι οι εξής:

1. **POST** : Οι κλήσεις POST αλλάζουν την κατάσταση ενός πόρου (π.χ. προσθήκη προϊόντος σε ένα καλάθι αγοράς)
2. **GET** : Οι κλήσεις GET ζητούν την αναπαράσταση ενός πόρου
3. **PUT** : Οι κλήσεις PUT δημιουργούν νέους πόρους ή ανανεώνουν το περιεχόμενο ενός υπάρχοντος πόρου (π.χ. η ανανέωση της διεύθυνσης ενός πελάτη)
4. **DELETE** : Με την κλήση DELETE γίνεται η διαγραφή ενός πόρου

Επιπλέον, η προσέγγιση των RESTful υπηρεσιών διαδικτύου ορίζει ότι οι υπηρεσίες διαδικτύου πρέπει να διέπονται από τις ακόλουθες **αρχές** :

- ✓ **διευθυνσιοδότηση** (Addressability)

Μια διαδικτυακή υπηρεσία ταξινομείται ως κατευθυνόμενη αν εκθέτει τις ενδιαφέρουσες πτυχές των δεδομένων της μέσα από τους πόρους, καθένα με το δικό του μοναδικό URI. Λαμβάνοντας υπόψη ότι ένας μοναδικός πόρος μπορεί να υποστηρίξει διάφορες αναπαραστάσεις, είναι χρήσιμο να δοθεί ένα διαφορετικό URI σε κάθε αναπαράσταση.

- ✓ **Άνευ κατάστασης** (Stateless)

Κάθε αίτηση που πραγματοποιείται από τον client στον server πρέπει να περιέχει όλες τις πληροφορίες που είναι αναγκαίες για την κατανόηση του αιτήματος και μόνο. Τέτοιες πληροφορίες περιγράφουν πληροφορίες που είναι σχετικές με τον πόρο αλλά και με την διαδρομή που έχει ακολουθήσει ο πελάτης μέσω της εφαρμογής. Όμως, ο εξυπηρετητής δεν πρέπει ποτέ να γνωρίζει την κατάσταση του πελάτη, δηλαδή δεν θα πρέπει να γνωρίζει τι έγινε με μια προηγούμενη κλήση τη στιγμή που επεξεργάζεται την τρέχουσα. Ο πελάτης πρέπει να χειρίζεται μόνος του την κατάσταση του και τα session του. Αυτή η «έλλειψη κατάστασης» σε

μια REST υπηρεσία απαλλάσσει τον εξυπηρετητή από την ανάγκη συγχρονισμού των δεδομένων μιας συνόδου με μια εξωτερική εφαρμογή.

Για να γίνει περισσότερο αντιληπτό, ακολουθεί ένα **παράδειγμα**.

Υποθέτουμε ότι ένα html έγγραφο μεταφέρεται από έναν εξυπηρετητή σε ένα πελάτη. Το αρχείο αυτό μπορεί να περιέχει, όπως είπαμε και προηγουμένως, συνδέσμους οι οποίοι να αναφέρονται σε άλλους πόρους στο Διαδίκτυο. Ο πελάτης μπορεί είτε να παραμείνει στον ίδιο πόρο είτε να πλοηγηθεί σε άλλους πόρους του αρχείου που πιθανόν να υπάρχει περιορισμένη πρόσβαση (π.χ. μετάβαση σε μια ιστοσελίδα αγοράς όπου απαιτείται εισαγωγή στοιχείων πιστωτικής κάρτας). Εάν επιλέξει την δεύτερη περίπτωση, τότε αλλάζει η κατάσταση του πόρου γιατί μεταβαίνει σε ένα άλλο διαδικτυακό τόπο. Σε αυτή τη περίπτωση, το session που έχει δημιουργηθεί για τον συγκεκριμένο πελάτη, πρέπει να διατηρηθεί εξ' ολοκλήρου σε αυτόν.

Αυτός ο περιορισμός προωθεί την βελτίωση των ιδιοτήτων της ορατότητας (visibility), της αξιοπιστίας (reliability) και της επεκτασιμότητας (scalability). Η ορατότητα βελτιώνεται διότι ένα σύστημα παρακολούθησης δεν χρειάζεται να κοιτάει πέρα από ένα ενιαίο σημείο αναφοράς του αιτήματος, προκειμένου να καθοριστεί η πλήρης φύση του αιτήματος. Η αξιοπιστία βελτιώνεται, διότι διευκολύνεται το έργο της ανάκαμψης από τις επιμέρους αποτυχίες. Η επεκτασιμότητα βελτιώνεται επειδή δεν χρειάζεται να αποθηκευτεί καμία κατάσταση μεταξύ των αιτημάτων στον διακομιστή και επιπλέον απλοποιεί περαιτέρω την υλοποίησή τους, επειδή ο διακομιστής δεν απαιτείται να διαχειρίζεται τη χρήση των πόρων των αιτημάτων. Αυτό σημαίνει ότι ο πελάτης είναι υπεύθυνος για την αποστολή όλων των πληροφοριών που απαιτούνται για την επιτυχή εκτέλεση της αίτησης [17, 18].

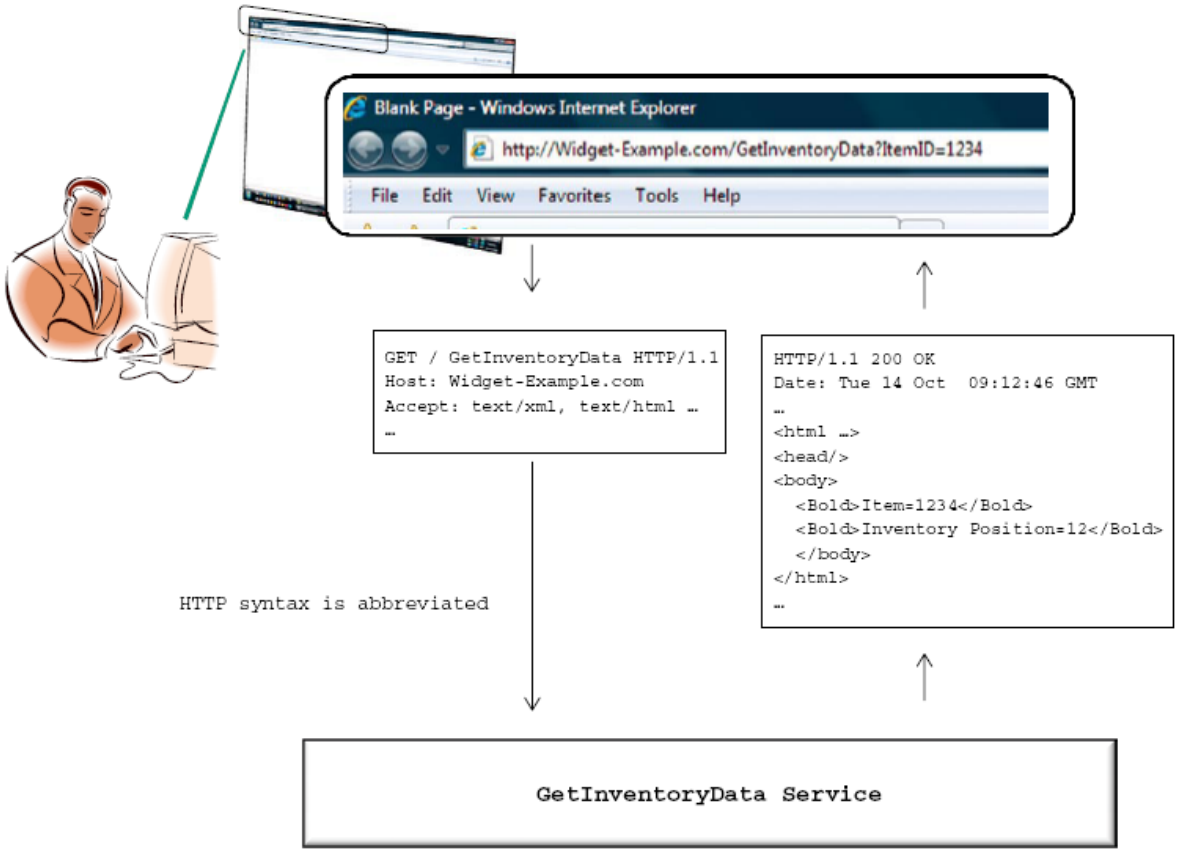
- ✓ **συνεκτικότητα** (Connectedness)

Για να υπάρχει συνεκτικότητα, οι πόροι θα πρέπει να δείχνουν σε άλλους πόρους, καθοδηγώντας τον πελάτη σε άλλες καταστάσεις. Μια σύνδεση πραγματοποιείται κάθε φορά που μια αναπαράσταση πόρων παρέχει ένα μοναδικό αναγνωριστικό, δηλαδή ένα URI ενός άλλου πόρου. Σε μια μη συνδεδεμένη υπηρεσία, ο πελάτης πρέπει να χρησιμοποιεί προκαθορισμένους κανόνες για την κατασκευή κάθε URI που θέλει να επισκεφθεί [19].

Οι αρχές αυτές περιγράφουν την αρχιτεκτονική των συστημάτων και των αλληλεπιδράσεων που απαρτίζουν το Web.

Παρακάτω ακολουθεί ένα απλοικό **παράδειγμα ενός RESTful αιτήματος** που στέλνεται από έναν πελάτη σε έναν εξυπηρετητή (Σχήμα 3.6). Σύμφωνα με αυτό, από αριστερά παρουσιάζεται το αίτημα του πελάτη και από δεξιά το αίτημα απάντησης από τον εξυπηρετητή.

Αρχικά, ένας χρήστης πληκτρολογεί με τη βοήθεια ενός browser δηλαδή, ενός προγράμματος πελάτη τη URL διεύθυνση με την οποία θέλει να συνθεθεί. Η URL διέθυνση αντιστοιχεί σε ένα συγκεκριμένο URI και τύπο πόρου, εδώ HTML. Όταν λοιπόν ο χρήστης πατήσει Enter, ο browser στέλνει ένα HTTP GET αίτημα στον εξυπηρετητή που φιλοξενεί το πόρο, ζητώντας την υπηρεσία GetInvetoryData, όπως φαίνεται στο σχήμα. Στη συνέχεια, ο εξυπηρετητής λαμβάνει το αίτημα, το επεξεργάζεται και στέλνει ένα αίτημα απάντησης στο browser το οποίο περιέχει τον αριθμό απάντησης του αιτήματος (200 OK), την επικεφαλίδα (Date) και το κυρίως σώμα της επιστρεφόμενης ιστοσελίδας που περικλείεται μέσω των html tags (<html></html>). Αξίζει να σημειώσουμε ότι ένα αίτημα που χρησιμοποιεί τη μέθοδο GET δεν έχει ποτέ κυρίως σώμα (body), για αυτό και εδώ δεν υπάρχει.



Σχήμα 3.6 Μια απλή RESTful αλληλεπίδραση

3.7 Σύγκριση SOAP με REST

Όπως περιγράψαμε εξ'αρχής, υπάρχουν δύο τρόποι για την ανάπτυξη των υπηρεσίων Διαδικτύου: η παραδοσιακή, βασισμένη στο πρωτόκολλο «SOAP» και η εννοιολογικά απλούστερη και πιο σύγχρονη, βασισμένη στο μοντέλο «REST». Το SOAP είναι πραγματικά ένα πρωτόκολλο για XML-based κατανεμημένα συστήματα πληροφορικής αλλά με περίπλοκη μορφή, ενώ το REST είναι ένα απλό, καθαρό και καλά-αποδεδειγμένο μοντέλο. Δεν απαιτεί μια σημαντική σειρά πολύπλοκων προτύπων. Ως αποτέλεσμα, μπορεί να έχει και κάποια πλεονεκτήματα απόδοσης πάνω από τις υπηρεσίες Ιστού [20].

Μερικά από τα πλεονεκτήματα και αδυναμίες αυτών περιγράφονται παρακάτω [20]:

Πρωτόκολλο SOAP

Πλεονεκτήματα :

- ✓ Σχεδιασμένο για να χειρίζεται κατανεμημένα υπολογιστικά περιβάλλοντα
- ✓ Είναι το πρότυπο που επικρατεί για τις υπηρεσίες web, και ως εκ τούτου έχει καλύτερη υποστήριξη από άλλα πρότυπα (WSDL, WS-*[12])
- ✓ Ενσωματωμένη αντιμετώπιση των λαθών
- ✓ Επεκτασιμότητα

Μειονεκτήματα :

- ✓ θεωρητικά πιο δύσκολο από το REST
- ✓ Πιο αναλυτικό
- ✓ Πιο δύσκολο να αναπτυχθεί, απαιτεί τη χρήση μέσων

Πρωτόκολλο REST

Πλεονεκτήματα:

- ✓ Πολύ πιο απλό να αναπτυχθεί από ότι το SOAP
- ✓ Εύκολη εκμάθηση, λιγότερη εξάρτηση από τα εργαλεία
- ✓ Συνοπτικό

[12] Χρησιμοποιείται για να δείξει τις προδιαγραφές που σχετίζονται με τις υπηρεσίες Ιστού. Λόγω του ότι υπάρχουν πολλά πρότυπα, όπως WS-Addressing, WS-Discovery, WS-Federation, WS-Policy, WS-Security και WS-Trust χρησιμοποιεί το * για να τα συμπεριλάβει όλα.

- ✓ Πιο κοντά στο σχεδιασμό και τη φιλοσοφία του Web

Μειονεκτήματα:

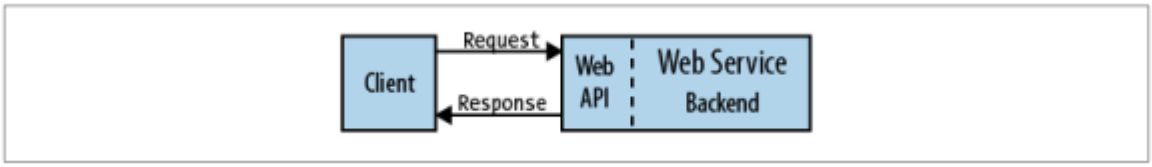
- ✓ Προϋποθέτει μία επικοινωνία σημείο-προς-σημείο και όχι μεταξύ ενδιάμεσων - δεν μπορεί δηλαδή, να χρησιμοποιηθεί για κατανεμημένο υπολογιστικό περιβάλλον όπου το μήνυμα μπορεί να περάσει μέσα από έναν ή περισσότερους ενδιάμεσους
- ✓ Έλλειψη προτύπων υποστήριξης για την ασφάλεια, την πολιτική, την αξιόπιστη ανταλλαγή μηνυμάτων
- ✓ Συνδεδεμένο με το μοντέλο μεταφοράς HTTP

Κεφάλαιο 4
Μεθοδολογία

Σε αυτό το κεφάλαιο αναλύουμε τις νέες τεχνολογίες (web APIs) και πρωτόκολλα (OAuth 2.0) που έχουν σήμερα προταθεί για την ανάπτυξη των υπηρεσιών διαδικτύου καθώς και η περιγραφή της προτεινόμενης μεθοδολογίας που εφαρμόστηκε για τη σχεδίαση και ανάπτυξη μιας διαδικτυακής υπηρεσίας βασισμένης σε τελευταίες τεχνολογίες και πρότυπα.

4.1 Τι είναι τα web APIs

Τα web APIs (Application Programming Interfaces) αποτελούν εξέλιξη στις υπηρεσίες Ιστού (Web 2.0) όπου έμφαση έχει δοθεί στην απομάκρυνση από τις υπηρεσίες Ιστού που βασίζονται σε υλοποίηση SOAP έναντι εκείνων που βασίζονται σε REST. Ένα web API είναι μια διασύνδεση που διευκολύνει την επικοινωνία, επιτρέποντας σε διαφορετικές εφαρμογές να αλληλεπιδρούν αποτελεσματικά με αυτή, όπως φαίνεται στο Σχήμα 4.1 [21].



Σχήμα 4.1 Αναπαράσταση ενός Web API

Από τεχνικής πλευράς, ένα Web API είναι μια βιβλιοθήκη που περιλαμβάνει προδιαγραφές για ρουτίνες, δομές δεδομένων, κλάσεις αντικειμένων και μεταβλητές, διευκολύνοντας την αλληλεπίδραση μεταξύ διαφορετικών εφαρμογών επιτρέποντας την ανταλλαγή πληροφορίας. Όμως, αξίζει να σημειώσουμε ότι, οι προγραμματιστές θα πρέπει να διασφαλίζουν την ακεραιότητα του κώδικα υλοποίησης των Web APIs που δημιουργούν, αποφεύγοντας έτσι τη δυνατότητα πρόσβασης άλλων εφαρμογών στο κώδικα που τα υλοποιεί [22].

Όταν χρησιμοποιείται στο πλαίσιο της ανάπτυξης ιστοσελίδων, ένα web API συνήθως ορίζεται ως ένα σύνολο Hypertext Transfer Protocol (HTTP) μηνυμάτων αιτήματος, σύμφωνα με τον ορισμό της δομής των μηνυμάτων απάντησης, η οποία είναι συνήθως σε XML ή JSON μορφή [23].

Ειδικότερα, στο περιεχόμενο ενός API:

- ✓ γίνεται επίκληση μέσω του Διαδικτύου
- ✓ σχεδόν πάντα χρησιμοποιεί το HTTP (ή HTTPS) ως πρωτόκολλο επικοινωνίας
- ✓ συχνά χρησιμοποιεί XML για να αντιπροσωπεύσει μια απάντηση
- ✓ συχνά χρησιμοποιεί είτε HTTP παραμέτρους ερωτήματος ή κάποιο XML για να αντιπροσωπεύσει ένα αίτημα

Για αυτούς τους λόγους, ένα καλό API συχνά δεν απαιτεί κανένα ιδιαίτερο client-side λογισμικό. Για να είναι επιτυχημένο αρκεί να μπορεί να χρησιμοποιηθεί από πολλούς διαφορετικούς τύπους πελατών στο Διαδίκτυο, από κυριολεκτικά οποιοδήποτε περιβάλλον προγραμματισμού και να είναι αρκετά απλό [22].

Στη πράξη λοιπόν, η χρησιμοποίηση των APIs επιτρέπει σε διαδικτυακές πλατφόρμες να δημιουργήσουν μια ανοικτή αρχιτεκτονική για την ανταλλαγή περιεχομένου και δεδομένων μεταξύ τους αλλά και μεταξύ των εφαρμογών. Με τον τρόπο αυτό, το περιεχόμενο που δημιουργείται σε ένα μέρος μπορεί να προβάλλεται και να ενημερώνεται σε διάφορες τοποθεσίες στο διαδίκτυο. Αντιπροσωπευτικά παραδείγματα, αποτελούν οι πλατφόρμες κοινωνικής δικτύωσης Twitter και Facebook, οι οποίες εξαιτίας της δημοσίευσης των APIs τους,

παρέχουν στους χρήστες τους εύκολη και γρήγορη πρόσβαση από οποιαδήποτε συσκευή (iphone, smart phone, Η/Υ) και παράλληλα γίνονται περισσότερο δημοφιλές [22, 24].

Ένα Web API που υπακούει στο αρχιτεκτονικό στυλ του REST, είναι ένα REST API. Έχοντας ένα REST API, μια υπηρεσία ιστού γίνεται RESTful. Στην εργασία αυτή, τα REST APIs που βοήθησαν στην ανάπτυξη της εφαρμογής είναι: το MediaWiki REST API, Moodle REST API, και WordPress REST API για την πρόσβαση κάθε εξωτερικής υπηρεσίας και χρηστών καθώς και την ένταξη αυτών των RESTful υπηρεσιών στο Moodle.

Το REST API προσφέρει 7 διαφορετικούς τρόπους ελέγχου της ταυτότητας και εξουσιοδότησης σε μια εφαρμογή που θέλει να αλληλεπιδράσει με μια άλλη. Εμείς χρησιμοποιήσαμε τον τρόπο που προορίζεται για χρήση από web εφαρμογές μιας και έχουμε τρεις διαφορετικές web πλατφόρμες. Η πιστοποίηση (authentication) και εξουσιοδότηση (authorization) σε αυτή τη περίπτωση γίνεται μέσω του πρωτοκόλλου OAuth 2.0.

4.2 Το Πρωτόκολλο OAuth 2.0

Το πρωτόκολλο OAuth 2.0 είναι ένα ανοιχτό πρωτόκολλο επικοινωνίας μεταξύ εφαρμογών που επιτρέπει την πιστοποίηση χρηστών χρησιμοποιώντας APIs. Αναπτύχθηκε από τον οργανισμό Internet Engineering Task Force (IETF OAuth Working Group) και αποτελεί εξέλιξη του OAuth, λόγω του ότι διατηρεί τη συνολική αρχιτεκτονική και προσέγγιση που υιοθετούν προηγούμενες εκδόσεις (OAuth 1.0). Το πρωτόκολλο OAuth 2.0 παρέχει ένα αυξημένο επίπεδο ασφαλείας, γιατί η πιστοποίηση του χρήστη γίνεται στο πάροχο της υπηρεσίας με αποτέλεσμα να μην γνωστοποιούνται τα προσωπικά του διαπιστευτήρια (username και password) σε τρίτες εφαρμογές [20]. Η διαδικασία πιστοποίησης πραγματοποιείται μέσω ενός μοναδικού κλειδιού (API token) που μπορεί να συσχετιστεί με οποιονδήποτε λογαριασμό χρήστη μιας εφαρμογής [25].

Στη περίπτωση μας, επιτρέπει στους χρήστες να μοιράζονται προσωπικά τους δεδομένα (άρθρα, συζητήσεις, σχόλια κ.α.) που είναι αποθηκευμένα σε μια διαδικτυακή εφαρμογή, (π.χ. MediaWiki) με μια άλλη εφαρμογή (π.χ. Moodle) χωρίς να χρειάζεται να δώσουν στην δεύτερη εφαρμογή τους κωδικούς τους. Δηλαδή, αντί για το username και password δίνουν κάποια API tokens, τα οποία παρέχουν συγκεκριμένα δικαιώματα πρόσβασης και για περιορισμένη χρονική διάρκεια.

Τα «**API tokens**» είναι μοναδικά 32 ψηφίων αλφαριθμητικά αναγνωριστικά που εκδίδονται από το διακομιστή και χρησιμοποιούνται από την εφαρμογή πελάτη για να συνδέσει επικυρωμένα αιτήματα με τον ιδιοκτήτη των πόρων. Πιο συγκεκριμένα, είναι αναγνωριστικά που δημιουργούνται μια φορά και χρησιμοποιούνται σε κάθε HTTP αίτηση με σκοπό την πιστοποίηση και εξουσιοδότηση του χρήστη του εν λόγω αιτήματος. Τα API tokens έχουν ένα αντίστοιχο κοινό μυστικό κλειδί που χρησιμοποιείται από την εφαρμογή πελάτη για να καθορίσει την κυριότητα του token καθώς και την εξουσιοδότηση να εκπροσωπεί τον ιδιοκτήτη των πόρων.

Ένα **παράδειγμα token** είναι : **Token** : 12345678-1234-1234-1234-1234567890ab

Ο σκοπός λοιπόν αυτής της διαδικασίας ελέγχου πιστοποίησης, γίνεται για την αναγνώριση του χρήστη που υποβάλλει την αίτηση, δηλαδή για να ελέγξει αν ο χρήστης είναι πιστοποιημένος για να μπορέσει να εκτελέσει την αιτούμενη ενέργεια [25].

Το OAuth 2.0 ορίζει τέσσερις τύπους χορήγησης άδειας (grant types): κωδικό εξουσιοδότησης (authorization code), απλοποιημένο κωδικό εξουσιοδότησης (implicit authorization code), κωδικό διαπιστευτηρίων του ιδιοκτήτη των πόρων (resource owner password credentials) και αυτή των διαπιστευτηρίων πελάτη (client credentials). Επιπλέον, παρέχει ένα μηχανισμό για τον καθορισμό επιπλέον τύπων χορήγησης άδειας, όπου κάθε ένας από αυτούς, ορίζει μια εξουσιοδοτημένη άδεια αλληλεπίδρασης μεταξύ των τεσσάρων μερών: πελάτη, ιδιοκτήτη των πόρων, διακομιστή αδείας και διακομιστή πόρων [26].

4.2.1 Ρόλοι στο πρωτόκολλο OAuth 2.0

Το πρωτόκολλο OAuth 2.0, ορίζει τέσσερις ρόλους που εργάζονται από κοινού για τη χορήγηση και την παροχή πρόσβαση σε προστατευόμενους - περιορισμένους πόρους που απαιτούν έλεγχο ταυτότητας [20]. Αυτοί είναι :

- **Εφαρμογή Πελάτης** (client application)
  Είναι μια HTTP εφαρμογή που ζητάει πρόσβαση στους πόρους που είναι αποθηκευμένοι στο διακομιστή πόρων χρησιμοποιώντας ταυτοποιημένες αιτήσεις.

- **Ιδιοκτήτης των πόρων** (resource owner)
  Είναι ένας χρήστης ή μια εφαρμογή ικανή να επιτρέπει την πρόσβαση και τον έλεγχο των

προστατευόμενων πόρων που πρόκειται να μοιραστεί, χρησιμοποιώντας πιστοποιήσεις για τον έλεγχο ταυτότητας με το διακομιστή αδείας.

- **Διακομιστής πόρων** (resource server)
  Είναι ο διακομιστής που φιλοξενεί τους προστατευόμενους πόρους, μπορεί να δέχεται και να ανταποκρίνεται σε αιτήματα των προστατευόμενων πόρων χρησιμοποιώντας επικυρωμένες token αιτήσεις.

- **Διακομιστής άδειας** (authorization server)
  Είναι ένας διακομιστής έκδοσης πρόσβασης στην εφαρμογή πελάτη, μετά την επιτυχή επικύρωση από τον ιδιοκτήτη των πόρων και την απόκτηση της άδειας. Αυτή η άδεια εκφράζεται σε μορφή token και ενός κοινού μυστικού κλειδιού.

Συχνά όμως, ο διακομιστής άδειας μπορεί να είναι ο ίδιος με το διακομιστή πόρων ή μπορεί να είναι μια ξεχωριστή οντότητα. Κι' αυτό γιατί, κάθε μεμονωμένος διακομιστής άδειας μπορεί να εκδώσει διακριτικά πρόσβασης αποδεκτά από πολλαπλούς διακομιστές πόρων. Έτσι, οι ρόλοι είναι δυνατόν να αναφέρονται ως εξής :

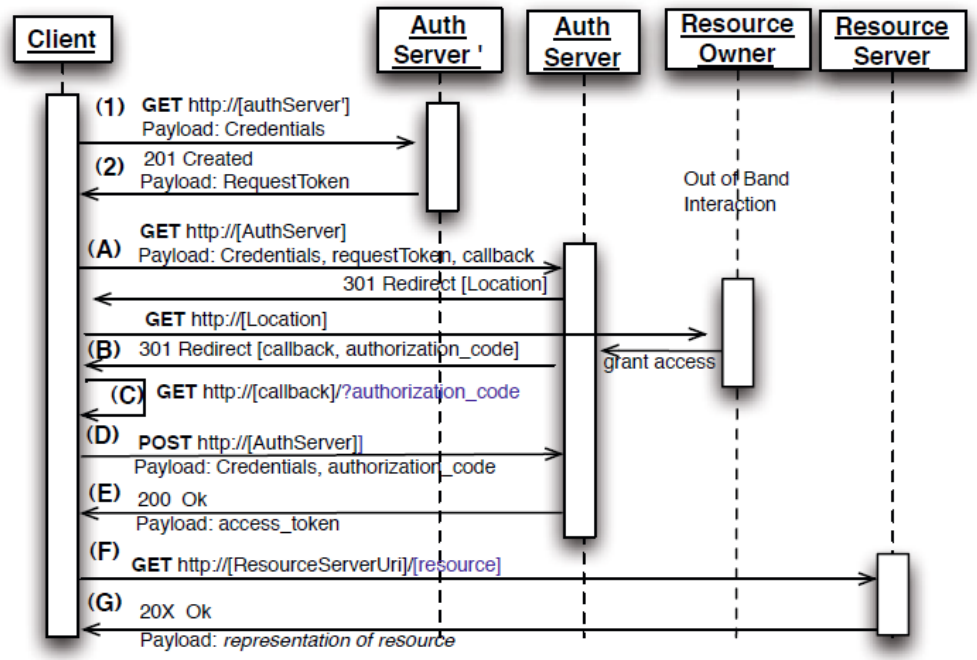
- **Consumer:** client application
- **Service Provider:** resource server
- **User:** resource owner

και τα αντίστοιχα διαπιστευτήρια που χρησιμοποιούνται είναι τα εξής:

- **Consumer Key και Secret Key:** client credentials
- **Request Token και Secret:** temporary credentials
- **Access Token και Secret:** token credentials

4.2.2 Η Αρχιτεκτονική του πρωτοκόλλου OAuth 2.0

Η αρχιτεκτονική του πρωτοκόλλου OAuth 2.0 είναι πολύ απλή και ο ορισμός της βασίζεται στην αφηρημένη ροή πρωτοκόλλου όπως απεικονίζεται στο παρακάτω Σχήμα 4.2, το οποίο περιγράφει την αλληλεπίδραση μεταξύ των παραπάνω ρόλων.



Σχήμα 4.2 Διάγραμμα ακολουθίας βασικής ροής πρωτοκόλλου OAuth 2.0

Πρίν εξηγήσουμε την αρχιτεκτονική του πρωτοκόλλου, αξίζει να σημειώσουμε ότι, στο OAuth 2.0, οι εφαρμογές πελάτες προβαίνουν μόνοι τους στη διαδικασία εγγραφής τους στον διακομιστή άδειας πρίν κάνουν χρήση των υπηρεσιών που αυτός προσφέρει. Έτσι, κατά τη διάρκεια της εγγραφής, εκδίδεται στην εφαρμογή πελάτης ένα δημόσιο αναγνωριστικό κλειδί, το οποίο στη συνέχεια χρησιμοποιεί για να αλληλεπιδράσει με τον διακομιστή άδειας.

Σύμφωνα λοιπόν με το παραπάνω σχήμα, η εφαρμογή πελάτης αρχικά αποκτά κάποια διαπιστευτήρια (consumer key, consumer secret key) και ζητά άδεια από τον ιδιοκτήτη των πόρων (resource owner) ή μέσω του διακομιστή άδειας (authorization server) ως ενδιάμεσος για να πετύχει πρόσβαση σε πόρους **(A).** Ο διακομιστής ελέγχει την ταυτότητα του ιδιοκτήτη των πόρων, μέσω μιας φόρμας δεδομένων. Αυτή η επικοινωνία λαμβάνει χώρα out-of-band (απευθείας διασύνδεση του client με τον διακομιστή) μεταξύ του ιδιοκτήτη των πόρων και του χρήστη για καταχώρηση των στοιχείων του χρήστη σε αυτόν. Μόλις ο ιδιοκτήτης των πόρων δώσει πρόσβαση στους απαιτούμενους πόρους, ο διακομιστής άδειας ανακατευθύνει το χρήστη στην αρχική διεύθυνση μαζί με ένα κωδικό έγκρισης (authorization code) που παρέχεται στην

εφαρμογή πελάτη **(B,C).** Ο κωδικός αυτός χρησιμοποιείται για να ζητήσει ένα διακριτικό πρόσβασης - access token **(D)** από το διακομιστή ελέγχου άδειας και αφού εγκριθεί και επικυρωθεί ο κωδικός, χορηγείται το access token στην εφαρμογή πελάτη **(E).** Στη συνέχεια, η εφαρμογή πελάτης χρησιμοποιεί αυτό το access token για να ζητήσει την πρόσβαση σε προστατευόμενους πόρους από τον διακομιστή πόρων στον οποίο βρίσκονται αποθηκευμένοι **(F).** Ο διακομιστής πόρων επαληθεύει το access token που έλαβε και αν είναι έγκυρο εξυπηρετεί το αίτημα **(G)** [26, 27].

4.3 Σχεδιασμός Μεθοδολογίας για πρόσβαση στην RESTful υπηρεσία

Η μεθοδολογία που θα χρησιμοποιηθεί περιλαμβάνει τους τρόπους με τους οποίους περιγράφονται οι υπηρεσίες αλλά και πώς μια διαδικτυακή πλατφόρμα μπορεί να τις αναγνωρίσει [23]. Σύμφωνα λοιπόν, με τις αρχές της προδιαγραφής REST τα βήματα που πρέπει να εφαρμοστούν για την προτεινόμενη μεθοδολογία σχεδιασμού της RESTful υπηρεσίας είναι τα παρακάτω:

Βήμα 1. Σχεδιασμός του Μοντέλου Αντικειμένου (Object Model) της MediaWiki και WordPress Restful υπηρεσίας

Βήμα 2. Αντιστοίχηση λειτουργιών της MediaWiki και WordPress Restful υπηρεσίας σε Restful URIs

Βήμα 3. Ενοποίηση των Restful URIs στην πλατφόρμα Moodle

i. Διασύνδεση Moodle RESTful API με MediaWiki RESTful API

ii. Διασύνδεση Moodle RESTful API με WordPress RESTful API

Στη συνέχεια της εργασίας περιγράφουμε αναλυτικά τα βήματα αυτά.

4.3.1 Σχεδιασμός του Μοντέλου Αντικειμένου της MediaWiki και WordPress RESTful υπηρεσίας.

Σε αυτό το βήμα σχεδιάζουμε το μοντέλο αντικειμένου των πόρων της κάθε υπηρεσίας που παίρνουν μέρος. Σχεδιάζουμε δηλαδή, ένα διάγραμμα κλάσεων (class diagram) παρουσιάζοντας

τα κύρια αντικείμενα και τις συσχετίσεις που υπάρχουν ανάμεσα τους αναπαριστώντας τους ως πόρους της MediaWiki υπηρεσίας και WordPress υπηρεσίας αντίστοιχα.

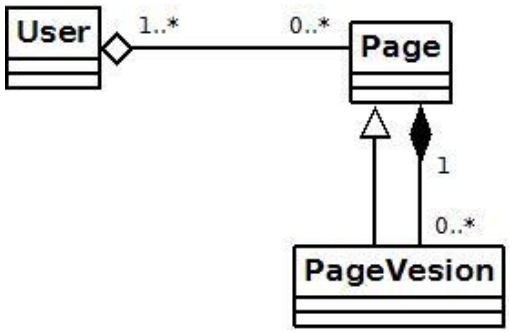
Έτσι, σύμφωνα με το **Σχήμα 4.3**, το αντικείμενο User συσχετίζεται με το αντικείμενο Page εκφράζοντας μια σχέση περιεκτικότητας ασθενούς μορφής και πολλαπλότητας 1 : n για να δηλώσει ότι ένας User μπορεί να συναλλάσσεται με πολλές Pages.

Επίσης, το αντικείμενο Page συσχετίζεται με το αντικείμενο PageVersion δηλώντας τις σχέσεις :

1) **Κληρονομικότητας**, αφού το αντικείμενο PageVersion αποτελεί ειδικότερη κατηγορία του αντικειμένο Page, δηλαδή, το αντικείμενο PageVersion κληρονομεί όλες τις ιδιότητες και χαρακτηριστικά του αντικειμένου Page.

2) **Περιεκτικότητας ισχυρής εξάρτησης,** αφού το αντικείμενο Page έχει την αποκλειστική ευθύνη διαχείρισης του αντικειμένου PageVersion. Με άλλα λόγια, το αντικέμενο PageVersion δεν μπορεί να υπάρχει αν δεν υπάρχει το αντικείμενο Page.

   Η πολλαπλότητα εδώ είναι 1 : n για να δηλώσει ότι μια Page μπορεί να συσχετίζεται με πολλά PageVersion.



Σχήμα 4.3 Διάγραμμα κλάσεων για τη MediaWiki RESTful υπηρεσία

Ενώ, στο **Σχήμα 4.4** ένα αντικείμενο User συσχετίζεται με πολλά Post εκφράζοντας μια σχέση περιεκτικότητας ασθενούς μορφής, που σημαίνει ότι αν πάψει να υπάρχει το αντικείμενο User, τότε το αντικείμενο Post εξακολουθεί να υφίσταται. Η πολλαπλότητα αυτής της σχέσης, είναι 1 : n για να δηλώσει ότι ένας User μπορεί να συσχετίζεται με πολλά Post.

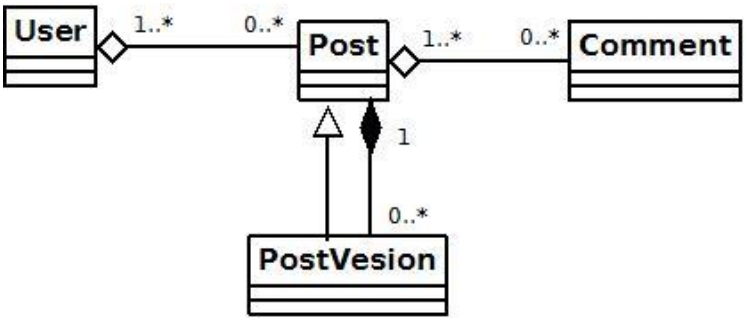
Επίσης, ένα αντικείμενο Post συσχετίζεται με πολλά Comments και πολλά PostVersion. Οι σχέσεις αυτές εκφράζονται μέσω της πολλαπλότητας 1 : n. Η συσχέτιση του αντικειμένου Post με το αντικειμένο Comment, εκφράζει μια σχέση περιεκτικότητας ασθενούς μορφής, δηλώνοντας ότι ένα Comment εξακολουθεί να υφίσταται ακόμα και αν διαγραφεί το αντικείμενο Post.

Τέλος, το αντικείμενο Post συσχετίζεται με το αντικείμενο PostVersion δηλώντας τις σχέσεις :

1) **Κληρονομικότητας**, αφού το αντικείμενο PostVersion αποτελεί ειδικότερη κατηγορία του αντικειμένου Post, δηλαδή, το αντικείμενο PostVersion κληρονομεί όλες τις ιδιότητες και χαρακτηριστικά του αντικειμένου Post.

2) **Περιεκτικότητας ισχυρής εξάρτησης,** αφού το αντικείμενο Post έχει την αποκλειστική ευθύνη διαχείρισης του αντικειμένου PostVersion. Με άλλα λόγια, το αντικέμενο PostVersion δεν μπορεί να υπάρχει αν δεν υπάρχει το αντικείμενο Post.

Η πολλαπλότητα και εδώ είναι 1 : n , δηλώντας έτσι ότι ένα Post μπορεί να συσχετίζεται με πολλά PostVersion.



Σχήμα 4.4 Διάγραμμα κλάσεων για τη WordPress RESTful υπηρεσία

4.3.2 Αντιστοίχιση λειτουργιών της MediaWiki και WordPress RESTful υπηρεσίας σε ReSTful URIs.

Όπως αναφέραμε σε προηγούμενο κεφάλαιο, ένα URI δεν μπορεί να «ζήσει» χωρίς την HTTP μέθοδο που μπορεί να εκτελέσει σε κάθε HTTP αίτημα. Η εφαρμογή των RESTful URIs παρέχει τη δυνατότητα περιγραφής των σχέσεων που απεικονίζονται σε ένα μοντέλο αντικειμένων με απλό τρόπο. Επομένως, αν έχουμε ένα καλά ορισμένο ReSTful URI έχουμε στην ουσία ένα RESTful API. Στην περίπτωση μας, ένα API μπορεί να περιγράψει τις σχέσεις Page-User, User-Post-Comment, User-Post, έτσι ώστε κάθε λειτουργία επίκλησης να μπορεί εύκολα να παραχθεί από αυτό το μοντέλο ως μια κλήση της HTTP μεθόδου (GET, PUT, POST, DELETE). Για παράδειγμα, η λειτουργία getPage(page_id) ανακτά τη σελίδα με το συγκεκριμένο id, και το αντίστοιχο RESTful URI που καλείται είναι /pages/{page_id}. Έτσι, όλοι οι πόροι (pages, users, posts, comments) που χρησιμοποιούνται σε καθεμιά από τις εμπλεκόμενες εφαρμογές υλοποιούνται με ανάλογο τρόπο όπως φαίνεται στον παρακάτω πίνακα 1 και πίνακα 2.

URI	HTTP METHOD	DESCRIPTION	OPERATION
/pages	GET	Retrieve the list of Pages	getPages()
/pages/{page_id}	GET	Retrieve a Page	getPage(page_id)
/pages/new	POST	Create a Page	createPage()
/pages/{page_id}	PUT	Update a Page	updatePage(page_id)
/user/user_id/pages	GET	Retrieve the Pages of a User	getPages(user_id)
/users/{user_id}	GET	Retrieve a User	getUser(user_id)
/users/new	POST	Create a User	createUser()
/users/{user_id}	PUT	Update a user	updateUser(user_id)
/users/{user_id}	DELETE	Delete a User	deleteUser(user_id)

Πίνακας 1. MediaWiki RESTful URIs

URI	HTTP METHOD	DESCRIPTION	OPERATION
/posts	GET	Retrieve the list of Pages	getPosts()
/ posts/{posts_id}	GET	Retrieve a Page	getPost(post_id)
/ posts/new	POST	Create a Page	createPost()
/ posts/{posts _id}	PUT	Update a Page	updatePost(post_id)
/ posts/{posts_id}	DELETE	Delete a Post	deletePost(post_id)
/user/{user_id}/pages	GET	Retrieve the Pages of a User	getUsers()
/users/{user_id}	GET	Retrieve a User	getUser(user_id)
/users/new	POST	Create a User	createUser()
/users/{user_id}	PUT	Update a user	updateUser(user_id)
/users/{user_id}	DELETE	Delete a User	deleteUser(user_id)
/comments	GET	Retrieve the list of Comments	getComments()
/comment/{comment_id}	GET	Retrieve a Comment	getComment(comment_id)
/comment	POST	Create a Comment	postComment()
/comment/{comment_id}	PUT	Update a Comment	putComment(comment_id)
/comment/{comment_id}	DELETE	Delete a Comment	deleteComment(comment_id)

Πίνακας 2. WordPress RESTful URIs

Για την περιγραφή των σχέσεων που απεικονίζονται στο μοντέλο αντικειμένων του προηγούμενου βήματος, δημιουργούμε τους παραπάνω δύο πίνακες, ένα για κάθε RESTful υπηρεσία (MediaWiki RESTful και WordPress RESTful).

Η **πρώτη** στήλη κάθε πίνακα, αναπαριστά τα URIs, δηλαδή τους πόρους που αντιστοιχούν σε κάθεμια από τις καθοριζόμενες URL διευθύνσεις. Στη **δεύτερη** στήλη αναφέρονται οι HTTP μέθοδοι (CRUD) που καλούνται σε καθένα από τα καθοριζόμενα URIs κατά τη διάρκεια των HTTP αιτημάτων. Δηλαδή, για να δημιουργήσουμε ένα νέο πόρο που προσδιορίζεται από ένα συγκεκριμένο URI χρησιμοποιούμε τη μέθοδο POST, για να ανακτήσουμε ένα πόρο χρησιμοποιούμε τη μέθοδο GET, για να αλλάξουμε ή ενημερώσουμε την κατάσταση ενός πόρου χρησιμοποιούμε τη μέθοδο PUT, ενώ για να διαγράψουμε ένα πόρο χρησιμοποιούμε τη μέθοδο DELETE. Η **τρίτη** στήλη περιγράφει το είδος της λειτουργίας που εκτελείται για κάθε ένα από τα καθοριζόμενα URIs. Η **τέταρτη** στήλη, αναφέρει τις λειτουργίες επίκλησης που εκτελούνται για κάθε συγκεκριμένο URI.

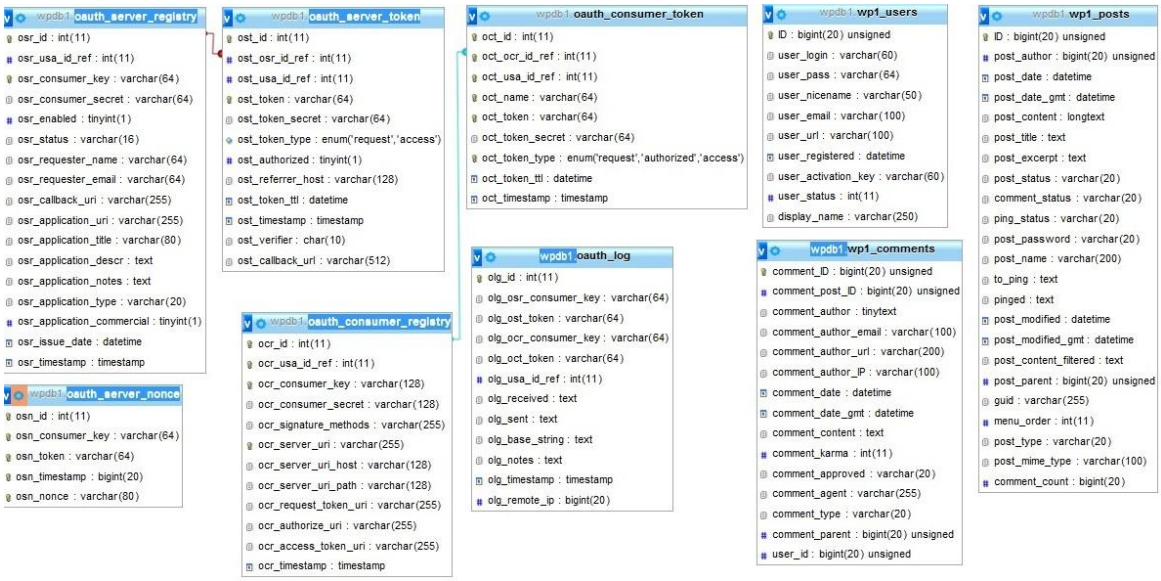
4.3.3 Ενοποίηση των ReSTful URIs στη πλατφόρμα Moodle

Σε αυτό το βήμα, γίνεται η ενοποίηση της RESTful υπηρεσίας που σχεδιάσαμε στα προηγούμενα βήματα για την διαδικτυακή πλατφόρμα MediaWiki και WordPress στη πλατφόρμα Moodle. Με αυτό τον τρόπο θα διαπιστώσουμε πως οι προτεινόμενες RESTful υπηρεσίες μπορούν να έχουν πρόσβαση σε δραστηριότητες μιας μαθησιακής πλατφόρμας όπως είναι το Moodle. Επίσης, παρουσιάζονται οι πίνακες που προστέθηκαν στα αντίστοιχα σχήματα των Βάσεων δεδομένων της κάθε πλατφόρμας χωριστά για την υλοποίηση των παραπάνω υπηρεσιών.

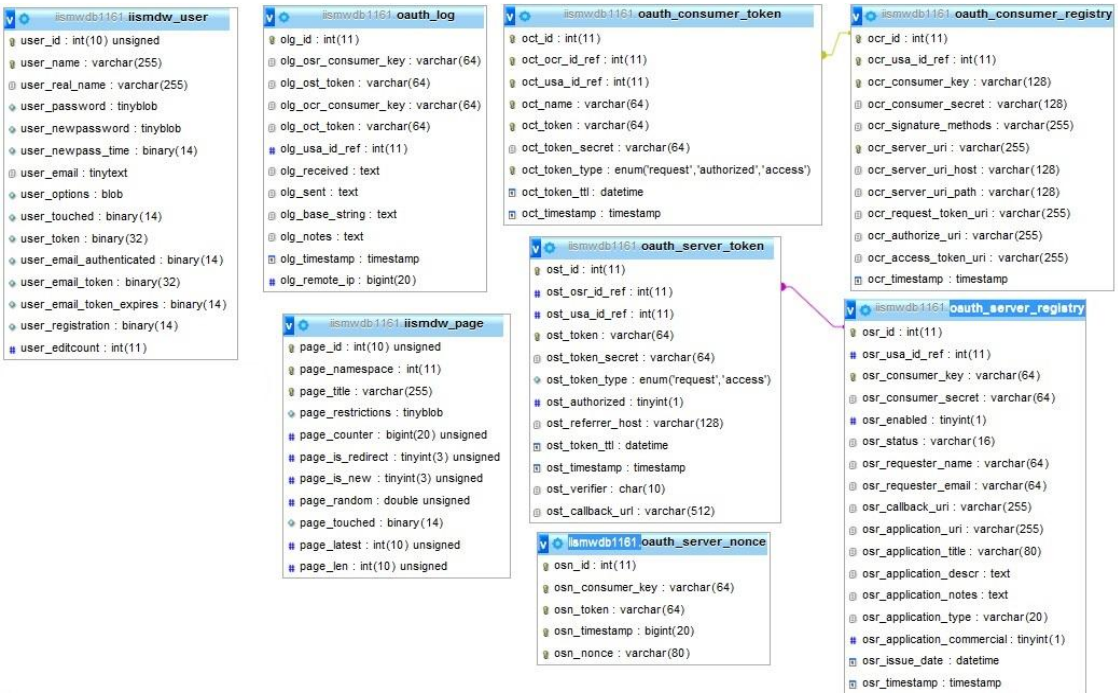
4.3.3.1 Επιπρόσθετοι πίνακες που δημιουργήθηκαν στα σχήματα των Β.Δ των αντίστοιχων πλατφορμών.

- **Πίνακες που προστέθηκαν στη Βάση δεδομένων του WordPress και του MediaWiki.**

Οι επιπρόσθετοι πίνακες που προστέθηκαν στο σχήμα της Β.Δ. του WordPress και του MediaWiki παρουσιάζονται στα παρακάτω σχήματα αντίστοιχα (Σχ. 4.5 και Σχ. 4.6).



Σχήμα 4.5 : Πίνακες που προστέθηκαν στην Β.Δ. του WordPress



Σχήμα 4.6 : Πίνακες που προστέθηκαν στη Β.Δ. του MediaWiki

Όπως παρατηρούμε, οι πίνακες που προστέθηκαν είναι οι ακόλουθοι :

Ο πίνακας "**oauth_server_registry**": Όπου θα αποθηκεύονται τα δεδομένα αιτημάτων πιστοποίησης από τον διακομιστή OAuth . Επίσης, εδώ πρέπει να σημειώσουμε ότι, τα δεδομένα που αποθηκεύονται σε αυτόν τον πίνακα συσχετίζονται και με τα στοιχεία χρηστών (διαχειριστών) που προϋπάρχουν στον πίνακα "Users" της κάθε πλατφόρμας.

Ο πίνακας "**oauth_server_token**": Όπου θα αποθηκεύονται τα δεδομένα για την επαλήθευση των αιτημάτων που έχουν αποσταλεί στον διακομιστή πιστοποίησης OAuth από τον Consumer. Όταν η επαλήθευση αυτή είναι θετική, τότε επιστρέφεται μέσω του διακομιστή πιστοποίησης η ταυτότητα (το user_id) του συσχετιζόμενου χρήστη.

Ο πίνακας "**oauth_consumer_token**": Όπου θα αποθηκεύονται δεδομένα, τα οποία θα χρησιμοποιούνται για την υπογραφή των αιτήσεων που αποστέλλονται από έναν Consumer προς τον διακομιστή. Το κάθε παραγόμενο κλειδί συσχετίζεται κάθε φορά μονό με ένα χρήστη, καθώς και μόνο ένα μοναδικό κλειδί επιτρέπεται να συμμετέχει στον συνδυασμό επικοινωνίας του χρήστη με τον διακομιστή.

Ο πίνακας "**oauth_consumer_registry**": Όπου θα αποθηκεύονται οι κωδικοί των Consumers που αντλήθηκαν από άλλους διακομιστές. Επίσης, εδώ πρέπει να αναφέρουμε ότι, στο πεδίο "ocr_consumer_key", αποθηκεύεται για κάθε Consumer ένα μυστικό κλειδί το οποίο παράγεται από τον διακομιστή. Επιπρόσθετα δεδομένα που αποθηκεύονται σε αυτόν τον πίνακα είναι η διεύθυνση URI του κάθε διακομιστή, η οποία και συσχετίζεται στον υπολογισμό του μυστικού κλειδιού για κάθε Consumer.

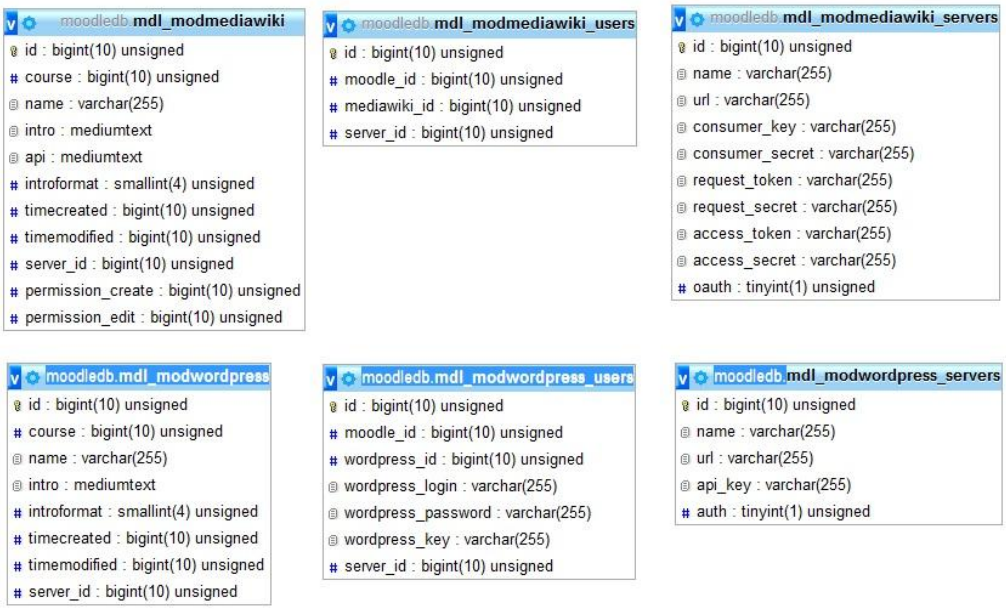
Ο πίνακας "**oauth_consumer_nonce**": Οι εγγραφές που καταχωρούνται σε αυτόν τον πίνακα είναι μοναδικές για κάθε Consumer, συγκεκριμένα μια εγγραφή για κάθε Consumer. Με τον τρόπο αυτό αποτρέπονται πιθανές επιθέσεις στον διακομιστή.

Ο πίνακας "**oauth_log**": Διατηρεί το ιστορικό αιτημάτων OAuth για κάθε Consumer.

Τέλος, οι επιπρόσθετοι πίνακες "**wp1_users**", "**wp1_posts**", "**wp1-comments**" του Σχ.4.5 που αφορούν το WordPress καθώς και οι πίνακες "**iismdw_user**" και "**iismdw_page**" του Σχ.4.6 που αφορούν το MediaWiki, αποτελούν μέρος ενός μεγαλύτερου σχήματος Β.Δ. στην κάθε πλατφόρμα αντίστοιχα. Ο λόγος που τους παρουσιάζουμε είναι διότι, μετά την επιτυχή πιστοποίηση χρηστών που θα επιτυγχάνεται μεταξύ του Moodle με το WordPress και το MediaWiki, τα μηνύματα κειμένου που θα ανταλλάσσονται μεταξύ τους θα αποθηκεύονται σε αυτούς τους πίνακες.

- **Πίνακες που προστέθηκαν στην Βάση δεδομένων του Moodle.**

Οι επιπρόσθετοι πίνακες που προστέθηκαν στο σχήμα της Β.Δ. του Moodle παρουσιάζονται στο παρακάτω σχήμα (Σχ. 4.7).



Σχήμα 4.7 : Πίνακες που προστέθηκαν στην Β.Δ. του Moodle

Για την διασύνδεση του Moodle με το WordPress και το MediaWiki αναπτύχθηκαν δυο διαφορετικά modules, ένα για κάθε πλατφόρμα. Η αρχιτεκτονική σχεδίαση αυτών των modules στηρίχθηκε στο πρότυπο που έχει παρουσιαστεί στην δημοσίευση με τίτλο "ReST-Based Web Access to Learning Design Services" [1].

- Για την διασύνδεση του **Moodle** με το **WordPress** οι πίνακες που προστέθηκαν είναι :

Ο πίνακας "**mdl_modwordpress**": Όπου αποθηκεύονται τα δεδομένα που παράγονται κατά την διάρκεια ενεργοποίησης του WordPress module στο Moodle.

Ο πίνακας "**mdl_modwordpress_users**": Όπου αποθηκεύονται δεδομένα που αφορούν την συσχέτιση χρηστών του Moodle και του WordPress.

Ο πίνακας "**mdl_modwordpress_servers**": Όπου αποθηκεύονται δεδομένα που αφορούν τους διακομιστές REST API του module WordPress στο Moodle.

- Για την διασύνδεση του **Moodle** με το **MediaWiki** οι πίνακες που προστέθηκαν είναι :

Ο πίνακας "**mdl_modmediawiki**": Όπου αποθηκεύονται τα δεδομένα που παράγονται κατά την διάρκεια ενεργοποίησης του Mediawiki module στο Moodle.

Ο πίνακας "**mdl_modmediawiki_users**": Όπου αποθηκεύονται δεδομένα που αφορούν την συσχέτιση χρηστών του Moodle και του Mediawiki.

Ο πίνακας "**mdl_modmediawiki_servers**": Όπου αποθηκεύονται δεδομένα που αφορούν τους διακομιστές REST API του module Mediawiki στο Moodle.

4.3.3.2 Διασύνδεση Moodle RESTful API με MediaWiki RESTful API

Η διαδικτυακή πλατφόρμα MediaWiki, που παίζει το ρόλο του Service Provider χρησιμοποιεί το πρωτόκολλο OAuth 2.0 για την πιστοποίηση των χρηστών και την κωδικοποίηση των μηνυμάτων που ανταλλάσσονται με την πλατφόρμα Moodle. Ένας client του Moodle θα παίζει το ρόλο του Consumer της εφαρμογής.

Έτσι, ένας **Consumer** του **Moodle**, κάνει ένα αίτημα στον **Service Provider** μέσω το ακόλουθου URL: http://localhost/mediawiki/index.php/Special:Thesis/

Για να επιτευχθεί η επικοινωνία μεταξύ τους θα πρέπει πρώτα να γίνει πιστοποίηση. Έτσι, οι διαδικασίες που πρέπει να εκτελεστούν στο MediaWiki Restful API είναι οι ακόλουθες :

1) Αρχικά, δημιουργείται ένας νέος Consumer του Moodle και σε αυτόν προστίθενται η διεύθυνση URL του Service Provider με τον οποίο θέλει να επικοινωνήσει.
2) Στη συνέχεια, ο Consumer αποστέλλει στον Service Provider το αντίστοιχο αίτημα διασύνδεσης.
3) Ο Service Provider λαμβάνει το αίτημα μέσω της διεύθυνσης http://localhost/mediawiki/index.php/Special:Thesis/request-token, το οποίο περιέχει δύο αναγνωριστικά κλειδιά (request-tokens), το consumer_key και το consumer_secret που θα τον βοηθήσουν να αναγνωρίσει τον Consumer.
4) Ο Service Provider αποστέλλει ένα μήνυμα απόκρισης στον Consumer με δύο αναγνωριστικά κλειδιά, το oauth_token και oauth_token_secret για την πιστοποίηση του Consumer μαζί με οδηγίες για την ανακατεύθυνση του Consumer στην αρχική URL διεύθυνση για την επίτευξη της διασύνδεσης του με τον Service Provider.
5) Έχοντας, ο Consumer τα τέσσερα αυτά κλειδιά, τα χρησιμοποιεί για να επικοινωνήσει με τον Service Provider, στέλνοντας το αντίστοιχο μήνυμα αλληλεπίδρασης στον Service Provider.
6) Ο Service Provider αποστέλλει ένα μήνυμα απόκρισης που περιέχει δυο νέα αναγνωριστικά κλειδιά (access tokens), το oauth_token και oauth_token_secret, μέσω της διεύθυνσης http://localhost/mediawiki/index.php/Special:Thesis/access-token, τα οποία επιβεβαιώνουν την πρόσβαση του Consumer στους πόρους του Service Provider.

Στη συνέχεια, εφόσον έχει γίνει η πιστοποίηση του Consumer με τα αντίστοιχα αναγνωριστικά κλειδιά (consumer key, consumer secret, oauth token και oauth token secret), μπορούμε να έχουμε πρόσβαση στο MediaWiki API μέσω Restful για να διαχειριστούμε τις σελίδες και τους

χρήστες της εφαρμογής. Πιο συγκεκριμένα, ακολουθεί αναλυτική περιγραφή για καθένα από αυτά (χρήστες και σελίδες).

➢ Διαχείριση σελίδων μέσω MediaWiki API

Εμφάνιση σελίδων χρησιμοποιώντας το MediaWiki API

Ο εγγεγραμμένος χρήστης της πλατφόρμας MediaWiki, μπορεί να προβάλλει μέσω της διεύθυνσης http://localhost/mediawiki/index.php/Special:Thesis/pages, τα δεδομένα (κωδικός, τίτλος, κείμενο) που αφορούν τις σελίδες που έχουν δημιουργήσει οι πιστοποιημένοι χρήστες του Moodle. Η εμφάνιση αυτών των σελίδων επιτυγχάνεται καλώντας τη συνάρτηση getPages(). Η συνάρτηση αυτή, αντλεί όλα τα παραπάνω δεδομένα από το xml αρχείο του Moodle Restful API και τα εμφανίζει στη πλατφόρμα του MediaWiki.

```
function  getPages(){   // κλήση συνάρτησης
    $res = select('page',array('page_id', 'page_title')); //επιλογή πεδίων σελίδας
    $dbr = wfGetDB( DB_SLAVE );  //ανάκτηση δεδομένων από Β.Δ
    $result = array();  //δημιουργία πίνακα για τα δεδομένα
    while ( $row = $dbr->fetchObject( $res ) )
    { //δημιουργία αντικειμένου Article με βάση το title της σελίδας
  $art = new Article(Title::newFromText($row->page_title));
   //καταχώρηση στο πίνακα
  array_push($result, array($row->page_id, $row->page_title, $art->getComment()));
    }
    $dbr->freeResult( $res );
    return $result;  //επιστροφή δεδομένων σελίδων
    return $res; }
```

Εμφάνιση μιας σελίδας χρησιμοποιώντας το MediaWiki API

Μέσω της διεύθυνσης http://localhost/mediawiki/index.php/Special:Thesis/pages/{page-id}, ο εγγεγραμμένος χρήστης της πλατφόρμας, μπορεί να προβάλει μια συγκεκριμένη σελίδα που έχουν δημιουργήσει οι χρήστες του Moodle. Η εμφάνιση αυτής της σελίδας επιτυγχάνεται καλώντας τη συνάρτηση getPage(). Η συνάρτηση αυτή, αντλεί μια σελίδα από το Moodle και στη συνέχεια εμφανίζεται τόσο στο Moodle, όσο και στο MediaWiki, επιστρέφοντας το τίτλο, το κείμενο και την περίληψη της συγκεκριμένης σελίδας.

```
function getPage($page){
    $res = select('page','page_title','page_id = '.$page); //επιλογή δεδομένων
    $dbr = wfGetDB( DB_SLAVE ); //ανάκτηση δεδομένων από Β.Δ
    $row = $dbr->fetchObject( $res ); //επιστροφή αντικειμένου σελίδας
    return $row->page_title;  //επιστροφή τίτλου σελίδας
}
```

Δημιουργία σελίδας χρησιμοποιώντας το MediaWiki API

Η διεύθυνση http://localhost/mediawiki/index.php/Special:Thesis/pages/new, παρέχει σε ένα εγγεγραμμένο χρήστη της πλατφόρμας, τη δυνατότητα δημιουργίας μιας σελίδας στο MediaWiki μέσω του Moodle, η οποία την ίδια χρονική στιγμή δημιουργείται/προβάλλεται και στο MediaWiki. Για την δημιουργία αυτής της σελίδας χρησιμοποιείται η συνάρτηση createPage (). Η συνάρτηση αυτή, αντλεί τα δεδομένα (τίτλο, κείμενο, περίληψη) από το xml αρχείο του Moodle και τα αποθηκεύει στα αντίστοιχα πεδία της βάσης που αφορούν τις σελίδες του MediaWiki έτσι ώστε να μπορέσει να ανακτηθεί και να προβληθεί στην σελίδα του MediaWiki.

```
function createPage($name, $text, $summary){
    // δημιουργια αντικειμένου Title με βάση το name
    $title = Title::newFromText( $name );
    if ( is_null($title) ) {  //ο τίτλος της σελίδας δεν υπάρχει
            throw new Exception("Invalid data title\n", "001");
    }
    //ανάκτηση ArticleID με βάση το name για την καταχώρηση του τίτλου
    $aid = $title->getArticleID( GAID_FOR_UPDATE );
    if ($aid != 0) {
            throw new Exception("Duplicate article '$name'\n", "001" );
    }
    $art = new Article($title); //δημιουργία αντικειμένου για το τίτλο της σελίδας
    //προσθήκη text και summary στο αντικείμενο art για τη νέα σελίδα
    $art->insertNewArticle($text, $summary);
}
```

Ενημέρωση σελίδας χρησιμοποιώντας το MediaWiki API

Ένας εγγεγραμμένος χρήστης της πλατφόμας, για να ενημερώσει τα δεδομένα (τίτλος, πλήρες κείμενο, περίληψη) μιας ήδη καταχωρημένης σελίδας του Moodle και κατ' επέκταση του MediaWiki, αρκεί να επισκεφτούμε την αντίστοιχη σελίδα που θέλουμε σύμφωνα με τη διεύθυνση http://localhost/mediawiki/index.php/Special:Thesis/pages/{page-id}. Για να γίνει αυτό, χρησιμοποιείται η συνάρτηση updatePage(), η οποία θα ενημερώσει τα αντίστοιχα πεδία της βάσης (name, text, summary) της συγκεκριμένης σελίδας στην εφαρμογή του MediaWiki.

```
function updatePage($name, $text, $summary){
    // δημιουργια αντικειμένου Title με βάση το name της σελίδας
    $title = Title::newFromText( $name );
    if ( is_null($title) ) {   // μη εντοπισμός του title
           throw new Exception("Invalid data title\n", "001");
    }
    $aid = $title->getArticleID( GAID_FOR_UPDATE );
```

```
    if ($aid == 0) {
         //ο τίτλος της σελίδας δεν υπάρχει
         throw new Exception("Not exists article '$name'\n", "001");
    }
    $art = new Article($title); //δημιουργία αντικειμένου για το τίτλο σελίδας
    $art->updateArticle( $text, $summary); //ενημέρωση πεδίων της σελίδας
}
```

➢ Διαχείριση χρηστών μέσω MediaWiki API

Εμφάνιση χρηστών χρησιμοποιώντας το MediaWiki API

Ο εγγεγραμμένος χρήστης της πλατφόρμας MediaWiki, χρησιμοποιώντας τη διεύθυνση http://localhost/mediawiki/index.php/Special:Thesis/users, μπορεί να προβάλλει τα δεδομένα (κωδικός χρήστη, όνομα χρήστη, πραγματικό όνομα χρήστη, email χρήστη), που αφορούν τους χρήστες που έχουν δημιουργηθεί μέσω του Moodle. Η προβολή αυτών των στοιχείων επιτυγχάνεται μέσω της συνάρτησης getUsers(). Η συνάρτηση αυτή, αντλεί όλα τα παραπάνω δεδομένα από το xml αρχείο του Moodle Restful API και τα εμφανίζει στην εφαρμογή του MediaWiki.

```
function getUsers(){
    $fields = array('user_id', 'user_name', 'user_real_name', 'user_password',
              'user_email');
    $res = select('user',$fields); //επιλογή δεδομένων του χρήστη
    $dbr = wfGetDB( DB_SLAVE ); //ανάκτηση δεδομένων από Β.Δ
    $result = array();//επιστροφή των πεδίων του πίνακα που περιέχει τα δεδομένα
    while ( $row = $dbr->fetchObject( $res ) ) {
   //καταχώρηση δεδομένων χρήστη στο πίνακα
       array_push($result, array($row->user_id, $row->user_name,
       $row->user_real_name, $row->user_email));
    }
    $dbr->freeResult( $res );
    return $result; // επιστροφή δεδομένων χρηστών
}
```

Εμφάνιση ενός χρήστη χρησιμοποιώντας το MediaWiki API

Μέσω της διεύθυνσης http://localhost/mediawiki/index.php/Special:Thesis/users/{user-id}, ο εγγεγραμμένος χρήστης της πλατφόρμας MediaWiki, μπορεί να προβάλει ένα συγκεκριμένο χρήστη του Moodle στη πλατφόρμα του MediaWiki. Για την επίτευξη αυτή χρησιμοποιείται η συνάρτηση getUser(). Η συνάρτηση αυτή, αντλεί ένα χρήστη από το xml αρχείο του Moodle Restful API και τον εμφανίζει στο MediaWiki, επιστρέφοντας το όνομα χρήστη, το πραγματικό όνομα χρήστη και το αντίστοιχο email του.

```
function getUser($ID){
// δημιουργία αντικειμένου User με βάση το user id
    $user = User::newFromId($ID);
// αναζήτηση του User
    $find = $user->loadFromId();
    if ($find)  // εάν βρεθεί ο User
        return $user;  //επιστροφή του User
    return null; //αλλιώς επιστρέφει null
}
```

Δημιουργία χρήστη χρησιμοποιώντας το MediaWiki API

Μέσω της διεύθυνσης http://localhost/mediawiki/index.php/Special:Thesis/users, ο εγγεγραμμένος χρήστης της πλατφόρμας MediaWiki μπορεί να δημιουργήσει ένα χρήστη μέσω του Moodle, ο οποίος την ίδια χρονική στιγμή προβάλλεται και στο MediaWiki. Η συνάρτηση που χρησιμοποιείται για να επιτευχθεί αυτό είναι η createUser(). Η συνάρτηση αυτή, αντλεί τα δεδομένα (όνομα, κωδικό πρόσβασης, email και πραγματικό όνομα χρήστη) από το xml αρχείο του Moodle Restful API και τα αποθηκεύει στα αντίστοιχα πεδία της βάσης που αφορούν τους χρήστες του MediaWiki έτσι ώστε να μπορέσει να ανακτηθεί και να προβληθεί στην σελίδα του MediaWiki.

```
function createUser($data){  //κλήση συνάρτησης
    global $wgRequest;
    $user = User::newFromName($data['name']);
    if(!$user)
        throw new Exception("Invalid data\n", "001"); // μη έγκυρα στοιχεία
    if ( !$user->getID() ) {
        $user->setPassword($data['password']);
        $user->createNew($user->getName(), array(
                                        "password" => $user->mPassword,
                                        "email" => $data['email'],
                                        "real_name" => $data['realname']));
    } else {
        $user->setPassword($data['password']);
        echo "The password has been changed";
    }
    $user->saveSettings();
}
```

Ενημέρωση δεδομένων ενός χρήστη χρησιμοποιώντας το MediaWiki API

Για να ενημερώσει ο εγγεγραμμένος χρήστης της πλατφόρμας MediaWiki τα δεδομένα (όνομα, κωδικός πρόσβασης, e-mail, πραγματικό όνομα) ενός ήδη εγγεγραμμένου χρήστη του Moodle και κατ' επέκταση του MediaWiki, αρκεί να επισκεφεί τον αντίστοιχο χρήστη, σύμφωνα με τη διεύθυνση http://localhost/mediawiki/index.php/Special:Thesis/users/{user-id} και έπειτα η συνάρτηση updateUser() θα ενημερώσει τα αντίστοιχα πεδία της βάσης (name, password, e-mail, realname) του συγκεκριμένου χρήστη που επιλέξαμε στην εφαρμογή του MediaWiki.

```
function updateUser($ID, $data){
  // δημιουργία αντικειμένου User με βάση το user id
    $user = User::newFromId($ID);
    $user->load(); // εμφάνιση χρήστη
    if($user->mId == 0)
        throw new Exception("Not exists users '$data->name' with id
          '$ID'\n","001");  // δεν υπάρχουν χρήστες με αυτό το id
    foreach($data as $key=>$value){
        if(isset($value)){
            $method = 'set' . ucwords($key);
            call_user_func(array($user, $method), $value);
        }
    }
    $user->saveSettings();
}
```

4.3.3.3 Διασύνδεση Moodle RESTful API με WordPress RESTful API

Η διαδικτυακή εφαρμογή WordPress που παίζει το ρόλο του Service Provider χρησιμοποιεί το πρωτόκολλο OAuth για την πιστοποίηση των χρηστών και την κωδικοποίηση τα μηνυμάτων που ανταλλάσσονται με τον Consumer. Έτσι, ένας Consumer του Moodle, κάνει ένα αίτημα στον Service Provider μέσω το ακόλουθου URL: http://www.iconix.gr/thesis/wp1/api.

Για τη πιστοποίηση και την ασφαλή επικοινωνία μεταξύ τους, ακολουθείται και εδώ η ίδια διαδικασία πιστοποίησης όπως περιγράψαμε προηγουμένως για το MediaWiki Restful API.

Εφόσον λοιπόν, έχει γίνει η πιστοποίηση του Consumer με τα αντίστοιχα αναγνωριστικά κλειδιά (consumer key, consumer secret, oauth token και το oauth token secret), μπορούμε να έχουμε πρόσβαση στο WordPress API μέσω Restful για να διαχειριστούμε τα άρθρα, τα σχόλια και τους χρήστες της εφαρμογής. Παρακάτω, ακολουθεί αναλυτική περιγραφή για καθένα από αυτά.

➢ Διαχείριση άρθρων μέσω WordPress API

Εμφάνιση άρθρων χρησιμοποιώντας το WordPress API

Ο εγγεγραμμένος χρήστης της πλατφόρμας WordPress, χρησιμοποιώντας τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/posts, μπορεί να προβάλλει μια λίστα άρθρων με δεδομένα που αφορούν τα άρθρα που έχουν δημιουργηθεί από διάφορους χρήστες στο Moodle. Η συνάρτηση getPosts() χρησιμποιείται γι αυτό το σκοπό. Η συνάρτηση αυτή, αντλεί όλα τα δεδομένα από το xml αρχείο του Moodle Restful API και τα προβάλλει στη πλατφόρμα του WordPress.

Έτσι, τα δεδομένα εισόδου που απαιτούνται για την εκτέλεση της συνάρτησης για να αντλήσει τα άρθρα από το Moodle στο Wordpress είναι: το consumer key, consumer secret, oauth token και το oauth token secret. Όπως αναφέραμε και παραπάνω τα συγκεκριμένα κλειδιά είναι απαραίτητα για την πιστοποίηση και την ασφαλή διασύνδεση του Moodle με το Wordpress. Το σώμα της συνάρτησης getPosts() παρουσιάζεται παρακάτω:

```
protected function getPosts() {

        global $wpdb;

        return $this->_return($wpdb->get_results
                            ("SELECT * FROM ".$wpdb->posts." WHERE ID > 0 AND
                              post_type LIKE 'post'"));

                             }
```

Η εκτέλεση της συνάρτησης θα επιστρέψει ένα xml αρχείο που περιλαμβάνει δεδομένα για κάθε άρθρο όπως : ο μοναδικός κωδικός του άρθρου (ID), ο τίτλος του άρθρου (post_title), το πλήρες κείμενο του άρθρου (post_content), καθώς και ο τύπος του άρθρου (post_type), δηλαδή αν πρόκειται για μεμονωμένο άρθρο (page) ή άρθρο (post) που ανήκει σε κάποια κατηγορία.

Η δημιουργία του xml αρχείου επιτυγχάνεται ακολουθώντας τα standards που έχουν οριστεί για το WordPress API RestFull και έχει τη παρακάτω μορφή:

```
<posts>
       <post>
       <ID></ID>
       <post_content></post_content>
       <post_title></post_title>
       <guid></guid> // group User id
       <post_type></post_type>
       </post>
 <post>.....</post>
</posts>
```

Εμφάνιση ενός άρθρου χρησιμοποιώντας το WordPress API

Προσπελάζοντας τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/post/{post-id}, ο εγγεγραμμένος χρήστης της πλατφόρμας WordPress, μπορεί να προβάλει ένα άρθρο του Moodle στο WordPress. Η συνάρτηση getPost() χρησιμοποιείται για αυτό το σκοπό. Η συνάρτηση αυτή, αντλεί ένα άρθρο από το xml αρχείο του Moodle Restful API και το εισάγει στη πλατφόρμα του WordPress χρησιμοποιώντας το WordPress Restful API έτσι ώστε να μπορέσει να ανακτηθεί και να προβληθεί.

Το σώμα της συνάρτησης getPost() παρουσιάζεται παρακάτω:

```
protected function getPost($post) {
       //ανάκτηση post
        $post_data = get_post($post);
        if(is_null($post_data)){ εάν είναι null το post δεν υπάρχει
            throw new Exception("Post '$post' does not exists \n","404");
        }
        // ανάκτηση των comments του post
        $post_data->comments = get_approved_comments($post);
         // εμφάνιση των comments του post
        return $this->_return($post_data);
    }
```

Η εκτέλεση της συνάρτησης θα επιστρέψει ένα xml αρχείο που περιλαμβάνει δεδομένα για το συγκεκριμένο άρθρο όπως: ο μοναδικός κωδικός του άρθρου (ID), ο τίτλος του άρθρου (post_title), το πλήρες κείμενο του άρθρου (post_content), ο τύπος του άρθρου (post_type) καθώς και διάφορες πληροφορίες που αφορούν στο σχολιασμό που έχει προστεθεί για το συγκεκριμένο άρθρο. Τέτοιες πληροφορίες είναι ο κωδικός του σχολίου (comment_ID), ο συγγραφέας του σχολίου (comment_author), η ημερομηνία εγγραφής του σχολίου (comment_date) και το πλήρες κείμενο του σχολίου (comment_content).

Η δομή του xml αρχείου είναι η παρακάτω:

```
<posts>
       <post>
       <ID></ID>
       <post_content></post_content>
       <post_title></post_title>
       <guid></guid>
       <post_type></post_type>
              <comments>
                  <post>
                  <comment_ID></comment_ID>
                  <comment_author></comment_author>
```

```
                <comment_date></comment_date>
                <comment_date_gmt></comment_date_gmt>
                <comment_content></comment_content>
                </post>
            </comments>
        </post>
</posts>
```

Δημιουργία άρθρου χρησιμοποιώντας το WordPress API

Η διεύθυνση http://www.iconix.gr/thesis/wp1/api/post, παρέχει σε έναν εγγεγραμμένο χρήστη της πλατφόρμας WordPress, τη δυνατότητα να δημιουργήσει ένα άρθρο στο WordPress μέσω του Moodle. Για τη δημιουργία του άρθρου καλείται η συνάρτηση postPost(). Η συνάρτηση αυτή, αντλεί τα δεδομένα (τίτλο άρθρου και πλήρες κείμενο άρθρου) από το xml αρχείο του Moodle Restful API και τα αποθηκεύει στα αντίστοιχα πεδία της βάσης (post_title και post_content) που αφορούν τα άρθρα του WordPress και ανήκουν στη σελίδα api, έτσι ώστε να είναι δυνατή η ανάκτηση και η προβολή του στην εφαρμογή του WordPress.

Το σώμα της συνάρτησης postPost() παρουσιάζεται παρακάτω:

```
protected function postPost($data) {
        $new_post = array();
        if (!isset($data['post_title']) || !isset($data['post_content'])) {
         throw new Exception("Insufficient parameters","400");
        }
        $new_post['post_title'] = $data['post_title'];
        $new_post['post_content'] = $data['post_content'];
        $new_post['post_status'] = 'publish';
        if (isset($data['post_author'])) {
         $new_post['post_author'] = $data['post_author'];
        } else {
         $new_post['post_author'] = 1;
        }
        if (isset($data['post_type'])) {
         $new_post['post_type'] = $data['post_type'];
        } else {
         $new_post['post_type'] = 'post';
        }
        $new_post_id = wp_insert_post( $new_post );
        if ($new_post_id > 0) {
         return $this->_return(get_post($new_post_id));
```

```
        } else {
            throw new Exception("Post '".$data['id']."' not modified \n","400");
        }
    }
```

Ενημέρωση ενός άρθρου χρησιμοποιώντας το WordPress API

Για να ενημερώσει ένας εγγεγραμμένος χρήστης της πλατφόρμας WordPress, τα δεδομένα (τίτλο άρθρου και πλήρες κείμενο άρθρου) ενός ήδη καταχωρημένου άρθρου στο Moodle και κατ' επέκταση στο WordPress, αρκεί να επισκεφτεί το αντίστοιχο άρθρο σύμφωνα με τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/post/{post-id} και έπειτα θα κληθεί η συνάρτηση putPost() για να ενημερώσει τα αντίστοιχα πεδία της βάσης (post_title και post_content) του συγκεκριμένου άρθρου στην εφαρμογή του WordPress.

Το σώμα της συνάρτησης putPost() παρουσιάζεται παρακάτω:

```
protected function putPost($data) {
        $update_post = array();
        if (!isset($data['id'])) {
         throw new Exception("Post ID needed","400");
        }
        $post_exists = get_post($data['id']);
        if ($post_exists) {
         $update_post['ID'] = $data['id'];
         if (isset($data['post_title'])) {
             $update_post['post_title'] = $data['post_title'];
         }
         if (isset($data['post_content'])) {
             $update_post['post_content'] = $data['post_content'];
         }
         $updated = wp_update_post( $update_post );
         if ($updated == 1) {
             return $this->_return(get_post($data['id']));
         } else {
             throw new Exception("Post '".$data['id']."' not modified \n","400");
         }
        } else {
         throw new Exception("Post '".$data['id']."' not found \n","404");
        }
    }
```

Διαγραφή ενός άρθρου χρησιμοποιώντας το WordPress API

Για τη διαγραφή ενός ήδη καταχωρημένου άρθρου στο Moodle και κατ' επέκταση στο WordPress, ο εγγεγραμμένος χρήστης της πλατφόρμας WordPress, επισκέπτεται το άρθρο της επιλογής του σύμφωνα με τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/post/{post-id} και έπειτα καλείται η συνάρτηση deletePost() για να διαγράψει από τη βάση του WordPress το αντίστοιχο άρθρο σύμφωνα με το id του άρθρου που επέλεξε.

Το σώμα της συνάρτησης deletePost() παρουσιάζεται παρακάτω:

```
protected function deletePost($post) {
        if (wp_delete_post($post)) {
         return array( 'ID' => $post, 'deleted' => true);
        } else {
         throw new Exception("Error deleting post","403");
        }
    }
```

Η εκτέλεση της παραπάνω συνάρτησης θα επιστρέψει το ακόλουθο xml αρχείο:

```
<posts>
       <ID>12</ID>
       <deleted>1</deleted></posts>
```

➢ Διαχείριση σχολίων μέσω WordPress API

Εμφάνιση σχολίων χρησιμοποιώντας το WordPress API

Ο εγγεγραμμένος χρήστης της πλατφόρμας WordPress, χρησιμοποιώντας τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/comments, μπορεί να προβάλλει όλα τα σχόλια που έχουν εισαχθεί για κάθε ένα από τα άρθρα που έχουν δημιουργηθεί στη εφαρμογή του Moodle. Για την προβολή αυτών των σχολίων καλείται η συνάρτηση getComments (). Η συνάρτηση αυτή, αντλεί όλα τα δεδομένα από το xml αρχείο του Moodle Restful API και τα προβάλλει στη πλατφόρμα του WordPress.

Έτσι, τα δεδομένα εισόδου που απαιτούνται για την εκτέλεση της συνάρτησης για να αντλήσει τα σχόλια από το Moodle στο Wordpress είναι : το consumer key, consumer secret, oauth token και το oauth token secret. Το σώμα της συνάρτησης getComments() παρουσιάζεται παρακάτω:

```
protected function getComments() {
        global $wpdb;
        $array = array();
        wpr_set_defaults($_POST, array('post_id' => false, 'comment_id' =>
false));
        if ($_POST['post_id'])
```

```
        $comments = $wpdb->get_results($wpdb->prepare("SELECT * FROM
" . $wpdb->comments . " WHERE comment_post_ID = %d AND comment_approved = 1 ORDER
BY comment_date ASC", $_POST['post_id']));

        elseif ($_POST['comment_id'])

        $comments = $wpdb->get_results($wpdb->prepare("SELECT * FROM
" . $wpdb->comments . " WHERE comment_ID = %d AND comment_approved = 1 ORDER BY
comment_date ASC", $_POST['comment_id']));

        else

        $comments = $wpdb->get_results("SELECT * FROM " . $wpdb->comments . "
WHERE comment_ID > 0 AND comment_approved = 1 ORDER BY comment_date ASC");


        if (count($comments) == 1) {

        return $this->_return($comments[0]);

        }

        foreach ($comments as $comment) {

        $array[] = $comment;

        }

        return $this->_return($array);

    }
```

Η εκτέλεση της συνάρτησης θα επιστρέψει ένα xml αρχείο περιλαμβάνοντας όλα τα σχόλια που έχουν καταχωρηθεί και πιο συγκεκριμένα για κάθε ένα από αυτά πληροφορίες όπως : ο κωδικός του σχολίου (comment_ID), ο κωδικός του άρθρου που ανήκει το σχόλιο (comment_post_ID), ο συγγραφέας του σχολίου (comment_author), το url του συγγραφέα του σχολίου (comment_author_url), η ημερομηνία καταχώρησης του σχολίου (comment_date) και το πλήρες κείμενο του σχολίου (comment_content).

Η δομή του xml αρχείου είναι η παρακάτω:

```
<comments>
       <comment>
       <comment_ID></comment_ID>
       <comment_post_ID></comment_post_ID> <comment_author></comment_author>
       <comment_author_url></comment_author_url>
       <comment_date></comment_date> <comment_date_gmt></comment_date_gmt>
       <comment_content></comment_content>
       </comment>
<comment> … </coment>
…
</comments>
```

Εμφάνιση ενός σχολίου χρησιμοποιώντας το WordPress API

Προσπελάζοντας τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/comment/{comment-id}, ένας εγγεγραμμένος χρήστης της πλατφόρμας WordPress, μπορεί να προβάλλει ένα σχόλιο από το Moodle στο WordPress. Για την επίτευξη αυτού, χρησιμοποιείται η συνάρτηση getComment(). Η συνάρτηση αυτή, αντλεί ένα σχόλιο από το xml αρχείο του Moodle Restful API και το προβάλλει στην εφαρμογή του WordPress χρησιμοποιώντας το WordPress Restful API

Το σώμα της συνάρτησης getComment() παρουσιάζεται παρακάτω:

```
protected function getComment($post) {
        $comment_data = get_comment($post);
        if(is_null($comment_data)){
            throw new Exception("Comment '$post' not found \n","404");
        }
        return $this->_return($comment_data);
    }
```

Η εκτέλεση της συνάρτησης θα επιστρέψει ένα xml αρχείο με πληροφορίες που αφορούν το συγκεκριμένο σχόλιο που έχει επιλεχθεί όπως: ο κωδικός του σχολίου (comment_ID), ο κωδικός του άρθρου που ανήκει το σχόλιο (comment_post_ID), ο συγγραφέας του σχολίου (comment_author), το url του συγγραφέα του σχολίου (comment_author_url), η ημερομηνία καταχώρησης του σχολίου (comment_date) και το πλήρες κείμενο του σχολίου (comment_content).

Η δομή του xml αρχείου είναι η παρακάτω:

```
<comments>
       <comment>
       <comment_ID></comment_ID> <comment_post_ID></comment_post_ID>
       <comment_author></comment_author>
       <comment_author_url />
       <comment_date></comment_date>
       <comment_date_gmt></comment_date_gmt>
       <comment_content></comment_content>
       </comment>
</comments>
```

Δημιουργία ενός σχολίου χρησιμοποιώντας το WordPress API

Η διεύθυνση http://www.iconix.gr/thesis/wp1/api/comment, παρέχει τη δυνατότητα σε ένα εγγεγραμμένο χρήστη της πλατφόρμας WordPress, να δημιουργήσει ένα σχόλιο στο WordPress που αφορά ένα άρθρο του Moodle. Η συνάρτηση που καλείται εδώ, είναι η postComment(). Η

συνάρτηση αυτή, αντλεί τα δεδομένα (πλήρες κείμενο σχολίου, το συγγραφέα του σχολίου, το email του συγγραφέα του σχολίου και το url του συγγραφέα του σχολίου) από το xml αρχείο του Moodle Restful API και τα αποθηκεύει στα αντίστοιχα πεδία της βάσης (comment_content, comment_author, comment_author_email και comment_author_url) που αφορούν στα σχόλια του WordPress και ανήκουν στη σελίδα api, έτσι ώστε να είναι δυνατή η ανάκτηση και η προβολή τους στην εφαρμογή του WordPress.

Το σώμα της συνάρτησης postComment() παρουσιάζεται παρακάτω:

```
protected function postComment($data) {
  // δημιουργία πίνακα για τη καταχώρηση των δεδομένων του comment
    $comment_data = array();
    $comment_data['comment_post_ID'] = $data['id'];
    if (isset($data['comment_author'])) {
     $comment_data['comment_author'] = $data['comment_author'];
    } else {
     $comment_data['comment_author'] = 1;
    }
    if (isset($data['comment_author_email']))
     $comment_data['comment_author_email'] = $data['comment_author_email'];
    if (isset($data['comment_author_url']))
     $comment_data['comment_author_url'] = $data['comment_author_url'];
    $comment_data['comment_content'] = $data['comment_content'];
    $comment_data['comment_author_IP'] = $_SERVER["HTTP_CLIENT_IP"];
    $comment_data['comment_agent'] = $_SERVER['HTTP_USER_AGENT'];
    $comment_data['comment_date'] = date('Y-m-d H:i:s');
    $comment_data['comment_date_gmt'] = date('Y-m-d H:i:s');
    $comment_data['comment_approved'] = 1;
   // εισαγωγή του νέου comment στη βάση και επιστροφή του id του
    $new_comment_id = wp_insert_comment($comment_data);
    if ($new_comment_id > 0) { // επιστροφή του comment
   // εμφάνιση του comment
     return $this->_return(get_comment($new_comment_id));
    } else {  // μήνυμα λάθους κατά την προσθήκη του comment
     return new WP_Error('error', __('Error adding your comment.'));
    }
}
```

Η εκτέλεση της παραπάνω συνάρτησης θα επιστρέψει ένα xml αρχείο περιλαμβάνοντας πληροφορίες που αφορούν το συγκεκριμένο σχόλιο που δημιουργήθηκε. Τέτοιες πληροφορίες είναι: ο κωδικός του σχολίου (comment_ID), ο κωδικός του άρθρου που ανήκει το σχόλιο

(comment_post_ID), ο συγγραφέας του σχολίου (comment_author), το url του συγγραφέα του σχολίου (comment_author_url), η ημερομηνία καταχώρησης του σχολίου (comment_date) και το πλήρες κείμενο του σχολίου (comment_content).

Έτσι, η δομή του xml αρχείου που δημιουργείται είναι η παρακάτω:

```
<comments>
	<comment>
	<comment_ID></comment_ID>
	<comment_post_ID></comment_post_ID>
	<comment_author></comment_author>
	<comment_author_url></comment_author_url>
	<comment_date></comment_date>
	<comment_date_gmt></comment_date_gmt>
	<comment_content></comment_content>
	</comment>
</comments>
```

Ενημέρωση ενός σχολίου χρησιμοποιώντας το WordPress API

Για να ενημερώσει ένας εγγεγραμμένος χρήστης της πλατφόρμας, τα δεδομένα (πλήρες κείμενο σχολίου, το συγγραφέα του σχολίου, το email του συγγραφέα του σχολίου και το url του συγγραφέα του σχολίου) ενός ήδη καταχωρημένου σχολίου στο Moodle και κατ' επέκταση στο WordPress, αρκεί να επισκεφτεί το αντίστοιχο σχόλιο σύμφωνα με τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/comment/{comment-id}. Η συνάρτηση putComment() θα κληθεί τότε, για να ενημερώσει τα αντίστοιχα πεδία της βάσης (comment_content, comment_author, comment_author_email και comment_author_url) που αφορούν στο συγκεκριμένου σχολίο που επιλέχθηκε στην εφαρμογή του WordPress.

Το σώμα της συνάρτησης putComment() παρουσιάζεται παρακάτω:

```
protected function putComment($data) {
	 $comment_data = array(); //δημιουργία πίνακα για τα comment
	if (isset($data['id']))
	 $comment_data['comment_ID'] = $data['id'];
	if (isset($data['comment_author']))
	 $comment_data['comment_author'] = $data['comment_author'];
	if (isset($data['comment_author_email']))
	 $comment_data['comment_author_email'] = $data['comment_author_email'];
	if (isset($data['comment_author_url']))
	 $comment_data['comment_author_url'] = $data['comment_author_url'];
	if (isset($data['comment_content']))
```

```
        $comment_data['comment_content'] = $data['comment_content'];
        $comment_data['comment_author_IP'] = $_SERVER["HTTP_CLIENT_IP"];
        $comment_data['comment_agent'] = $_SERVER['HTTP_USER_AGENT'];
        $comment_data['comment_date'] = date('Y-m-d H:i:s');
        $comment_data['comment_date_gmt'] = date('Y-m-d H:i:s');
        $comment_data['comment_approved'] = 1;
       // ανάκτηση δεδομένων για ένα comment
        $comment_exists = get_comment($data['id']);
        if ($comment_exists) { //ενημέρωση δεδομένων για ένα comment
        $updated = wp_update_comment($comment_data);
        if ($updated == 1) {
            return $this->_return(get_comment($data['id']));
        } else { // μήνυμα ότι το comment δεν τροποποιήθηκε
          throw new Exception("Comment '".$data['id']."' not modified \n","400");
        }
        } else { // μήνυμα ότι το comment δεν βρέθηκε
        throw new Exception("Comment '".$data['id']."' not found \n","404");
        }
    }
```

Διαγραφή ενός σχολίου χρησιμοποιώντας το WordPress API

Για να διαγράψει ένας εγγεγραμμένος χρήστης της πλατφόρμας WordPress, ένα ήδη καταχωρημένο σχόλιο ενός άρθρου που έχει δημιουργηθεί μέσω του Moodle στο WordPress, αρκεί να επισκεφτεί το αντίστοιχο σχόλιο που επιθυμεί σύμφωνα με τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/comment/{comment-id}. Η συνάρτηση deleteComment() θα κληθεί για να διαγράψει από τη βάση του WordPress το αντίστοιχο σχόλιο σύμφωνα με το ID που επιλέξαμε.

Το σώμα της συνάρτησης deletePost() παρουσιάζεται παρακάτω:

```
protected function deleteComment($comment_id) {
    //ανάκτηση comment
    $comment_exists = get_comment($comment_id);
        if ($comment_exists) { //εάν υπάρχει το id του comment
        if (wp_delete_comment($comment_id)) {
            //διαγραφή comment
            return array('ID' => $comment_id, 'deleted' => true);
        } else { // μήνυμα ότι το comment δεν διαγράφηκε
            throw new Exception("Comment '".$data['id']."' not deleted \n","400");
```

```
        }
        } else { // μήνυμα ότι το comment δεν βρέθηκε
        throw new Exception("Comment '".$comment_id."' not found \n","404");
        }
    }
```

Η εκτέλεση της παραπάνω συνάρτησης θα επιστρέψει το ακόλουθο xml αρχείο:

```
<comment>
       <ID></ID>
       <deleted>1</deleted>
</comment>
```

➢ Διαχείριση χρηστών μέσω WordPress API

Εμφάνιση χρηστών χρησιμοποιώντας το WordPress API

Μέσω της διεύθυνσης http://www.iconix.gr/thesis/wp1/api/users, ο εγγεγραμμένος χρήστης της πλατφόρμας WordPress, μπορεί να εμφανίσει πληροφορίες που αφορούν πιστοποιημένους χρήστες που έχουν δημιουργηθεί στο Moodle. Για την προβολή αυτών των πληροφορίων καλείται η συνάρτηση getUsers(). Η συνάρτηση αυτή, αντλεί όλα τα παραπάνω δεδομένα από το xml αρχείο του Moodle Restful API και τα προβάλλει στην εφαρμογή του WordPress.

Το σώμα της παραπάνω συνάρτησης getUsers() παρουσιάζεται παρακάτω:

```
protected function getUsers() {
        global $wpdb;
        $array = array();  //δημιουργία πίνακα
  //ανάκτηση χρηστών από τη βάση
  $users = $wpdb->get_results("SELECT ID FROM " . $wpdb->users . " WHERE ID > 0");
        if (count($users) == 1) {
        return $this->_return($this->getUser($users[0]->ID));
        }
        foreach ($users as $user) {
        $array[] = $this->getUser($user->ID);
        } //επιστροφή του πίνακα χρηστών
        return $this->_return($array);
    }
```

Μετά την εκτέλεση της παραπάνω συνάρτησης, θα δημιουργηθεί ένα xml αρχείο χρηστών περιλαμβάνοντας τις απαραίτητες πληροφορίες που περιγράφουν τους πιστοποιημένους

χρήστες της εφαρμογής. Τέτοιες πληροφορίες είναι: ο μοναδικός κωδικός του χρήστη (ID) και ο κωδικός πρόσβασης χρήστη (user_login). Άλλες πληροφορίες όπως, user_email , display_name, first_name, last_name θεωρούνται προαιρετικά.

Έτσι, η δομή του xml αρχείου που δημιουργείται είναι η παρακάτω:

```
<users>
       <user>
       <ID></ID>
       <user_login></user_login>
       <user_email></user_email>
       <display_name></display_name>
       <first_name /><last_name />
       </user>
<user> … </user>
…
</users>
```

Εμφάνιση ενός χρήστη χρησιμοποιώντας το WordPress API

Η διεύθυνση http://www.iconix.gr/thesis/wp1/api/user/{user-id}, παρέχει σε ένα εγγεγραμμένο χρήστη της πλατφόρμας WordPress, να εμφανίσει τις απαραίτητες πληροφορίες που αφορούν ένα συγκεκριμένο πιστοποιημένο χρήστη που έχει δημιουργηθεί στο Moodle. Η συνάρτηση getUser() καλείται γι αυτό το σκοπό. Η συνάρτηση αυτή, αντλεί όλα τα παραπάνω δεδομένα από το xml αρχείο του Moodle Restful API και τα προβάλλει στην εφαρμογή του WordPress.

Το σώμα της παραπάνω συνάρτησης getUser() παρουσιάζεται παρακάτω:

```
protected function getUser($user) {
        return $this->_return(get_userdata($user));
    }
```

Έτσι, η δομή του xml αρχείου που δημιουργείται είναι η παρακάτω:

```
<users>
       <user>
       <ID></ID>
       <user_login></user_login>
       <user_email></user_email>
       <display_name></display_name>
       <first_name /><last_name />
       </user>
</users>
```

Όπως παρατηρούμε, μετά την εκτέλεση της παραπάνω συνάρτησης, θα δημιουργηθεί ένα xml αρχείο περιλαμβάνοντας τις απαραίτητες πληροφορίες που περιγράφουν ένα συγκεκριμένο πιστοποιημένο χρήστη της εφαρμογής. Τέτοιες πληροφορίες είναι: ο μοναδικός κωδικός του χρήστη (ID) και ο κωδικός πρόσβασης χρήστη (user_login). Άλλες πληροφορίες όπως, user_email , display_name, first_name, last_name θεωρούνται προαιρετικές.

Δημιουργία χρήστη χρησιμοποιώντας το WordPress API

Ένας εγγεγραμμένος χρήστης της πλατφόρμας WordPress, προσπελάζοντας τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/user, μπορεί να δημιουργήσει ένα χρήστη στο Moodle, μέσω του WordPress. Για να επιτευχθεί αυτό, καλείται η συνάρτηση postUser(). Η συνάρτηση αυτή, αντλεί τα απαραίτητα πεδία user_login και user_password, από το xml αρχείο του Moodle Restful API και τα αποθηκεύει στα αντίστοιχα πεδία της βάσης που αφορούν τους χρήστες του WordPress και ανήκουν στη σελίδα api, έτσι ώστε να μπορέσει να ανακτηθεί και να προβληθεί στην εφαρμογή του WordPress.

Το σώμα της παραπάνω συνάρτησης postUser() παρουσιάζεται παρακάτω:

```
protected function postUser($data) {
        $user_data = array(); //δημιουργία πίνακα user_data
        error_log(">>>>> user_login >>>+++>>>> ".$data['user_login']);
        error_log(">>>>> user_password >>>+++>>>> ".$data['user_password']);

        if (isset($data['user_login']) && isset($data['user_password']) &&
isset($data['user_email'])) {
        $user_data['user_login'] = $data['user_login'];
        $user_data['user_password'] = $data['user_password'];
        $user_data['user_email'] = $data['user_email'];
        if (isset($data['user_url'])) {
            $user_data['user_url'] = $data['user_url'];
        }
        if (isset($data['user_nicename'])) {
            $user_data['user_nicename'] = $data['user_nicename'];
        }
        if (isset($data['display_name'])) {
            $user_data['display_name'] = $data['display_name'];
        }
        if (isset($data['nickname'])) {
            $user_data['nickname'] = $data['nickname'];
        }
        if (isset($data['first_name'])) {
```

```
            $user_data['first_name'] = $data['first_name'];
        }
        if (isset($data['role'])) {
            $user_data['role'] = $data['role'];
        } else {
            $user_data['role'] = get_option('default_role');
        }
         //προσθήκη χρήστη στο πίνακα users και επιστροφή του id
        $new_user_id = wp_insert_user($user_data);
        }
        if (is_wp_error($new_user_id)) {
           error_log($new_user_id->get_error_message());
        return new WP_Error('error', __('Error adding new user.'));
        } else {
        return $this->_return(get_userdata($new_user_id));
        }
    }
```

Έτσι, η δομή του xml αρχείου που δημιουργείται είναι η παρακάτω:

```
<users>
       <user>
       <ID></ID>
       <user_login></user_login>
       <user_email></user_email>
       <display_name></display_name>
       <first_name /><last_name />
       </user>
</users>
```

Ενημέρωση δεδομένων χρήστη χρησιμοποιώντας το WordPress API

Εάν ένας εγγεγραμμένος χρήστης της πλατφόρμας WordPress, επιθυμεί να ενημερώσει τα δεδομένα ενός ήδη καταχωρημένου χρήστη στο Moodle και κατ' επέκταση στο WordPress, αρκεί να επισκεφτεί το αντίστοιχο χρήστη της επιλογής του, σύμφωνα με τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/user/{user-id}, και έπειτα η συνάρτηση putUser() θα κληθεί για να ενημερώσει τα αντίστοιχα πεδία της βάσης (user_login, user_password, user_email, user_url, user_nicename, display_name, nickname, first_name, role) που αφορούν τον επιλεγμένο χρήστη στην εφαρμογή του WordPress.

Έτσι, το σώμα της συνάρτησης putUser() παρουσιάζεται παρακάτω:

```
protected function putUser($data) {
      $user_data = array();
      $user_data['ID'] = $data['id'];

      if (isset($data['user_login'])) {
       $user_data['user_login'] = $data['user_login'];
      }
      if (isset($data['user_password'])) {
       $user_data['user_password'] = $data['user_password'];
      }
      if (isset($data['user_email'])) {
       $user_data['user_email'] = $data['user_email'];
      }
      if (isset($data['user_url'])) {
       $user_data['user_url'] = $data['user_url'];
      }
      if (isset($data['user_nicename'])) {
       $user_data['user_nicename'] = $data['user_nicename'];
      }
      if (isset($data['display_name'])) {
       $user_data['display_name'] = $data['display_name'];
      }
      if (isset($data['nickname'])) {
       $user_data['nickname'] = $data['nickname'];
      }
      if (isset($data['first_name'])) {
       $user_data['first_name'] = $data['first_name'];
      }
      if (isset($data['role'])) {
       $user_data['role'] = $data['role'];
      } else {
       $user_data['role'] = get_option('default_role');
      }
      $updated_user_id = wp_update_user($user_data);

      return $this->_return(get_userdata($updated_user_id));
  }
```

Διαγραφή ενός χρήστη χρησιμοποιώντας το WordPress API

Για να διαγράψει ένας εγγεγραμμένος χρήστης της πλατφόρμας WordPress, ένα ήδη καταχωρημένο χρήστη του Moodle και κατ' επέκταση του WordPress, αρκεί να προσπελάσει τη διεύθυνση http://www.iconix.gr/thesis/wp1/api/user/{user-id}, που αφορά ένα συγκεκριμένο χρήστη. Η συνάρτηση deleteUser() θα κληθεί για να διαγράψει τον αντίστοιχο χρήστη από τη βάση του WordPress σύμφωνα με το ID που επιλέχθηκε.

Το σώμα της συνάρτησης deleteUser() παρουσιάζεται παρακάτω:

```
protected function deleteUser($user_id) {
        if (mysql_query("DELETE FROM wp_users WHERE id='".$user_id."'")) {
         return array( 'ID' => $user_id, 'deleted' => true);
        } else {
         return new WP_Error('error', __('Error deleting user.'));
        }
    }
```

Έτσι, η δομή του xml αρχείου που δημιουργείται είναι η παρακάτω:

```
<user>
<ID></ID>
<deleted>1</deleted>
</user>
```

Κεφάλαιο 5
Σενάρια διασύνδεσης Moodle με WordPress και MediaWiki

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τα σενάρια που ακολουθούνται για την επιτυχή διασύνδεση της πλατφόρμας του Moodle με τη πλατφόρμα WordPress αλλά και της πλατφόρμας Moodle με τη πλατφόρμα MediaWiki για την επιτυχή ανταλλαγή δεδομένων μεταξύ τους.

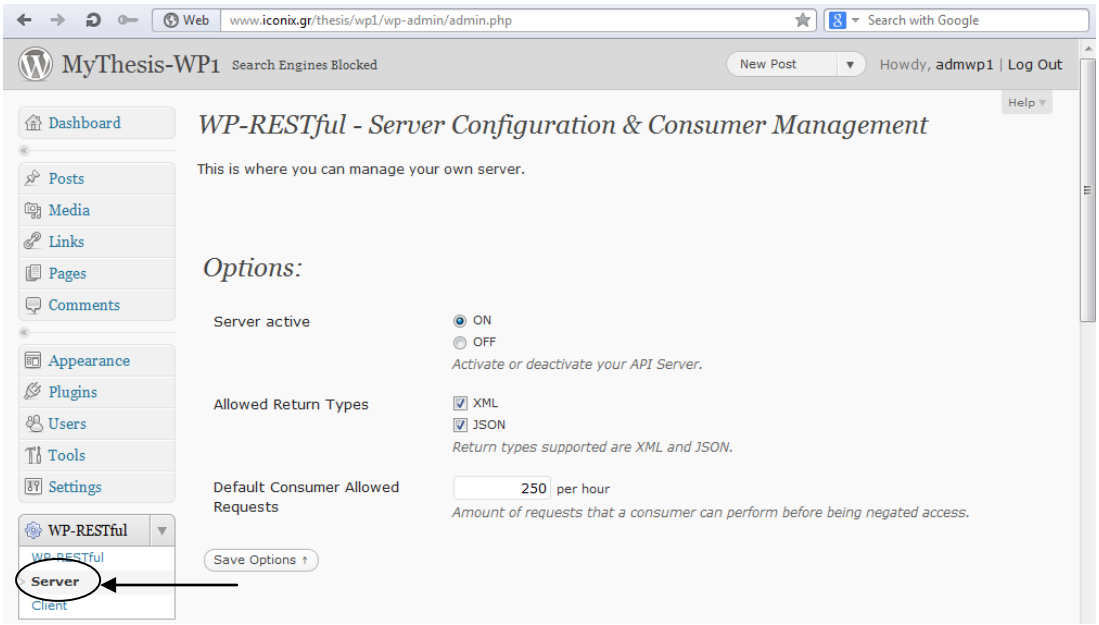
5.1 Σενάριο διασύνδεσης Moodle με WordPress

Για την επίτευξη της διασύνδεσης μεταξύ του Moodle και του WordPress προϋπόθεση είναι να έχουμε εγκαταστήσει και ρυθμίσει τα αντίστοιχα RESTFul και OAuth APIs με τα αντίστοιχα modules και στις δυο πλατφόρμες. Στην πλατφόρμα του Moodle θα δημιουργηθεί μια RESTFul υπηρεσία ως WordPress RESTful client. Ο συγκεκριμένος client θα είναι υπεύθυνος για την ασφαλή διασύνδεση του Moodle με το WordPress, καθώς και για την ορθή μεταφορά των δεδομένων, όπως για παράδειγμα των άρθρων / σελίδων / σχολίων, από το Moodle στο WordPress. Για την λήψη των δεδομένων του Moodle στο WordPress θα δημιουργηθεί μια RESTFul υπηρεσία, η οποία θα έχει τον ρόλο του RESTFul Server στην πλατφόρμα του WordPress και θα είναι υπεύθυνη για την λήψη των εξωτερικών μηνυμάτων που θα

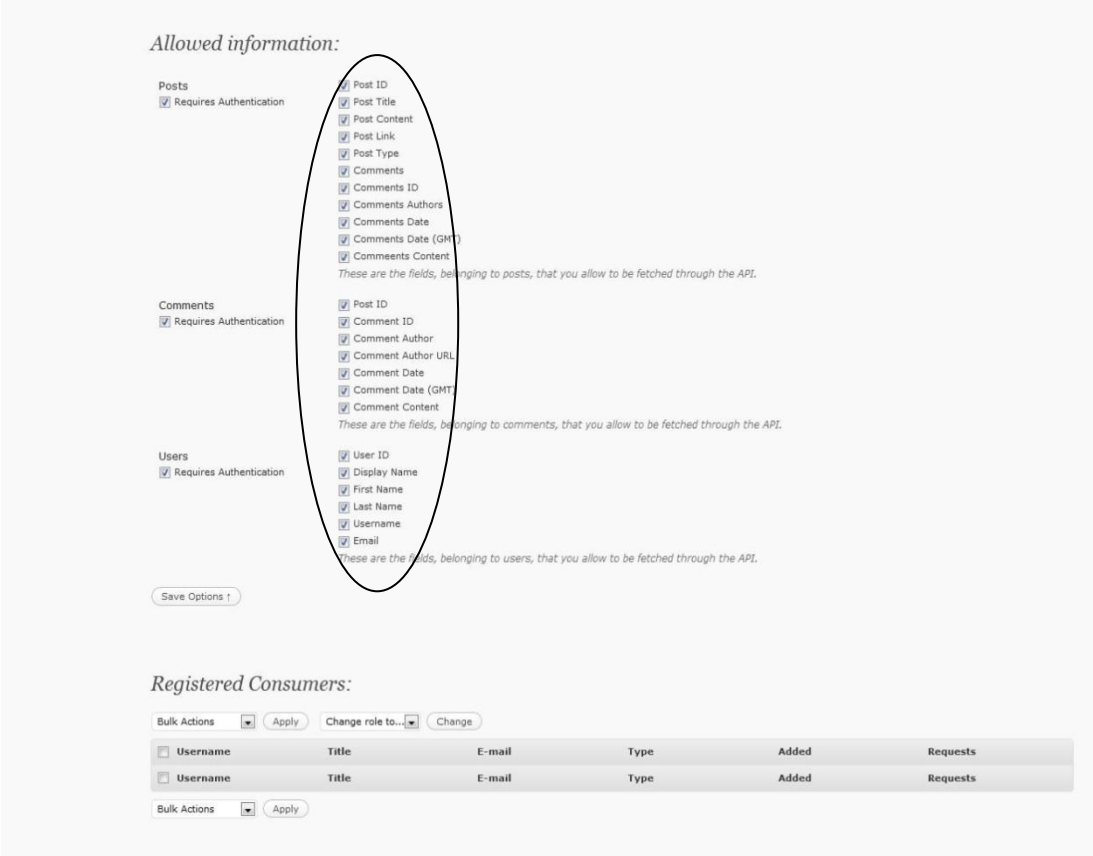
προέρχονται από το Moodle, καθώς και για την προσθήκη τους στην προοριζόμενη σελίδα ή κατηγορία του WordPress. Η διαδικασία αυτή παρουσιάζεται παρακάτω με στιγμιότυπα από την κάθε πλατφόρμα.

5.1.1 Ρύθμιση της υπηρεσίας WordPress RESTFul Server

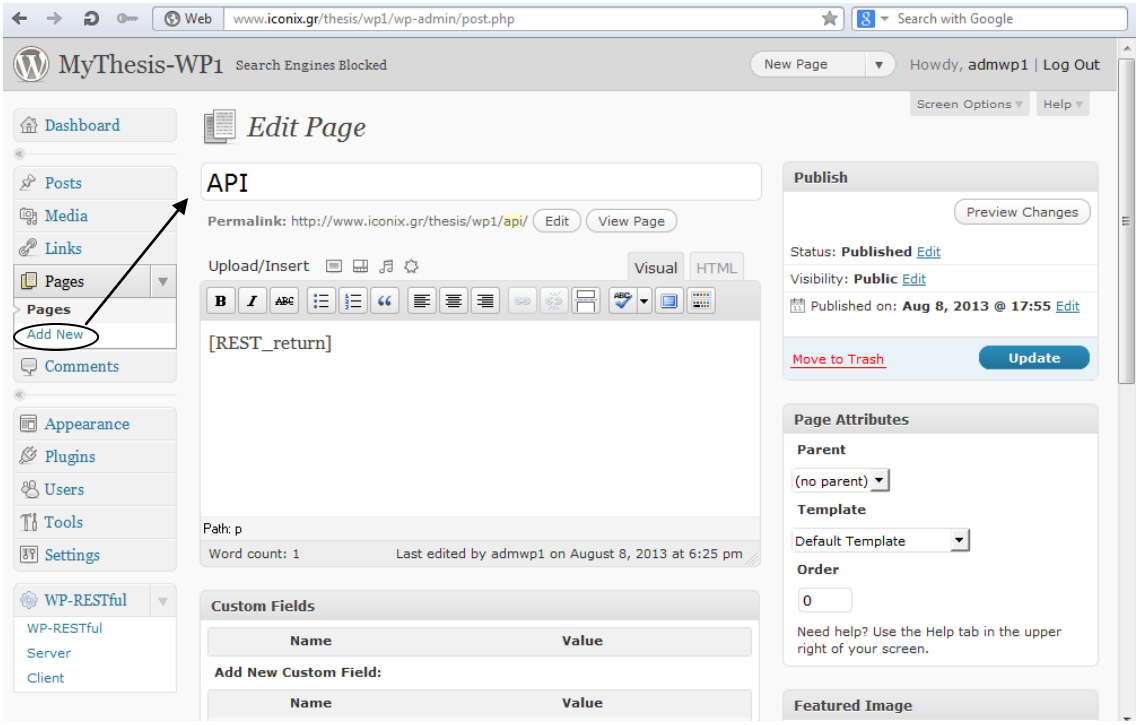
Στην Εικ.5.1 παρουσιάζεται η ενεργοποίηση και ο ρυθμός ανανέωσης λήψης δεδομένων από τις εξωτερικές εφαρμογές στις οποίες θα συνδεθεί η υπηρεσία WordPress RESTFul Server. Ενώ στην Εικ.5.2 που αποτελεί συνέχεια της Εικ.5.1 παρουσιάζεται η ρύθμιση των τύπων δεδομένων και μηνυμάτων που θα δέχεται η υπηρεσία WordPress RESTFul Server από τις αντίστοιχες υπηρεσίες WordPress RESTFul Clients των εξωτερικών εφαρμογών, όπου στην περίπτωση μας είναι το Moodle.



Εικ. 5.1 Ενεργοποίηση WordPress RESTFul Server



Εικ. 5.2 Ρύθμιση των δεδομένων και μηνυμάτων του WordPress RESTFul Server

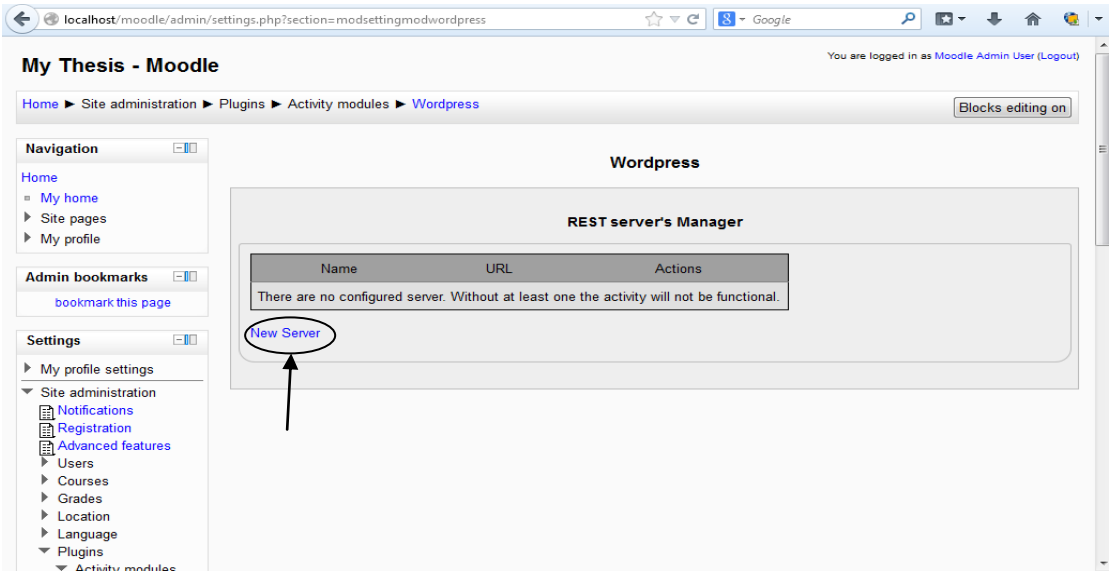


Εικ. 5.3 Δημιουργία νέας σελίδας στο WordPress για την προβολή δεδομένων του WordPress RESTFul Server

Στην Εικ.5.3 παρουσιάζεται η δημιουργία μιας σελίδας του WordPress με την ονομασία API, όπου σε αυτή την σελίδα θα προβάλλονται όλα τα δεδομένα (άρθρα / σελίδες / σχόλια) που θα λαμβάνει η υπηρεσία WordPress RESTFul Server από την υπηρεσία WordPress RESTFul Client του Moodle.

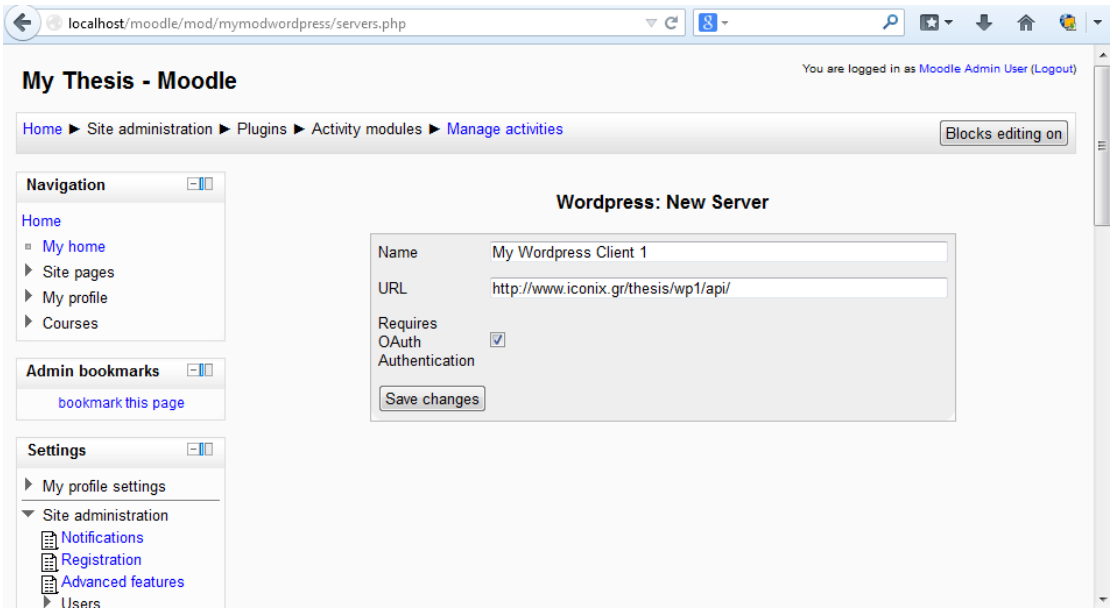
5.1.2 Ρύθμιση της υπηρεσίας WordPress RESTFul Client στο Moodle

Εφόσον έχουμε εισέλθει επιτυχώς στην πλατφόρμα του Moodle ως διαχειριστής, στη συνέχεια από το μενού Site administration → Plugins→ Activity modules→ WordPress ενεργοποιούμε την υπηρεσία WordPress RestFul Client, όπως παρουσιάζεται στην Εικ.5.4.



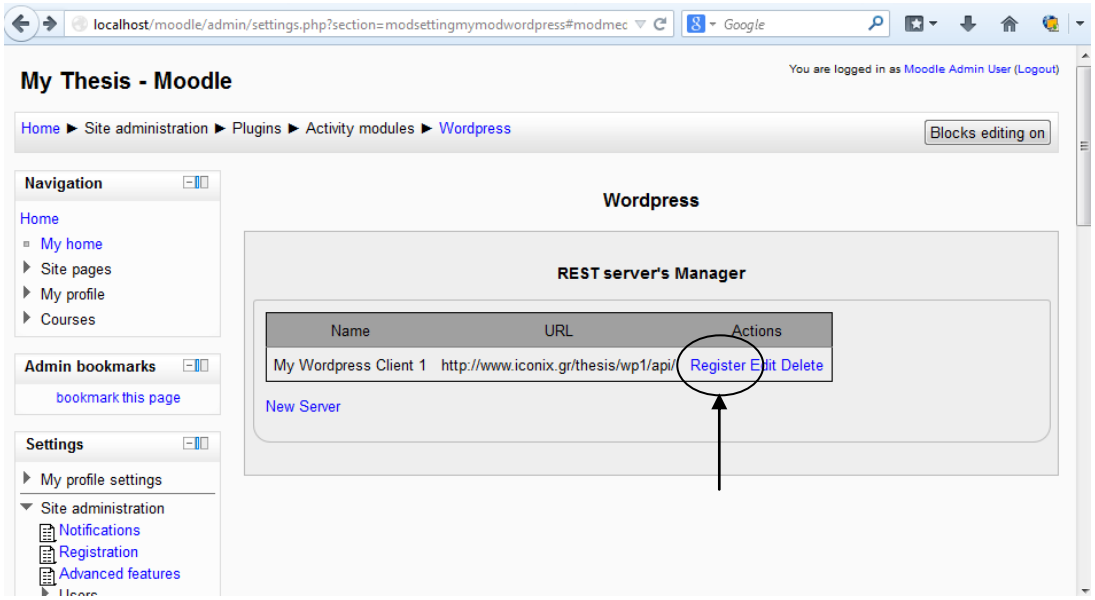
Εικ. 5.4 Ενεργοποίηση της υπηρεσίας WordPress RESTFul Client στο Moodle

Το επόμενο βήμα που ακλουθούμε για την προσθήκη και ρύθμιση του WordPress RESTFul Client είναι η συμπλήρωση της φόρμας διασύνδεσης της υπηρεσίας WordPress RESTFul Client με τα αντίστοιχα και απαραίτητα δεδομένα της υπηρεσίας WordPress RESTFul Server που ενεργοποιήσαμε παραπάνω. Για την μετάβαση σε αυτή την φόρμα επιλέγουμε τον σύνδεσμο «New Server» (Εικ.5.4), ενώ η διαδικασία συμπλήρωσης της παρουσιάζεται στην Εικ.5.5.



Εικ. 5.5 Συμπλήρωση της φόρμας διασύνδεσης της υπηρεσίας WordPress RESTFul Client με τα δεδομένα της υπηρεσίας WordPress RESTFul Server.

Το αποτέλεσμα της διασύνδεσης της υπηρεσίας WordPress RESTFul Client με την υπηρεσία WordPress RESTFul Server παρουσιάζεται στην παρακάτω Εικ. 5.6.

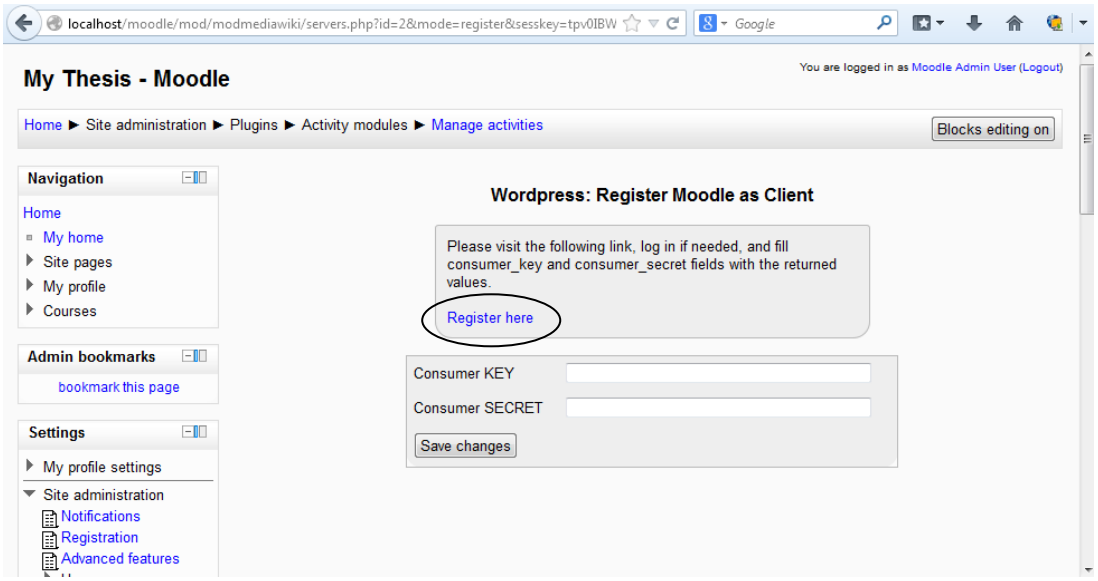


Εικ. 5.6 Πρώτο βήμα διασύνδεσης της υπηρεσίας WordPress RESTFul Client με την WordPress RESTFul Server.

Εφόσον έχουμε ολοκληρώσει επιτυχώς το πρώτο βήμα διασύνδεσης μεταξύ των υπηρεσιών WordPress RESTFul Client του Moodle και WordPress RESTFul Server του WordPress, το επόμενο βήμα που ακολουθεί είναι η πιστοποίηση της διασύνδεσης μεταξύ αυτών των δυο υπηρεσιών μέσω του πρωτοκόλλου OAuth 2.0 επιλέγοντας τον σύνδεσμο «Register» (Εικ.5.6). Η διαδικασία αυτή επιτυγχάνεται με την παραγωγή των consumer_key και consumer_token από

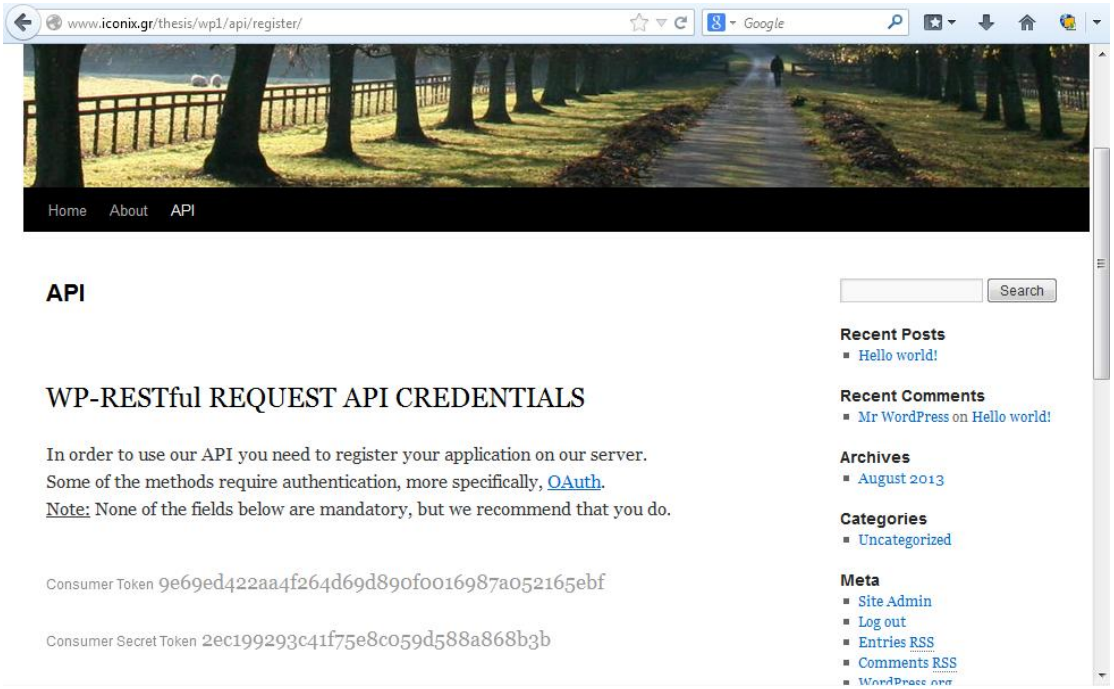
την υπηρεσία WordPress RESTFul Server του WordPress και με την προσθήκη αυτών των πιστοποιητικών στην υπηρεσία εγγραφής client του WordPress RESTFul Client του Moodle.

Για την λήψη των consumer_key και consumer_secret από την υπηρεσία WordPress RESTFul Server του WordPress επιλέγουμε τον σύνδεσμο «Register here», όπως παρουσιάζεται στην παρακάτω εικόνα (Εικ. 5.7).



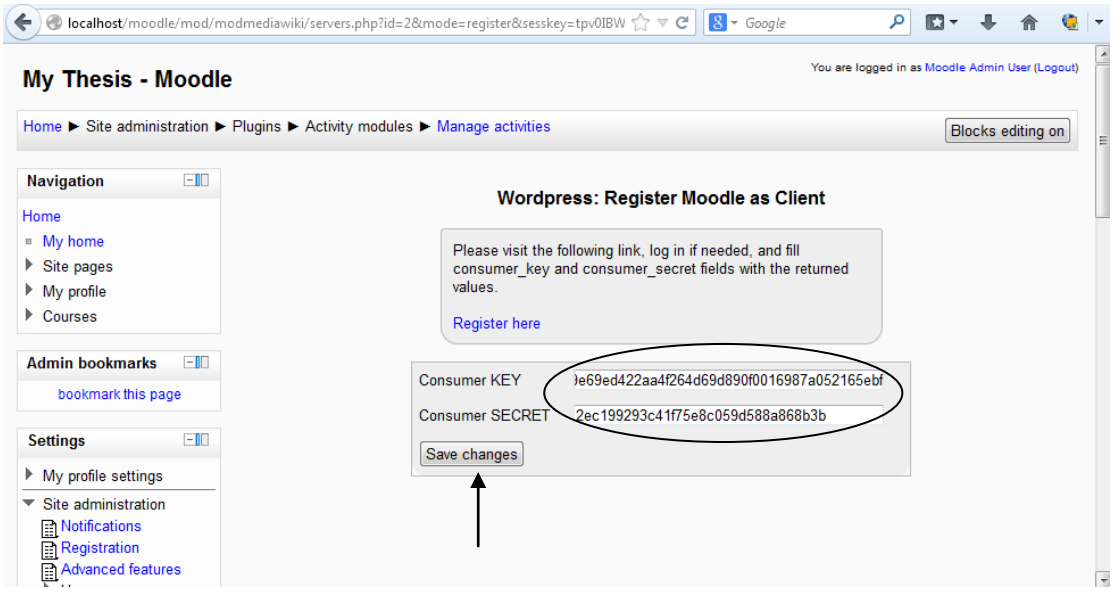
Εικ. 5.7 Λήψη των consumer_key και consumer_secret από την υπηρεσία WordPress RESTFul Server

Εφόσον έχουμε εκτελέσει την παραπάνω διαδικασία, η υπηρεσία WordPress RESTFul Client του Moodle, ανακατευθύνεται στην WordPress RESTFul Server του WordPress. Στη συνέχεια, η WordPress RESTFul Server υπηρεσία του WordPress παράγει και προβάλει σε μια σελίδα της, τα δυο πιστοποιητικά (consumer_key και consumer_secret) που απαιτούνται για την διασύνδεση με την υπηρεσία WordPress RESTFul Client του Moodle. Η διαδικασία αυτή παρουσιάζεται στην παρακάτω Εικ. 5.8.



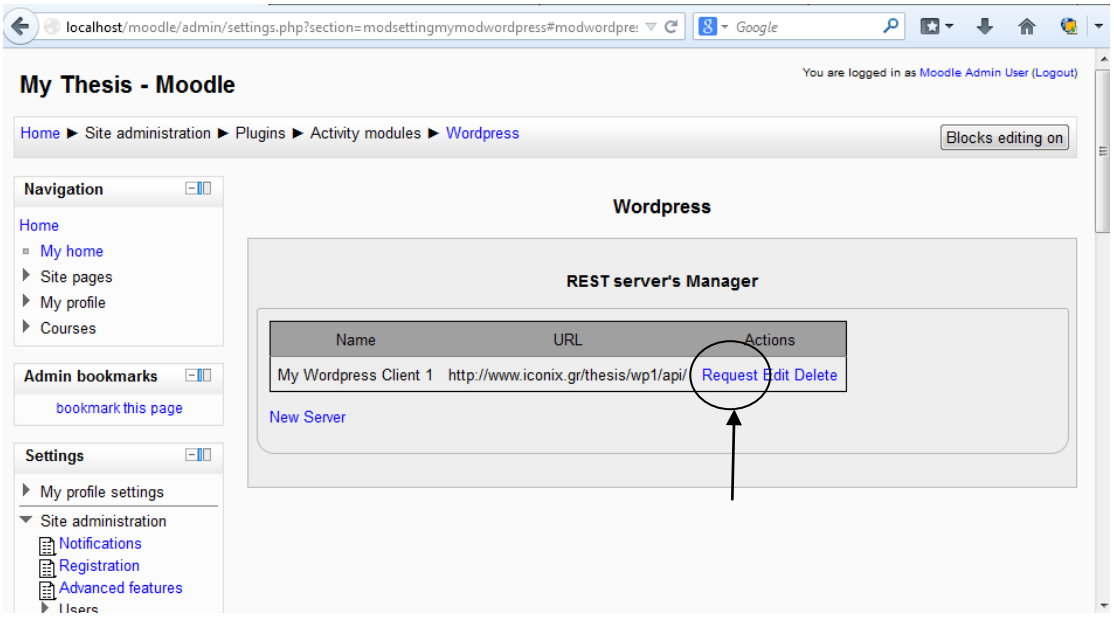
Εικ. 5.8 Προβολή των consumer_key και consumer_secret στο WordPress.

Τα δυο πιστοποιητικά που εμφανίζονται στην Εικ.5.8, τα αντιγράφουμε στα αντίστοιχα πεδία της φόρμας έγγραφης της υπηρεσίας WordPress RESTFul Client στο Moodle και στη συνέχεια τα αποθηκεύουμε επιλέγοντας το κουμπί «Save changes» της φόρμας εγγραφής.



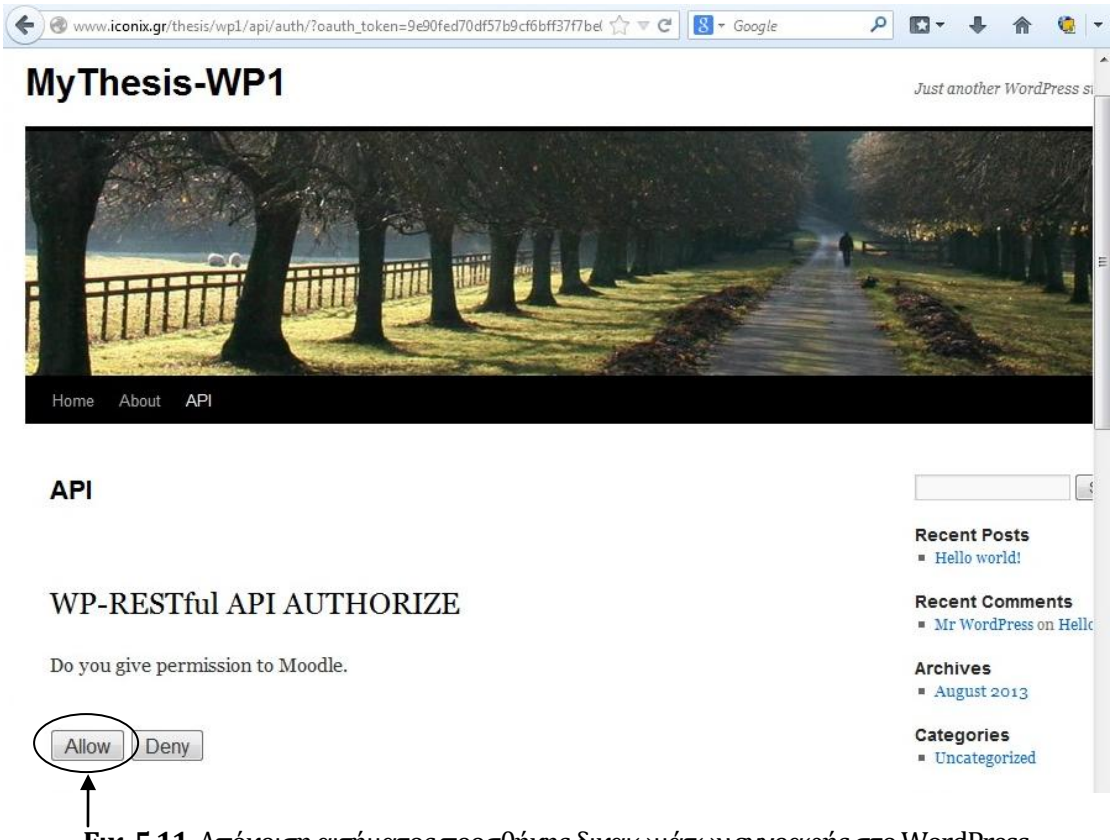
Εικ. 5.9 Προσθήκη των consumer_key και consumer_secret στη φόρμα εγγραφής της υπηρεσίας WordPress RESTFul Client στο Moodle

Τέλος, για την προσθήκη δικαιωμάτων εγγραφής της υπηρεσίας WordPress RESTFul Client του Moodle στο WordPress επιλέγουμε τον σύνδεσμο «Request» όπως παρουσιάζεται στην Εικ. 5.10.



Εικ. 5.10 Υποβολή αιτήματος δικαιωμάτων εγγραφής στο WordPress.

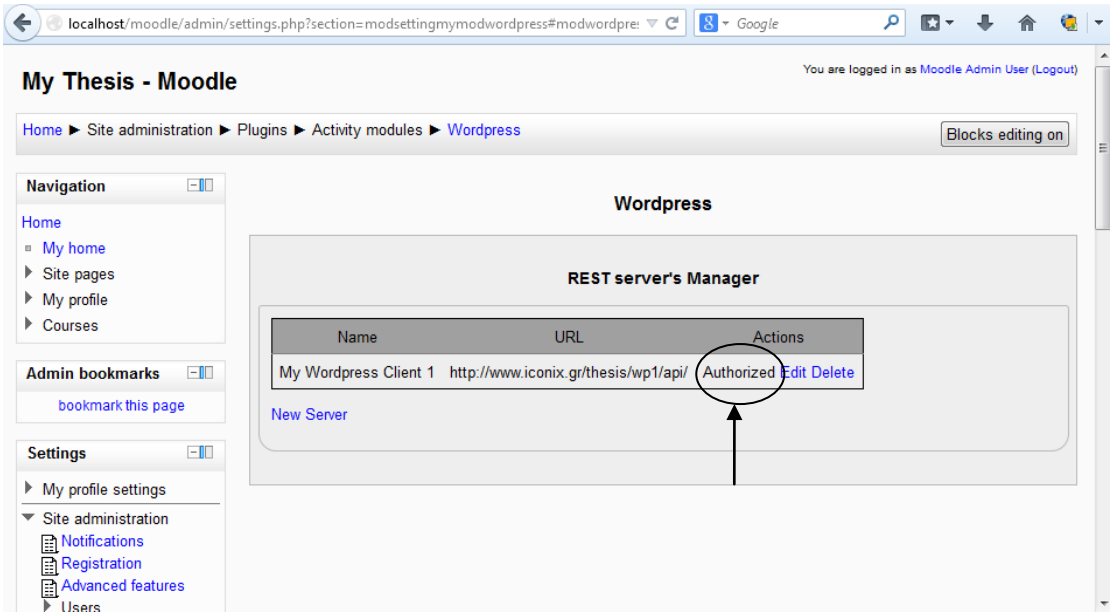
Έτσι, η υπηρεσία WordPress RESTFul Server του WordPress αποκρίνεται στο αίτημα αυτό όπως παρουσιάζεται στην Εικ.5.11, όπου και επιλέγουμε το κουμπί «Allow» για την ολοκλήρωση της διαδικασίας προσθήκης δικαιωμάτων εγγραφής (Εικ. 5.12).



Εικ. 5.11 Απόκριση αιτήματος προσθήκης δικαιωμάτων εγγραφής στο WordPress

Κατά την διάρκεια ολοκλήρωσης αυτής της διαδικασίας στην υπηρεσία WordPress RESTFul Client του Moodle αποστέλλονται, από την υπηρεσία WordPress RESTFul Server του WordPress,

δυο επιπρόσθετα πιστοποιητικά πρόσβασης και εξουσιοδότησης, τα οποία και αποθηκεύονται στην βάση δεδομένων του Moodle. Έτσι, με την ολοκλήρωση της διαδικασίας αυτής, η υπηρεσία WordPress RESTFul Client του Moodle έχει διασυνδεθεί και πιστοποιηθεί με την υπηρεσία WordPress RESTFul Server του WordPress, όπως παρουσιάζεται στην Εικ.5.12 ως Authorized. Ως εκ τούτου, όταν δημιουργούμε ένα άρθρο στην πλατφόρμα του Moodle, αυτό προστίθεται και προβάλλεται παράλληλα και στην πλατφόρμα του WordPress.

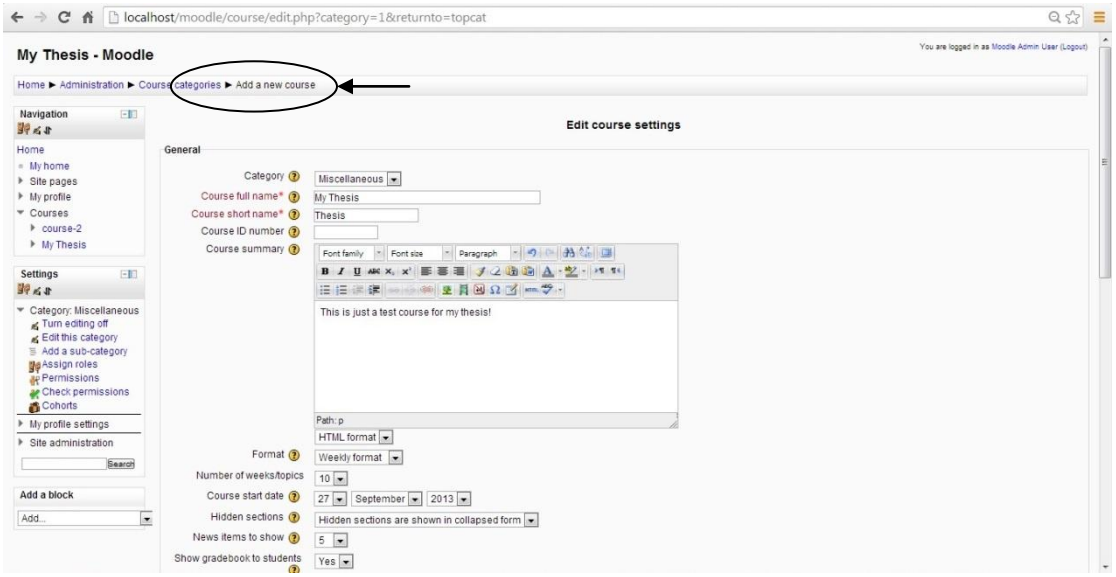


Εικ. 5.12 Ολοκλήρωση της διαδικασίας προσθήκης δικαιωμάτων εγγραφής στο WordPress

5.1.3 Δημιουργία νέου άρθρου στο WordPress μέσω Moodle Restful client

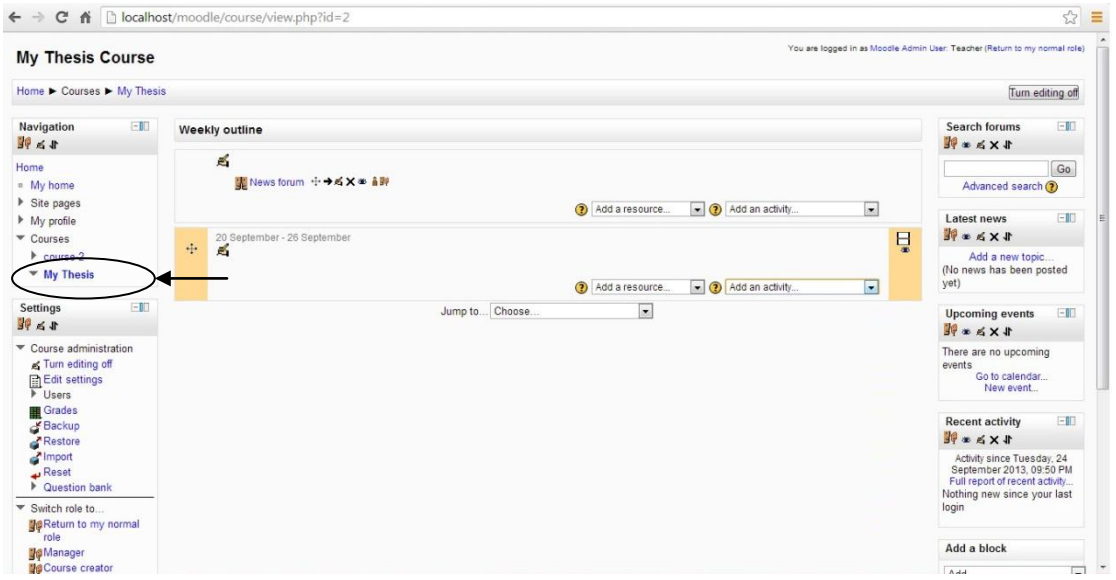
Για την προσθήκη ενός άρθρου στο WordPress μέσω του Moodle Restful client ακολουθούμε την παρακάτω διαδικασία:

a) Εφόσον έχει ολοκληρωθεί η πιστοποίηση των χρηστών μεταξύ των πλατφορμών του WordPress και του Moodle, δημιουργούμε ένα νέο μάθημα (course) με το όνομα "My Thesis" στην πλατφόρμα του Moodle ακολουθώντας την διαδρομή Administration > Course Categories > Add New Course, όπως παρουσιάζεται στην παρακάτω εικόνα (Εικ. 5.13). Για την επιτυχή δημιουργία του μαθήματος θα πρέπει να συμπληρώσουμε όλα τα προαπαιτούμενα πεδία της φόρμας που εμφανίζονται στην εικόνα (Εικ.5.13).



Εικ. 5.13 Δημιουργία ενός νέου μαθήματος (course) στο Moodle

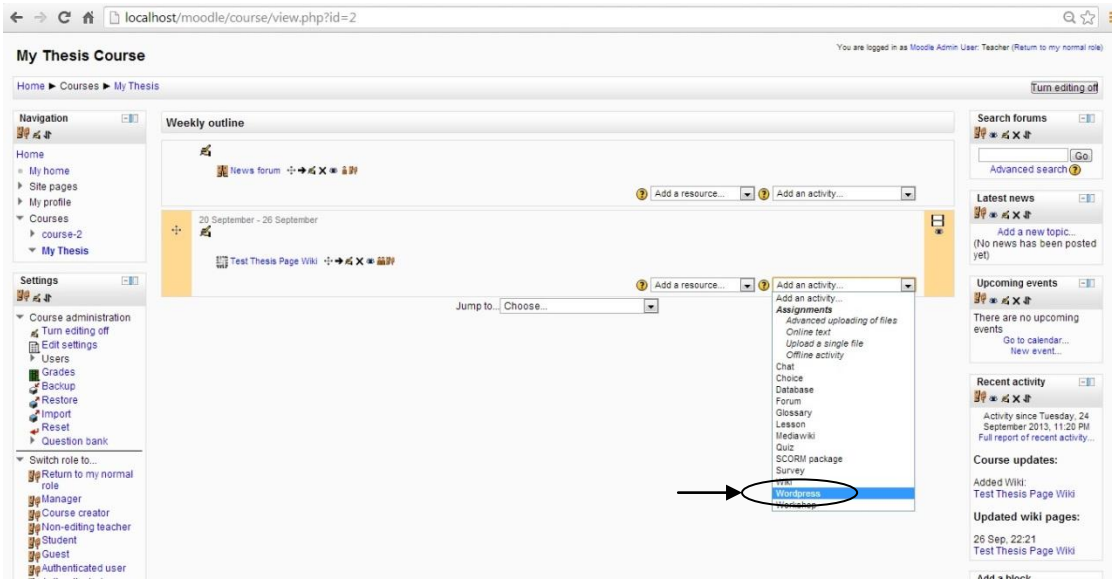
b) Εφόσον έχει δημιουργηθεί επιτυχώς το μάθημα "My Thesis" στην πλατφόρμα του Moodle, από το κεντρικό μενού πλοήγησης του Moodle, επιλεγούμε αυτό το μάθημα ακολουθώντας την διαδρομή Home > Courses > My Thesis όπως παρουσιάζεται και στην παρακάτω εικόνα (Εικ. 5.14).



Εικ. 5.14 Επιλογή του μαθήματος "My Thesis" στο Moodle

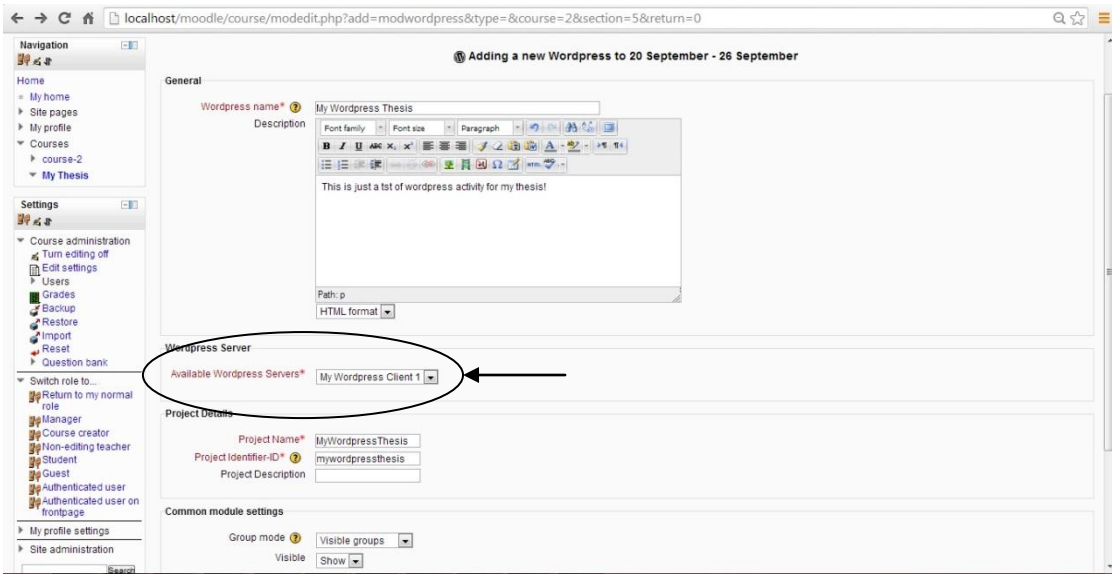
c) Εφόσον έχουμε επιλέξει το μάθημα "My Thesis" στο πλαίσιο της πρώτης εβδομάδας επιλέγουμε από το "Activity drop down" menu την επιλογή WordPress (Εικ.5.15). Μέσω αυτής της διαδικασίας, θα επιτύχουμε την ενεργοποίηση WordPress activity και την προσθήκη σε αυτό ενός νέου άρθρου όπου το περιεχόμενο του θα παρουσιάζεται

παράλληλα στο WordPress και στην πρώτη εβδομάδα του μαθήματος "My Thesis" στην πλατφόρμα του Moodle.



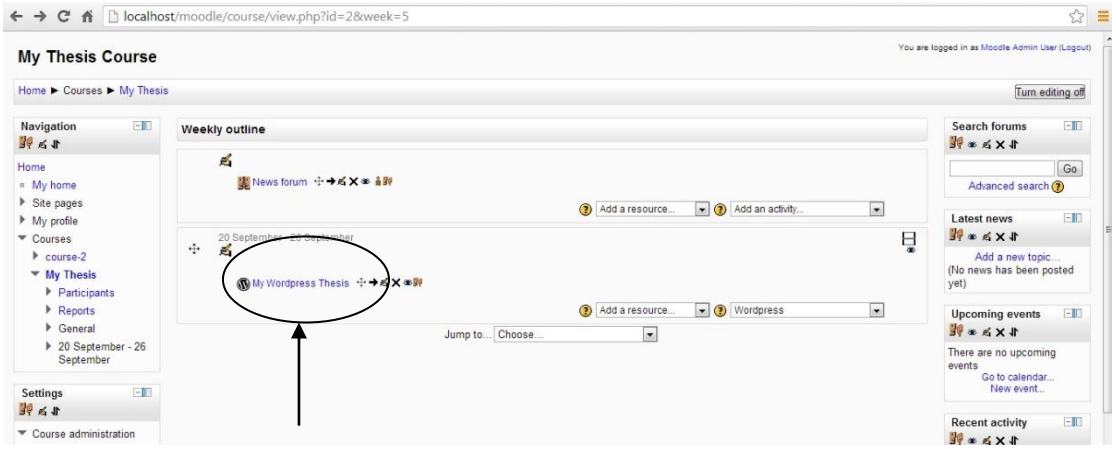
Εικ. 5.15 Ενεργοποίηση του WordPress activity στο μάθημα "My Thesis" στο Moodle

d) Για την επιτυχή ενεργοποίηση του WordPress activity και για την δημιουργία ενός νέου άρθρου στο WordPress, καθώς και στην πλατφόρμα του Moodle στο μάθημα "My Thesis", θα πρέπει να συμπληρώσουμε όλα τα προαπαιτούμενα πεδία της φόρμας που εμφανίζεται στην εικόνα (Εικ.5.16). Αξίζει να αναφέρουμε εδώ ότι, στα στοιχεία που συμπληρώνουμε στην φόρμα που παρουσιάζεται στην εικόνα (Εικ.5.16) ορίζουμε και τον WordPress Restful Client "My WordPress Client 1" που ενεργοποιήσαμε παραπάνω. Δηλαδή, στην πραγματικότητα ορίζουμε τον WordPress Sever που θα δημοσιεύεται το νέο άρθρο που θα δημιουργήσουμε παρακάτω.



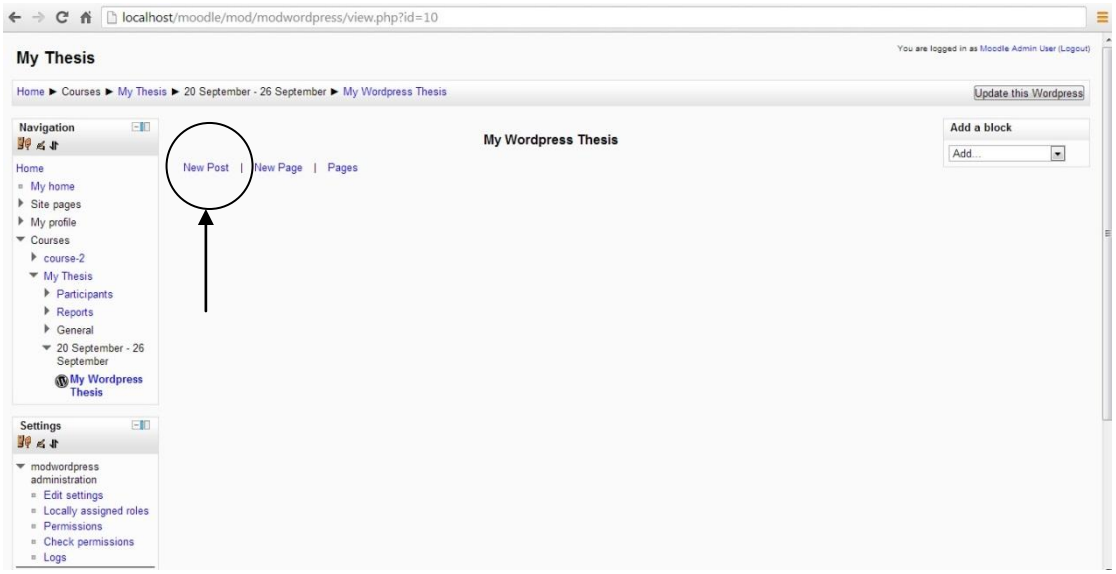
Εικ. 5.16 Ορισμός του WordPress Restful Client "My WordPress Client 1"

e) Εφόσον έχουμε ενεργοποιήσει επιτυχώς το WordPress Activity με το όνομα "My WordPress Thesis" στην πρώτη εβδομάδα του μαθήματος "My Thesis" στο Moodle, μπορούμε πλέον να προσθέσουμε και το πρώτο μας άρθρο στο WordPress Activity. Έτσι, για την δημιουργία του άρθρου στο WordPress, αρχικά επιλέγουμε τον σύνδεσμο του WordPress Activity "My WordPress Thesis", όπως παρουσιάζεται στην παρακάτω εικόνα (Εικ. 5.17).



Εικ. 5.17 Επιλογή του WordPress Activity "My WordPress Thesis" στο Moodle

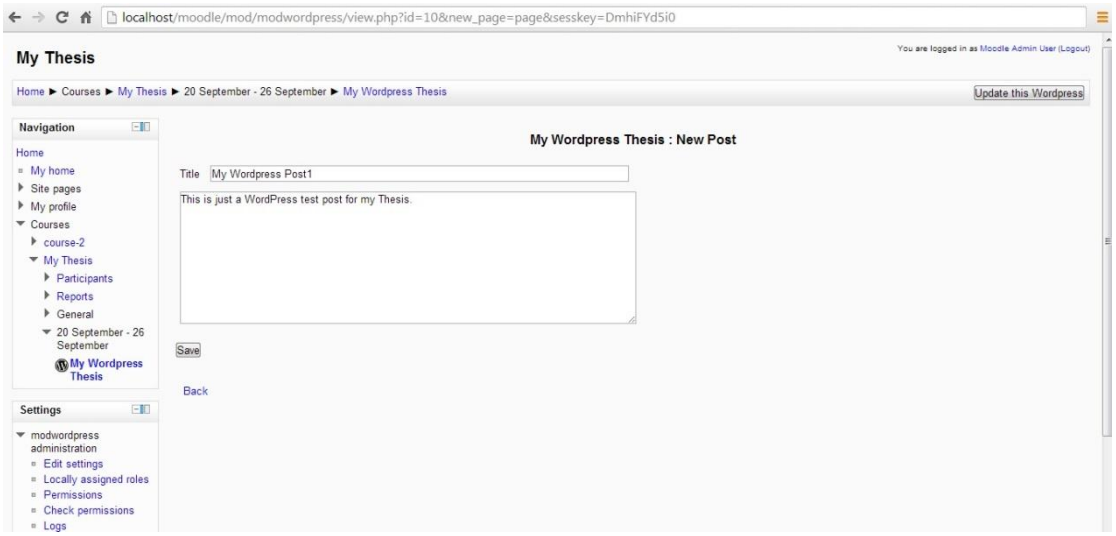
f) Εφόσον έχουμε επιλέξει το "My WordPress Thesis" activity, στο εσωτερικό μενού του, επιλέγουμε τον σύνδεσμο "New Post". Η διαδικασία αυτή παρουσιάζεται στην παρακάτω εικόνα (Εικ. 5.18).



Εικ. 5.18 Επιλογή του συνδέσμου "New Post" στο "My WordPress Thesis" activity για την δημιουργία νέου άρθρου

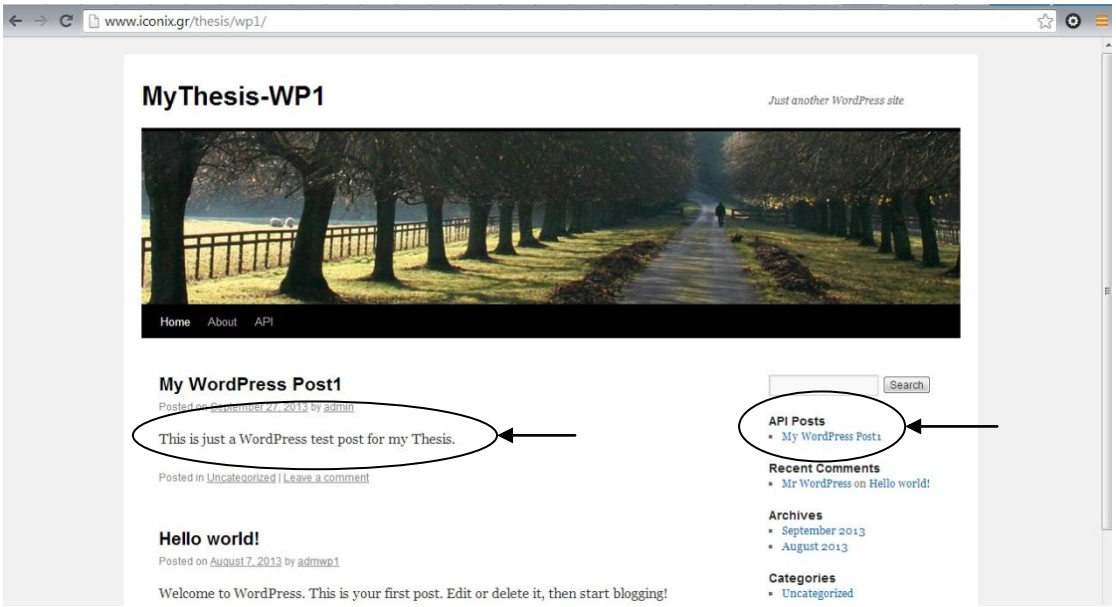
g) Στη συνέχεια συμπληρώνουμε τα στοιχεία του νέου άρθρου που θα παρουσιαστούν στην πλατφόρμα του WordPress και στην πλατφόρμα του Moodle στην πρώτη εβδομάδα του

μαθήματος “My Thesis” (Εικ.5.19). Για την αποθήκευση του άρθρου επιλέγουμε το κουμπί “Save” της φόρμας.



Εικ. 5.19 Δημιουργία του WordPress άρθρου με τίτλο “My WordPress Post1”

h) Τέλος, το άρθρο που δημιουργήσαμε στο Moodle παρουσιάζεται στο WordPress μέσω του WordPress Restful API Widget, όπως στην παρακάτω εικόνα. (Εικ. 5.20). Αξίζει να σημειώσουμε εδώ ότι, τα άρθρα που προστίθενται στο WordPress μέσω του Moodle αρχικά δεν κατηγοριοποιούνται. Όμως παρέχεται η δυνατότητα στον διαχειριστή του WordPress να τα κατηγοριοποιήσει σε μεταγενέστερο χρόνο.



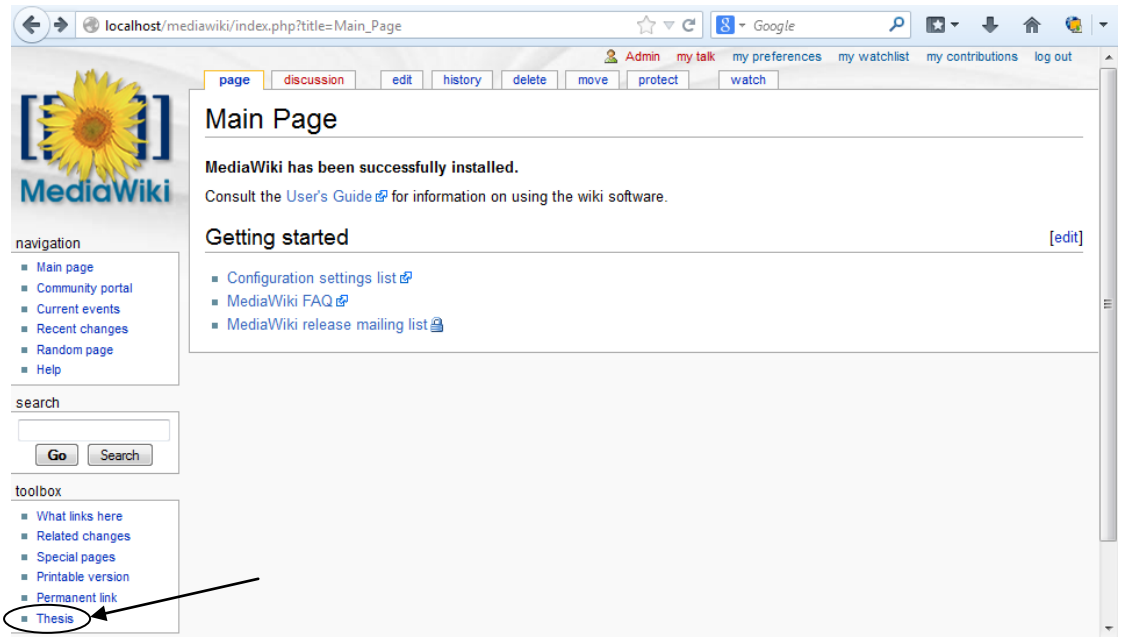
Εικ. 5.20 Προβολή του άρθρου στο WordPress με τίτλο “My WordPress Post1”

5.2 Σενάριο διασύνδεσης Moodle με MediaWiki

Όπως με την περίπτωση της διασύνδεσης μεταξύ του Moodle και του WordPress, και εδώ προϋπόθεση είναι να έχουμε εγκαταστήσει και ρυθμίσει τα αντίστοιχα RESTFul και OAuth APIs με τα αντίστοιχα modules και στις δυο πλατφόρμες. Στην πλατφόρμα του Moodle θα δημιουργηθεί μια RESTFul υπηρεσία ως MediaWiki RESTFul client. Ο συγκεκριμένος client θα είναι υπεύθυνος για την ασφαλή διασύνδεση του Moodle με το MediaWiki καθώς και για την ορθή μεταφορά των δεδομένων, όπως για παράδειγμα των σελίδων, από το Moodle στο MediaWiki. Για την λήψη των δεδομένων του Moodle στο MediaWiki θα δημιουργηθεί μια RESTFul υπηρεσία, η οποία θα έχει τον ρόλο του RESTFul Server στην πλατφόρμα του MediaWiki και θα είναι υπεύθυνη για την λήψη των εξωτερικών μηνυμάτων που θα προέρχονται από το Moodle, καθώς και για την προσθήκη τους στην προοριζόμενη σελίδα του MediaWiki. Η διαδικασία αυτή παρουσιάζεται παρακάτω με στιγμιότυπα από την κάθε πλατφόρμα.

5.2.1 Ρύθμιση της υπηρεσίας MediaWiki RESTFul Server

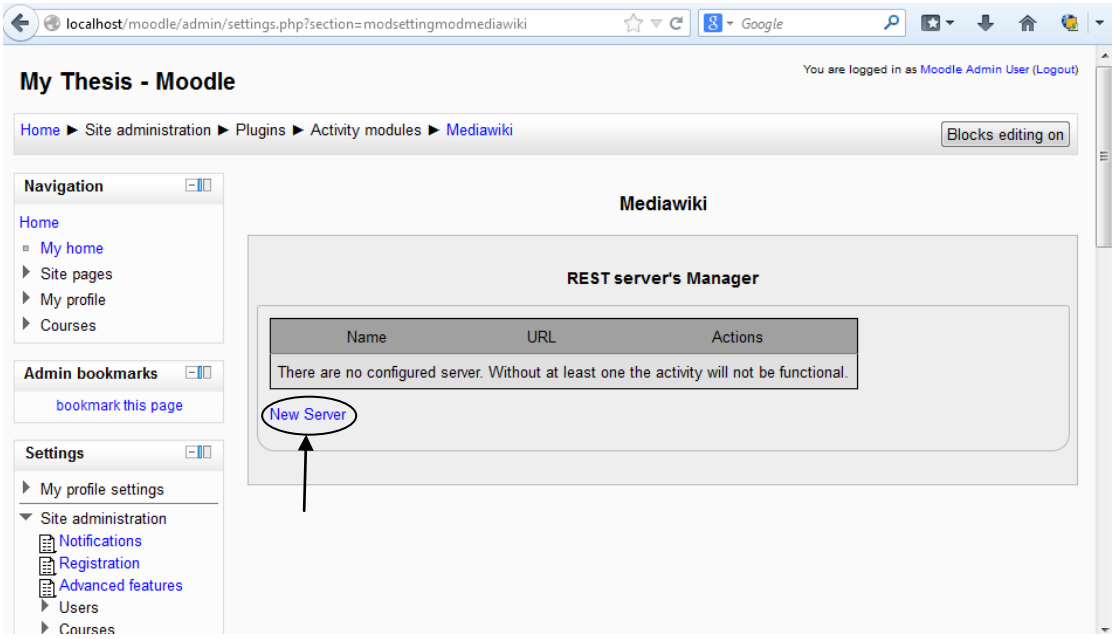
Στα extensions της πλατφόρμας του MediaWiki προσθέτουμε την υπηρεσία RESTFul Server που υλοποιήσαμε, όπου μετά την εκτέλεση της, στο μενού «Toolbox» της πλατφόρμας του MediaWiki εμφανίζεται ένας σύνδεσμος σελίδας με το όνομα «Thesis» (Εικ. 5.21). Σε αυτή την σελίδα θα προβάλλονται όλα τα δεδομένα που θα προέρχονται από την πλατφόρμα του Moodle, καθώς και διάφορα μηνύματα της υπηρεσίας MediaWiki RESTFul Server.



Εικ. 5.21 Ενεργοποίηση της υπηρεσίας MediaWiki RESTFul Server

5.2.2 Ρύθμιση της υπηρεσίας MediaWiki RESTFul Client στο Moodle

Εφόσον έχουμε εισέλθει επιτυχώς στην πλατφόρμα του Moodle ως διαχειριστής, στη συνέχεια από το μενού Site administration→ Plugins→ Activity modules→ MediaWiki ενεργοποιούμε την υπηρεσία MediaWiki RestFul Client, όπως παρουσιάζεται στην Εικ.5.22.

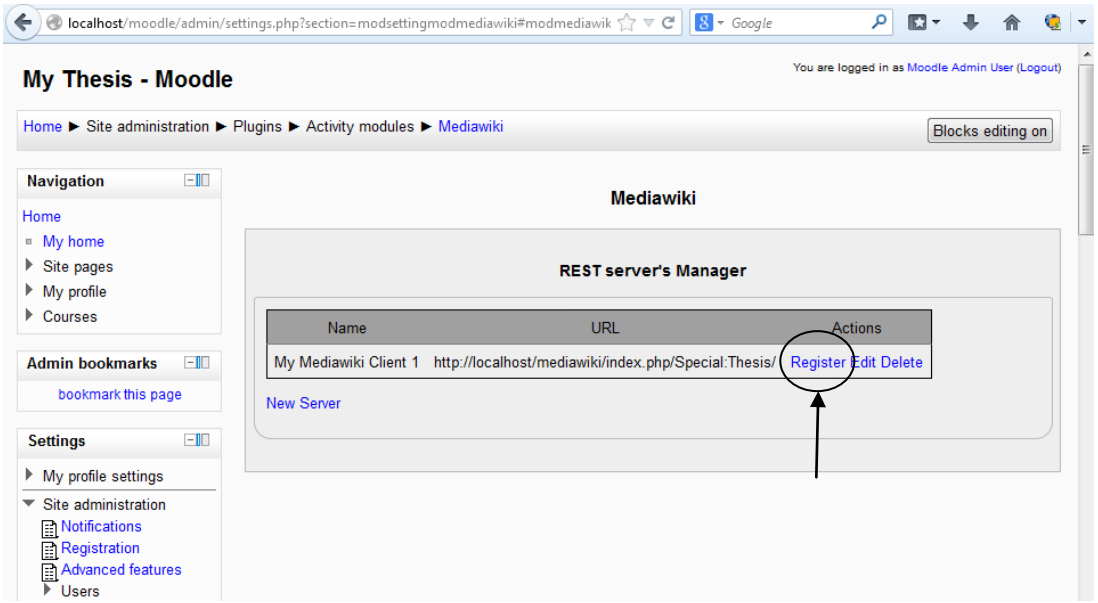


Εικ. 5.22 Ενεργοποίηση της υπηρεσίας MediaWiki RESTFul Client στο Moodle

Το επόμενο βήμα που ακολουθούμε για την προσθήκη και ρύθμιση του MediaWiki RESTFul Client είναι η συμπλήρωση της φόρμας διασύνδεσης της υπηρεσίας MediaWiki RESTFul Client με τα αντίστοιχα και απαραίτητα δεδομένα της υπηρεσίας MediaWiki RESTFul Server που ενεργοποιήσαμε παραπάνω. Για την μετάβαση σε αυτή την φόρμα επιλέγουμε τον σύνδεσμο «New Server» (Εικ. 5.22), ενώ η διαδικασία συμπλήρωσης της παρουσιάζεται στην Εικ. 5.23.

Εικ. 5.23 Συμπλήρωση της φόρμας διασύνδεσης της υπηρεσίας MediaWiki RESTFul Client με τα δεδομένα της υπηρεσίας MediaWiki RESTFul Server

Το αποτέλεσμα του πρώτου βήματος της διασύνδεσης της υπηρεσίας MediaWiki RESTFul Client με την υπηρεσία MediaWiki RESTFul Server παρουσιάζεται στην παρακάτω Εικ. 5.24.

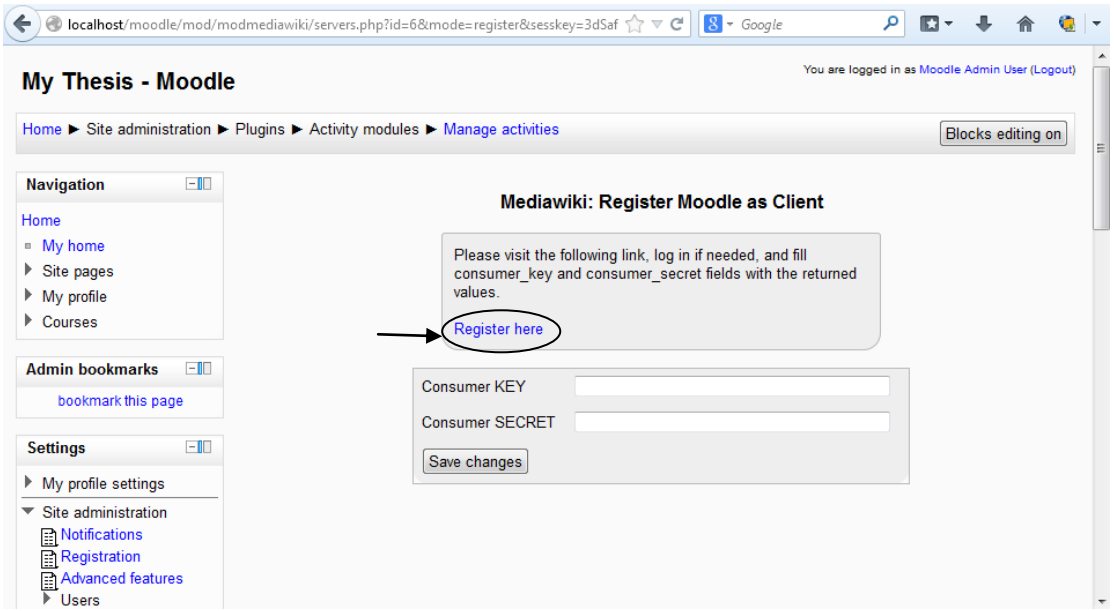


Εικ. 5.24 Πρώτο βήμα διασύνδεσης της υπηρεσίας MediaWiki RESTFul Client με την MediaWiki RESTFul Server

Εφόσον έχουμε ολοκληρώσει επιτυχώς το πρώτο βήμα διασύνδεσης μεταξύ των υπηρεσιών MediaWiki RESTFul Client του Moodle και Mediawiki RESTFul Server του MediaWiki, το επόμενο βήμα που ακολουθεί είναι η πιστοποίηση της διασύνδεσης μεταξύ αυτών των δυο υπηρεσιών μέσω του πρωτοκόλλου OAuth 2.0 επιλέγοντας τον σύνδεσμο «Register» (Εικ.5.24). Η διαδικασία αυτή επιτυγχάνεται με την παραγωγή των consumer_key και consumer_token από

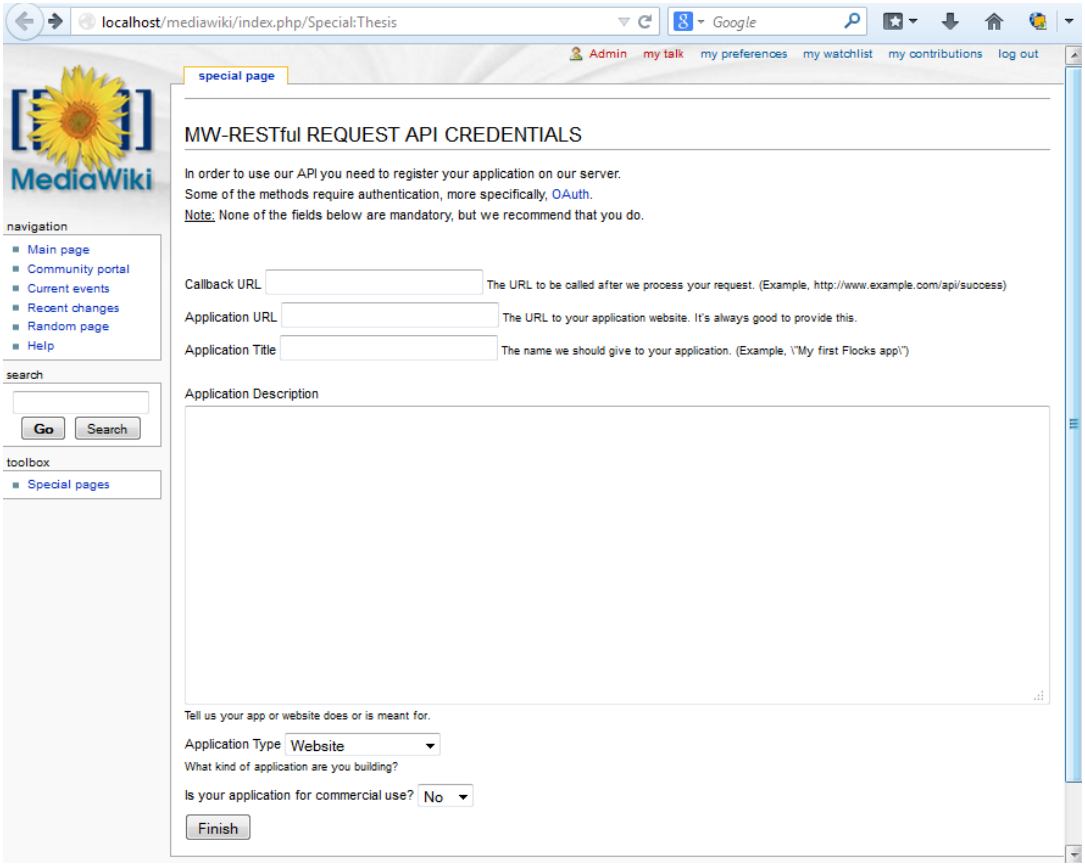
την υπηρεσία MediaWiki RESTFul Server του MediaWiki και με την προσθήκη αυτών των πιστοποιητικών στην υπηρεσία εγγραφής client του MediaWiki RESTFul Client του Moodle.

Για την λήψη των consumer_key και consumer_secret από την υπηρεσία MediaWiki RESTFul Server του MediaWiki επιλέγουμε τον σύνδεσμο «Register here», όπως παρουσιάζεται στην παρακάτω Εικ. 5.25.



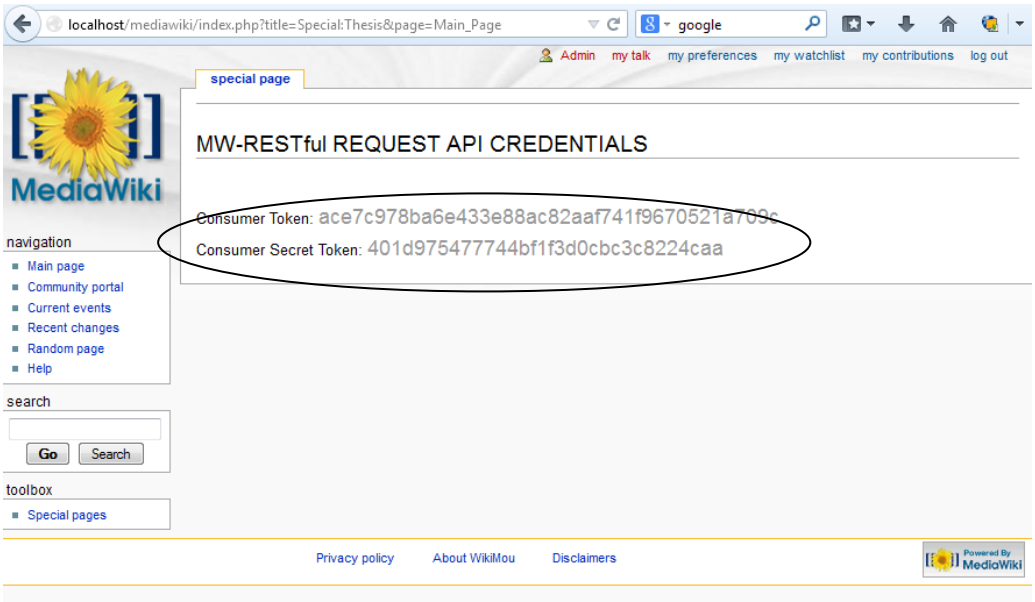
Εικ. 5.25 Λήψη των consumer_key και consumer_secret από την MediaWiki RESTFul Server υπηρεσία

Εφόσον έχουμε εκτελέσει την παραπάνω διαδικασία, η υπηρεσία MediaWiki RESTFul Client ανακατευθύνεται στην υπηρεσία MediaWiki RESTFul Server του MediaWiki, όπου αυτή με την σειρά της παράγει και προβάλει σε μια σελίδα της, τα δυο πιστοποιητικά (consumer_key και consumer_secret) που απαιτούνται για την διασύνδεση με την υπηρεσία MediaWiki RESTFul Client του Moodle. Η διαδικασία αυτή παρουσιάζεται στην παρακάτω Εικ. 5.27. Αξίζει να σημειώσουμε εδώ, ότι για την παραγωγή των δυο παραπάνω πιστοποιητικών η υπηρεσία MediaWiki RESTFul Server του MediaWiki θα μας παραπέμψει να συμπληρώσουμε μια φόρμα με στοιχεία που αφορούν την υπηρεσία MediaWiki RESTFul Client του Moodle (Εικ. 5.26).



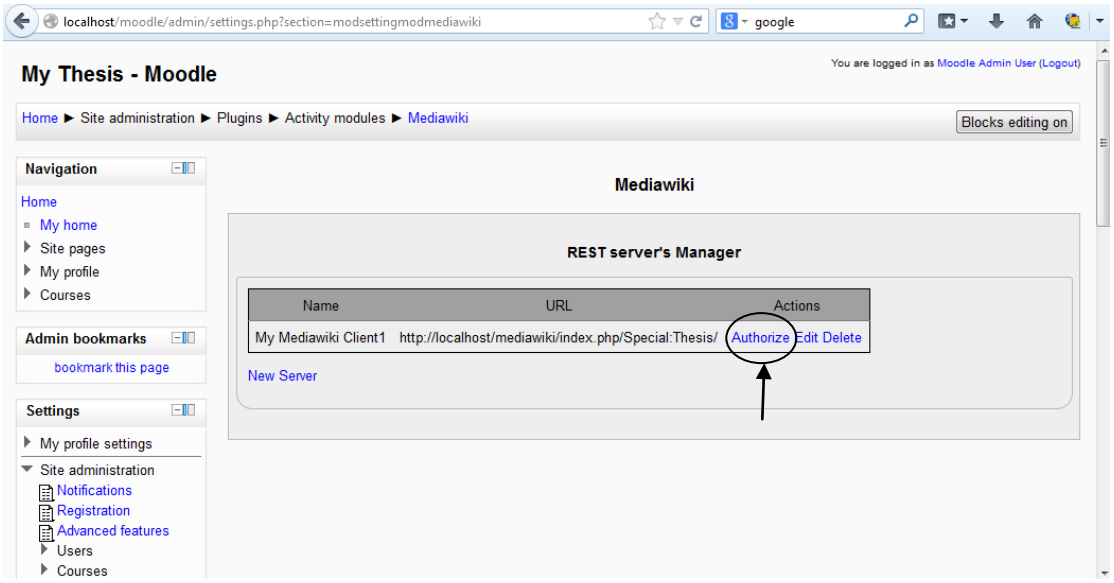
Εικ. 5.26 Φόρμα πιστοποίησης RESTFul Client της υπηρεσίας MediaWiki RESTFul Server του MediaWiki

Εφόσον συμπληρώσουμε την παραπάνω φόρμα (Εικ.5.26) και επιλέξουμε το κουμπί «Finish», δημιουργούνται τα δυο πιστοποιητικά όπως εμφανίζονται στην Εικ.5.27, τα οποία τα αντιγράφουμε στα αντίστοιχα πεδία της φόρμας εγγραφής της υπηρεσίας MediaWiki RESTFul Client στο Moodle και στη συνέχεια τα αποθηκεύουμε επιλέγοντας το κουμπί «Save changes» της φόρμας έγγραφης (Εικ.5.25).



Εικ. 5.27 Προβολή των consumer_key και consumer_secret στο MediaWiki.

Τέλος, για την προσθήκη δικαιωμάτων εγγραφής της υπηρεσίας MediaWiki RESTFul Client του Moodle στο MediaWiki επιλέγουμε τον σύνδεσμο «Authorize» όπως παρουσιάζεται στην εικόνα (Εικ.5.28).

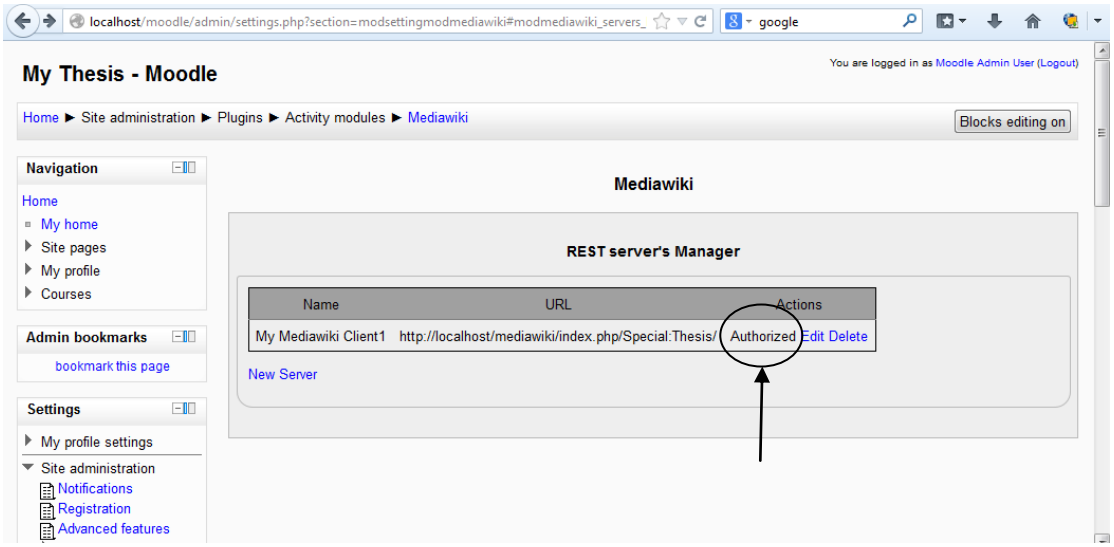


Εικ. 5.28 Υποβολή αιτήματος δικαιωμάτων εγγραφής στο MediaWiki

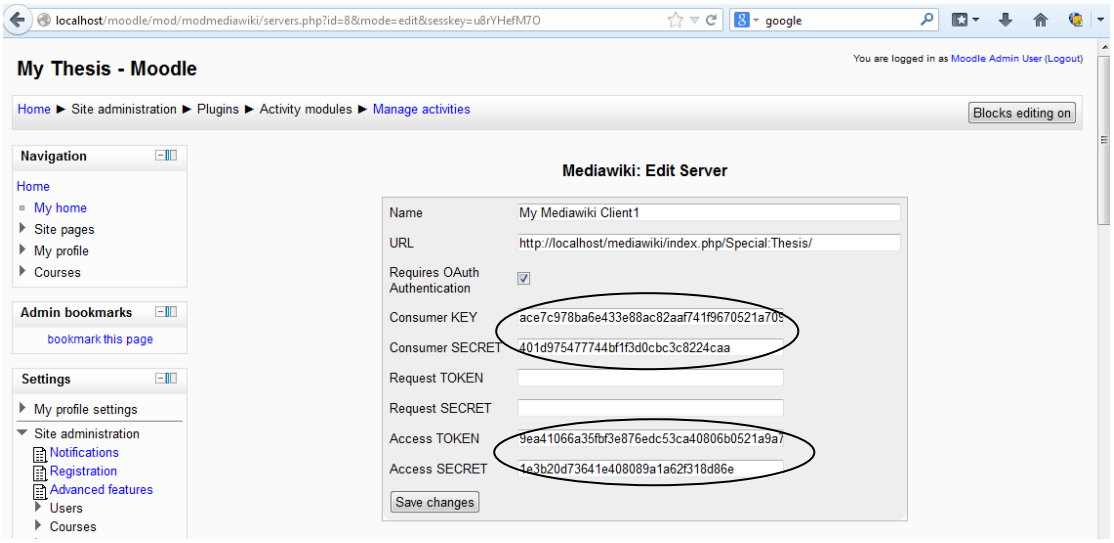
Έτσι, η υπηρεσία MediaWiki RESTFul Server στην πλατφόρμα του MediaWiki αποκρίνεται στο αίτημα αυτό όπως παρουσιάζεται στην Εικ.5.29, όπου και επιλέγουμε το κουμπί «Allow» για την ολοκλήρωση της διαδικασίας προσθήκης δικαιωμάτων εγγραφής (Εικ. 5.30).

Εικ. 5.29 – Απόκριση αιτήματος προσθήκης δικαιωμάτων εγγραφής στο MediaWiki

Όπως στην περίπτωση πιστοποίησης της υπηρεσίας WordPress RESTFul Client του Moodle έτσι και στην πιστοποίηση της υπηρεσίας MediaWiki RESTFul Client του Moodle αποστέλλονται, από την υπηρεσία MediaWiki RESTFul Server του MediaWiki, δυο επιπρόσθετα πιστοποιητικά πρόσβασης και εξουσιοδότησης, τα οποία και αποθηκεύονται στην βάση δεδομένων του Moodle (Εικ.5.31). Έτσι, με την ολοκλήρωση της διαδικασίας αυτής, η υπηρεσία MediaWiki RESTFul Client του Moodle έχει διασυνδεθεί και πιστοποιηθεί με την υπηρεσία MediaWiki RESTFul Server του MediaWiki, η οποία και παρουσιάζεται στην Εικ.5.30 ως Authorized. Έτσι, από εδώ και στο εξής, όταν δημιουργούμε ένα άρθρο στην πλατφόρμα του Moodle, αυτό προστίθεται και προβάλλεται παράλληλα και στην πλατφόρμα του MediaWiki ως μια νέα σελίδα (page).



Εικ. 5.30 Ολοκλήρωση της διαδικασίας εξουσιοδότησης δικαιωμάτων εγγραφής στο MediaWiki



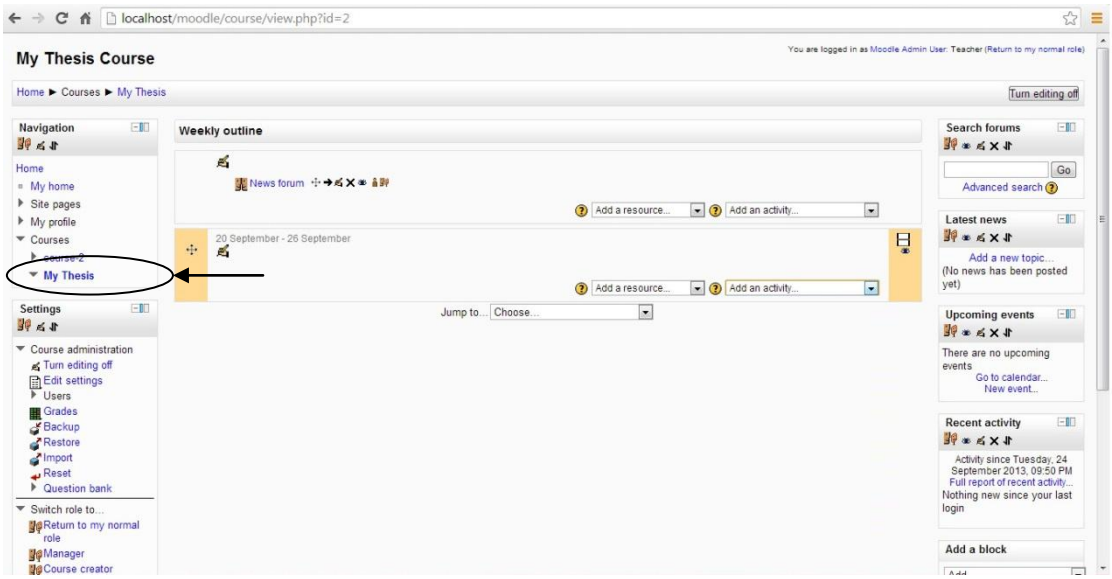
Εικ. 5.31 Προβολή των πιστοποιητικών που ανταλλάχθηκαν μεταξύ των RESTFul υπηρεσιών του Moodle και MediaWiki.

5.2.3 Δημιουργία νέας σελίδας στο MediaWiki μέσω του Moodle Restful client

Για την προσθήκη μιας νέας σελίδας στο MediaWiki μέσω του Moodle Restful client ακολουθούμε την παρακάτω διαδικασία:

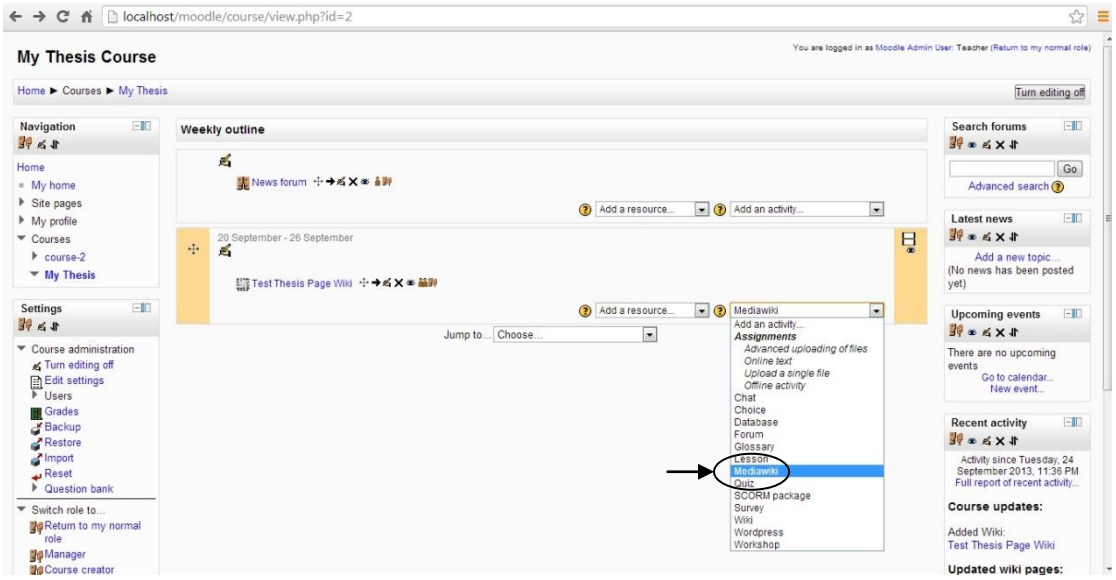
a) Εφόσον έχει ολοκληρωθεί η πιστοποίηση των χρηστών μεταξύ των πλατφορμών του MediaWiki και του Moodle και έχουμε δημιουργήσει το νέο μάθημα (course) με το όνομα

"My Thesis" στην πλατφόρμα του Moodle όπως παρουσιάσαμε στο παραπάνω σενάριο του WordPress, επιλεγούμε το μάθημα "My Thesis" στο Moodle ακολουθώντας την διαδρομή Home > Courses > My Thesis όπως παρουσιάζεται και στην παρακάτω εικόνα (Εικ. 5.32).



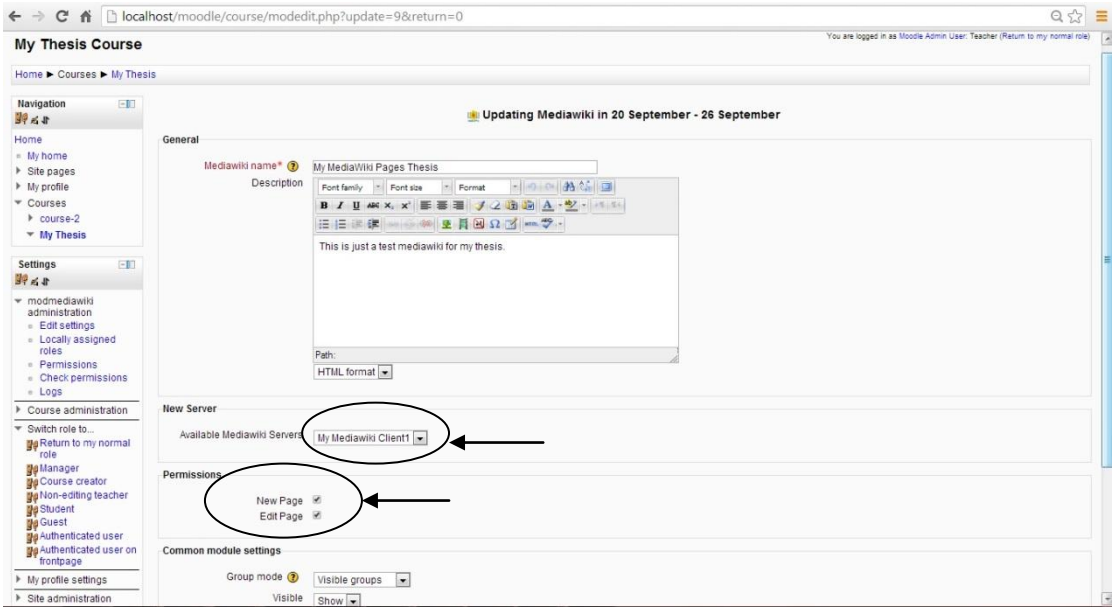
Εικ. 5.32 Επιλογή του μαθήματος "My Thesis" στο Moodle

b) Εφόσον έχουμε επιλέξει το μάθημα "My Thesis" στο πλαίσιο της πρώτης εβδομάδας επιλέγουμε από το "Activity drop-down menu" την επιλογή MediaWiki (Εικ.5.33). Μέσω αυτής της διαδικασίας, θα επιτύχουμε την ενεργοποίηση του MediaWiki activity και την προσθήκη σε αυτό μιας νέας σελίδας, όπου το περιεχόμενο της θα παρουσιάζεται παράλληλα στο MediaWiki και στην πρώτη εβδομάδα του μαθήματος "My Thesis" στην πλατφόρμα του Moodle.



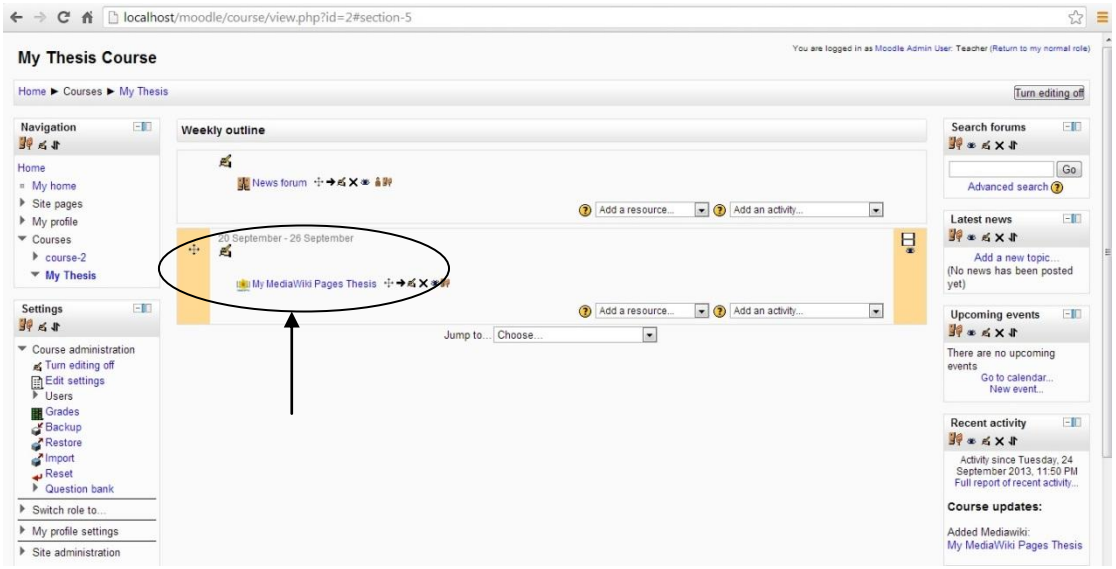
Εικ. 5.33 Ενεργοποίηση του MediaWiki activity στο μάθημα "My Thesis" στο Moodle

c) Για την επιτυχή ενεργοποίηση του MediaWiki activity και για την δημιουργία νέας σελίδας στο μάθημα "My Thesis" στην πλατφόρμα του Moodle, θα πρέπει να συμπληρώσουμε όλα τα προαπαιτούμενα πεδία της φόρμας που εμφανίζονται στην εικόνα (Εικ.5.34). Αξίζει να αναφέρουμε εδώ ότι στα στοιχεία που συμπληρώνουμε στην φόρμα του MediaWiki activity ορίζουμε και τον MediaWiki Restful Client "My Mediawiki Client1" όπου ενεργοποιήσαμε παραπάνω. Δηλαδή, στην πραγματικότητα ορίζουμε τον MediaWiki Sever, όπου θα δημοσιεύεται η νέα μας σελίδα που θα δημιουργήσουμε παρακάτω.
Επιπρόσθετα, στην καρτέλα "Permissions" μπορούμε να ορίσουμε και τα δικαιώματα διαχείρισης των σελίδων που θα έχει ο πιστοποιημένος χρήστης του Moodle.



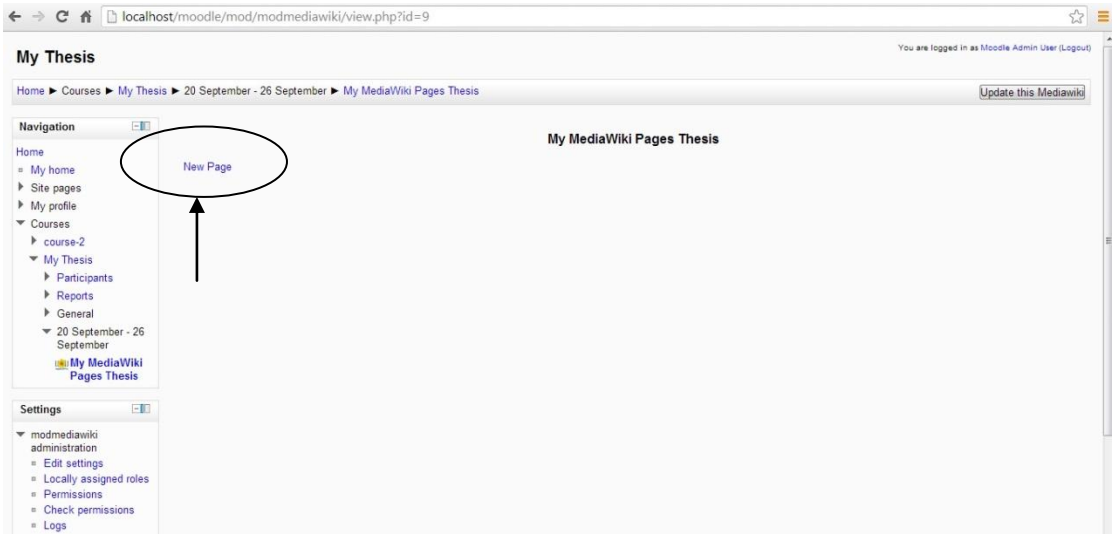
Εικ. 5.34 Ορισμός MediaWiki Restful Client "My Mediawiki Client1" και δικαιωμάτων διαχείρισης των σελίδων για το πιστοποιημένο χρήστη του Moodle

d) Εφόσον έχουμε ενεργοποιήσει επιτυχώς το MediaWiki Activity με το όνομα "My Mediawiki Pages Thesis" στην πρώτη εβδομάδα του μαθήματος "My Thesis" στο Moodle, μπορούμε πλέον να προσθέσουμε και την πρώτη μας σελίδα. Έτσι, για την δημιουργία της νέας σελίδας στο MediaWiki, αρχικά επιλέγουμε τον σύνδεσμο του MediaWiki Activity "My Mediawiki Pages Thesis", όπως παρουσιάζεται στην παρακάτω εικόνα (Εικ. 5.35).



Εικ. 5.35 Επιλογή του MediaWiki Activity "My Mediawiki Pages Thesis" στο Moodle.

e) Εφόσον έχουμε επιλέξει το "My Mediawiki Pages Thesis" activity, στο εσωτερικό μενού του επιλέγουμε, τον σύνδεσμο "New Page". Η διαδικασία αυτή παρουσιάζεται στην παρακάτω εικόνα (Εικ. 5.36).

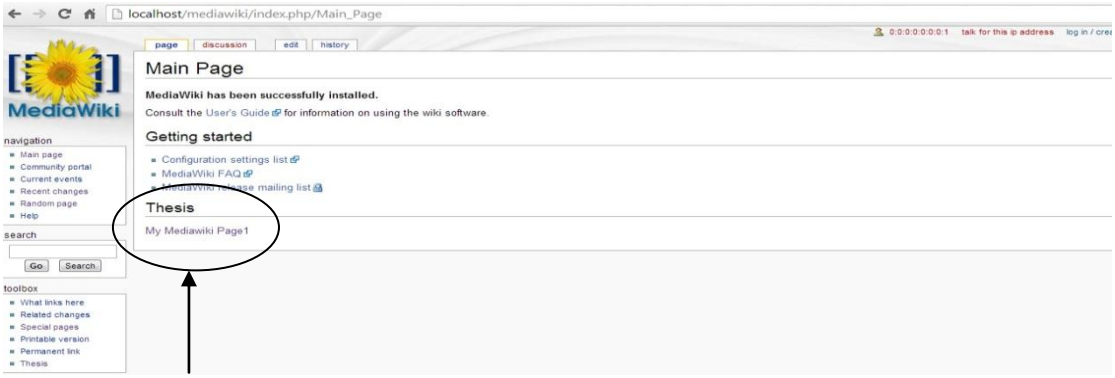


Εικ. 5.36 Επιλογή του συνδέσμου "New Page" στο "My Mediawiki Pages Thesis" activity για την δημιουργία νέας σελίδας

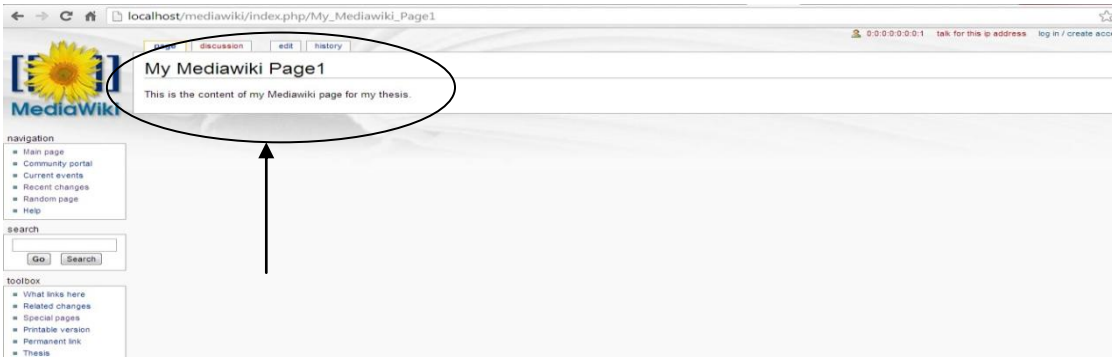
f) Στη συνέχεια συμπληρώνουμε τα στοιχεία της νέας σελίδας που θα παρουσιαστούν στην πλατφόρμα του MediaWiki και στην πλατφόρμα του Moodle στην πρώτη εβδομάδα του μαθήματος "My Thesis" (Εικ.5.37). Για την αποθήκευση της σελίδας επιλέγουμε το κουμπί "Save" της φόρμας.

Εικ. 5.37 Δημιουργία της σελίδας με τίτλο "My Mediawiki Page1" στο MediaWiki

g) Τέλος, η σελίδα που δημιουργήσαμε στο Moodle παρουσιάζεται στο MediaWiki μέσω του MediaWiki Restful API plugin "Thesis", όπως στις παρακάτω εικόνες (Εικ. 5.38 και Εικ. 5.39).



Εικ. 5.38 Προβολή της σελίδας με τίτλο "My Mediawiki Page1" στο MediaWiki



Εικ. 5.39 Προβολή περιεχομένου της σελίδας με τίτλο "My Mediawiki Page1" στο MediaWiki.

Κεφάλαιο 6
Συμπεράσματα και Μελλοντική Έρευνα

Σε αυτό το κεφάλαιο γίνεται μια σύντομη ανασκόπηση στα συμπεράσματα που καταλήξαμε από την τρέχουσα διπλωματική εργασία. Επίσης, περιγράφουμε εν συντομία την μελλοντική έρευνα που μπορεί να γίνει για την επίτευξη καλύτερης σχεδίασης και ανάπτυξης μιας επεκτάσιμης διαδικτυακής RESTful υπηρεσίας.

6.1 Συμπεράσματα

Σε αυτή τη διπλωματική εργασία περιγράφηκε μια μεθοδολογία για την ανάπτυξη μιας αρχιτεκτονικής προσέγγισης τύπου REST με σκοπό την πρόσβαση μιας εκπαιδευτικής διαδικτυακής υπηρεσίας σε ετερογενής διαδικτυακά περιβάλλοντα, επιτρέποντας την ασφαλής διασύνδεση και ανταλλαγή δεδομένων.

Η χρήση της PHP, μαζί με τη χρήση και των κατάλληλων βιβλιοθηκών και προγραμμάτων, κάνει εφικτή τη δημιουργία μιας διαδικτυακής υπηρεσίας με τη μορφή RESTful Web Service. Η διαδικτυακή αυτή υπηρεσία δεν εξαρτάται από το σύστημα του χρήστη, δεν απαιτεί εγκατάσταση επιπλέον λογισμικού και διατίθεται ελεύθερα προς κάθε ενδιαφερόμενο.

Εξαιτίας της εφαρμογής του πρωτοκόλλου OAuth 2.0 και της χρήσης των web APIs διεπαφών όπως είναι το Moodle API, MediaWiki API και το WordPress API οι οποίες επιτρέπουν τη δυναμική και συνδυαστική δημιουργία περιεχομένου, η συγκεκριμένη υλοποίηση επιτυγχάνει την πιστοποίηση των χρηστών για την ασφαλή επικοινωνία μεταξύ των ετερογενών διαδικτυακών πλατφορμών και εκπληρώνει την έννοια της διαλειτουργικότητας.

Ως αποτέλεσμα, οι απαιτήσεις πρόσβασης των υπηρεσιών για ενιαία ανοικτή ενοποίηση, έλεγχο και αποδέσμευση της διεπαφής του χρήστη, πληρούνται από το αρχιτεκτονικό ύφος και τη μεθοδολογία σχεδιασμού.

Συμπεραίνουμε λοιπόν ότι, η προτυποποίηση, η ανοιχτή αρχιτεκτονική, η ευκολία επικοινωνίας συμβατών και ασυμβάτων συστημάτων μεταξύ τους, καθιστούν τις διαδικτυακές RESTful υπηρεσίες ικανές να βοηθήσουν στην βελτίωση της αποτελεσματικότητας σε συστήματα ηλεκτρονικής μάθησης και όχι μόνο..

Είναι αδιαμφισβήτητο, ότι τα τελικά αποτελέσματα αυτής της εργασίας, συνεισφέρουν στην προαγωγή της γνώσης και της εμπειρίας και σε αυτά που συνιστούν τεχνολογικά εργαλεία, προάγοντας έτσι τη καινοτομική δραστηριότητα μέσω προηγμένων μορφών υπηρεσιών διαδικτύου.

6.2 Μελλοντική Έρευνα

Η προτεινόμενη αρχιτεκτονική που σχεδιάσαμε και υλοποιήσαμε στην εν λόγω εργασία, προσφέρει δυνατότητες επέκτασης εμπλουτίζοντας την με περισσότερες λειτουργικές απαιτήσεις στοχεύοντας στην βελτίωση της επικοινωνίας και της συνεργασίας τόσο των εμπλεκόμενων διαδικτυακών πλατφορμών όσο και των χρηστών.

Πιο συγκεκριμένα, μια ενδεχόμενη προσπάθεια επέκτασης αυτής, είναι η δυνατότητα επίτευξης μιας αμφίδρομης επικοινωνίας ανταλλαγής δεδομένων ανάμεσα σε ένα Consumer του Moodle και στο Service Provider της εφαρμογής, MediaWiki ή WordPress.

Λόγω του ότι ασχοληθήκαμε με την ενσωμάτωση της RESTful υπηρεσίας σε μερικές μόνο από τις λειτουργίες που αυτή μπορεί να προσφέρει σε ένα Consumer του Moodle, θα μπορούσαμε να ενσωματώσουμε όλη τη λίστα των λειτουργιών που καλύπτει τον πλήρη κύκλο ζωής της υπηρεσίας,

Τέλος, θα μπορούσε να μεριμνηθεί και να ενσωματωθεί στην υπηρεσία η δυνατότητα μεταφοράς δικαιωμάτων ομάδων χρηστών με τα αντίστοιχα δικαιώματα από την μια πλατφόρμα στην άλλη.

Βιβλιογραφία

[01] J.M. Dodero, E. Ghiglione (2008). "ReST-based web access to learning design services", IEEE Transactions on Learning Technologies vol.1 No.3 (July-Septembrer 2008) pp. 190–195.

[02] Paul Anderson (2007). "What is Web 2.0 ? Ideas, technologies and implications for education", JISC Technology & Standarts Watch (Feb 2007).

[03] Ivan Madjarov and Omar Boucelma (2006). "Data and Application Integration in Learning Content Management Systems: A Web Services Approach", Springer (2006), pp. 272 – 286.

[04] XML, http://www.informit.com/articles/article.aspx?p=31076&seqNum=2, (accessed on December 22, 2012)

[05] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pål Krogdahl, Min Luo, Tony Newling (2004). "Patterns: Service-Oriented Architectureand Web Services" Redbooks, First Edition (April 2004).

[06] Developer.com (2003). "Web Services Tutorial: Understanding XML and XML Schema - Part 1, April 23, 2003. Available online: http://www.developer.com/services/print.php/10928_2195981_2 (accessed on April 12, 2013).

[07] SOAP, http://www.w3schools.com/soap/soap_syntax.asp, (accessed on December19, 2011).

[08] http://www.w3.org/2001/03/14-annotated-WSDL-examples.html (accessed on August 20, 2013).

[09] O'REILLY Online Catalog "Java Web Services". Available online: http://oreilly.com/catalog/javawebserv/chapter/ch06.html.

[10] http://www.cs.colorado.edu/~kena/classes/7818/f08/lectures/lecture_4_uddi.pdf (accessed on June 7, 2013).

[11] http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/74bae690-0201-0010-71a5-9da49f4a53e2?QuickLink=index&overridelayout=true&5003637731092 (accessed on January 12, 2013)

[12] W3C Note (2001). "Web Services Description Language (WSDL) 1.1", 15 March 2001. Available online: http://www.w3.org/TR/wsdl#_introduction, (accessed on January 12, 2013).

[13] http://www.cs.colorado.edu/~kena/classes/7818/f06/lectures/WSDL.pdf (accessed on January 12, 2013)

[14] http://www.fdi.ucm.es/profesor/jjruz/websi/bibliografia/wsdl1.pdf (accessed on November 8, 2012)

[15] http://www.ibm.com/developerworks/webservices/newto/service.html (accessed on January 14, 2013)

[16] Heather Kreger - IBM Software Group (2001). "Web Services Conceptual Architecture (WSCA 1.0)", May 2001.

[17] http://en.wikipedia.org/wiki/Representational_state_transfer (accessed on April 3, 2013)

[18] Antonio Goncalves (2009). "Beginning Java EE 6 Platform with GlassFish 3 - From Novice to Professional", Chapter 15, 2009, pp 429-462.

[19] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accessed on February 2, 2013)

[20] Ajaxonomy (2008)."Web Services, Part 1: SOAP vs. REST". Available online: http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest, (accessed on January 11, 2012).

[21] Mark Masse (2011). "REST API. Designing Consistent Restful Web Service Interfaces". O'Reilly ISBN:978-1-449-31050-9

[22] http://www.makeuseof.com/tag/api-good-technology-explained/ (accessed on May 13, 2013)

[23] http://en.wikipedia.org/wiki/Application_programming_interface (accessed on May 15, 2013)

[24] https://blog.apigee.com/detail/what_is_an_API_and_how_different_than_Web_service (accessed on May 15, 2013)

[25] EdgeCast Networks (2011). "Web Services REST API", version 0.1 (3/3/2011)

[26] http://tools.ietf.org/html/rfc6749#section-1.1 (accessed on June 28, 2013)

[27] http://www.mediawiki.org/wiki/Auth_systems/OAuth/Design (accessed on June 30, 2013)

Αρκτικόλεξα - Συντομογραφίες

REST	Representational State Protocol
W3C	World Wide Web Consortium
SGML	Standard Generalized Markup Language
DTD	Document Type Definition
HTTP	Hypertext Transfer Protocol
SOAP	Simple Object Access Protocol
RPC's	Remote Procedure Calls
UDDI	Universal Description, Discovery and Integration
API	Application Programming Interface
WSDL	Web Service Dercription Language
URI	Uniform Resource Indentifier
DCOM	Distributed Component Object Model
CORBA	Common Object Request Broker Architecture
IBM	International Business Machines Corporation
SOA	Service Oriented Architecture
SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
WSFL	Web Services Flow Language
XHTML	Extensible Hypertext Markup Language
JSON	JavaScript Object Notation
RDF	Resource Description Framework
IETF	Internet Engineering Task Force

Παράρτημα Α
Ενδεικτικά Παραδείγματα

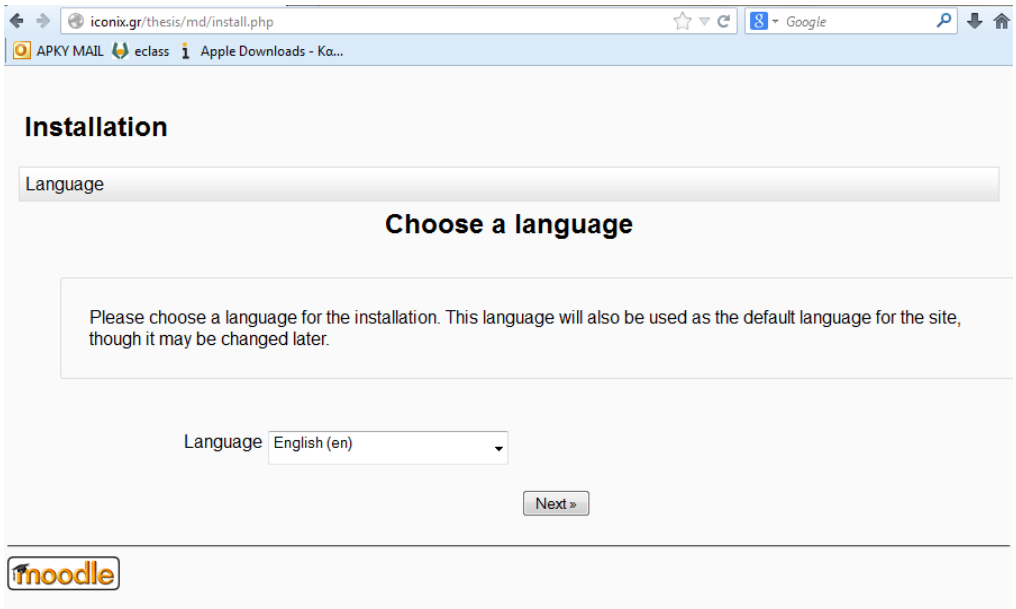
Α.1 Σύνταξη WSDL Εγγράφου

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <definitions name="FooSample"
3        targetNamespace=http://tempuri.org/wsdl/
4        xmlns:wsdlns="http://tempuri.org/wsdl/"
5        xmlns:typens="http://tempuri.org/xsd"
6        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
8        xmlns:stk="http://schemas.microsoft.com/soap-toolkit/wsdl-extension"
9        xmlns="http://schemas.xmlsoap.org/wsdl/">
10 <types>
11       <schema targetNamespace="http://tempuri.org/xsd"
12       xmlns="http://www.w3.org/2001/XMLSchema"
13       xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
14       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
15       elementFormDefault="qualified" >
16       </schema>
17 </types>
```

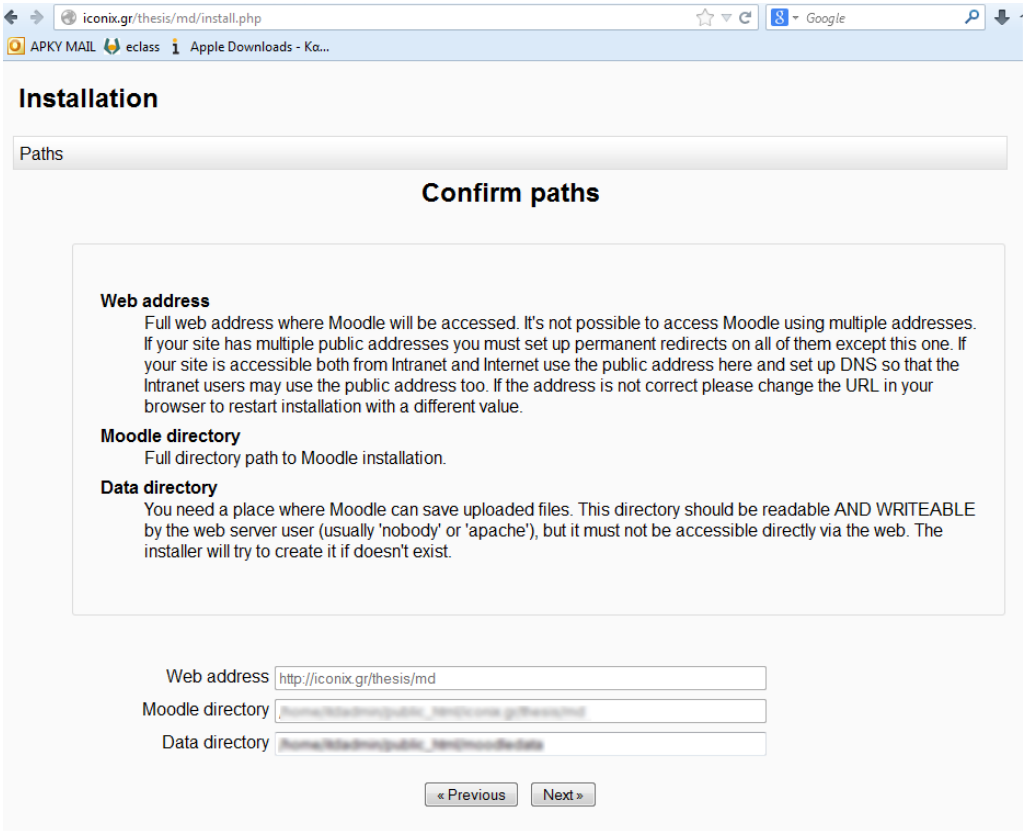
```
<message name="Simple.foo">
      <part name="arg" type="xsd:int"/>
      </message>
      <message name="Simple.fooResponse">
      <part name="result" type="xsd:int"/>
</message>
<portType name="SimplePortType">
      <operation name="foo" parameterOrder="arg" >
      <input message="wsdlns:Simple.foo"/>
      <output message="wsdlns:Simple.fooResponse"/>
      </operation>
</portType>
<binding name="SimpleBinding" type="wsdlns:SimplePortType">
      <stk:binding preferredEncoding="UTF-8" />
      <soap:binding style="rpc"
       transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="foo">
        <soap:operation
        soapAction="http://tempuri.org/action/Simple.foo"/>
        <input>
            <soap:body use="encoded"
            namespace="http://tempuri.org/message/"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </input>
          <output>
            <soap:body use="encoded"
            namespace="http://tempuri.org/message/"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
          </output>
      </operation>
</binding>
<service name="FOOSAMPLEService">
      <port name="SimplePort" binding="wsdlns:SimpleBinding">
      <soap:address location="http://carlos:8080/FooSample/FooSample.asp"/>
      </port>
</service>
</definitions>
```

Παράρτημα Β
Εγκαταστάσεις Πλατφορμών

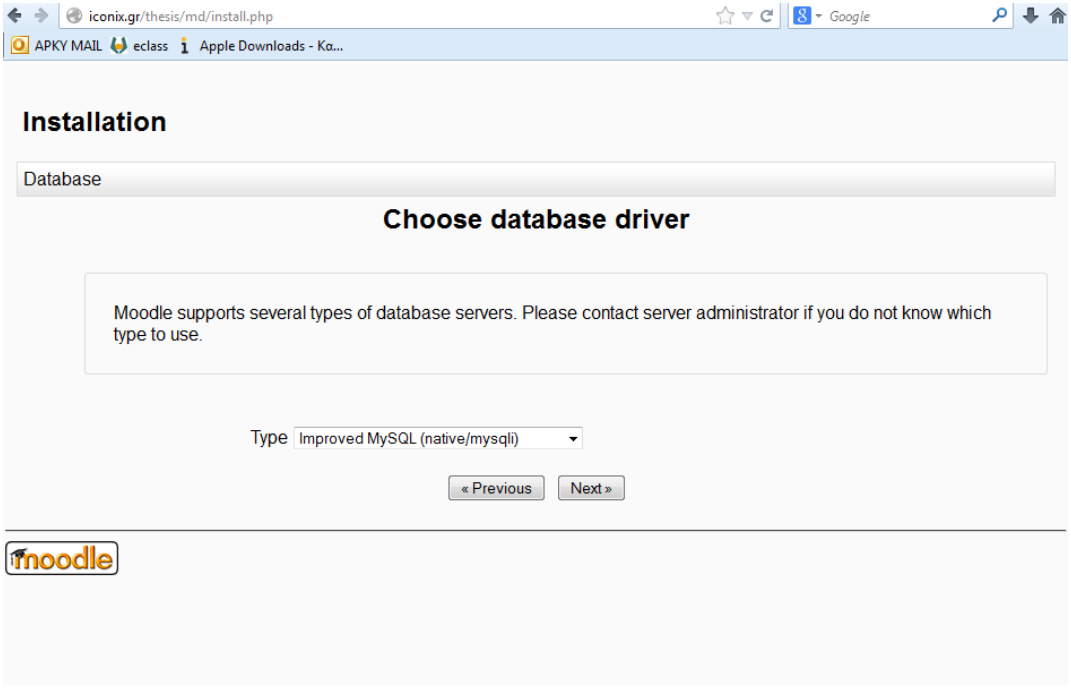
Β.1 Εγκατάσταση Moodle



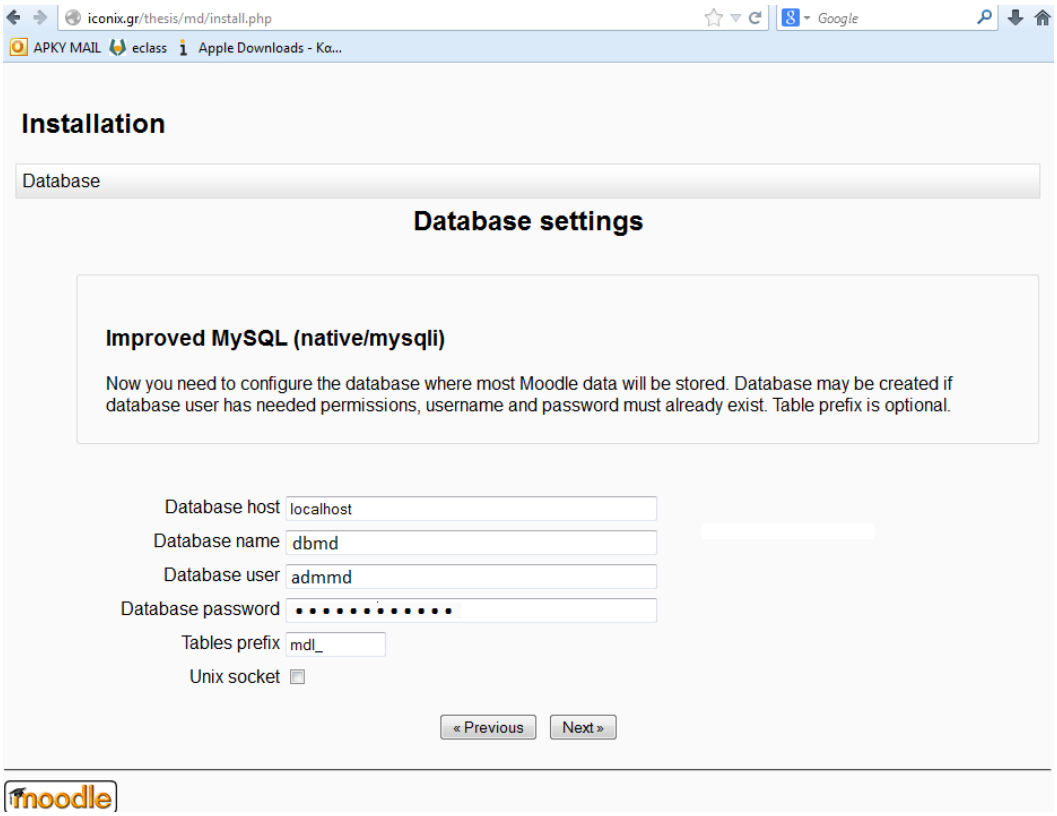
Εικ.1 Επιλογή της γλώσσας Εγκατάστασης του Moodle



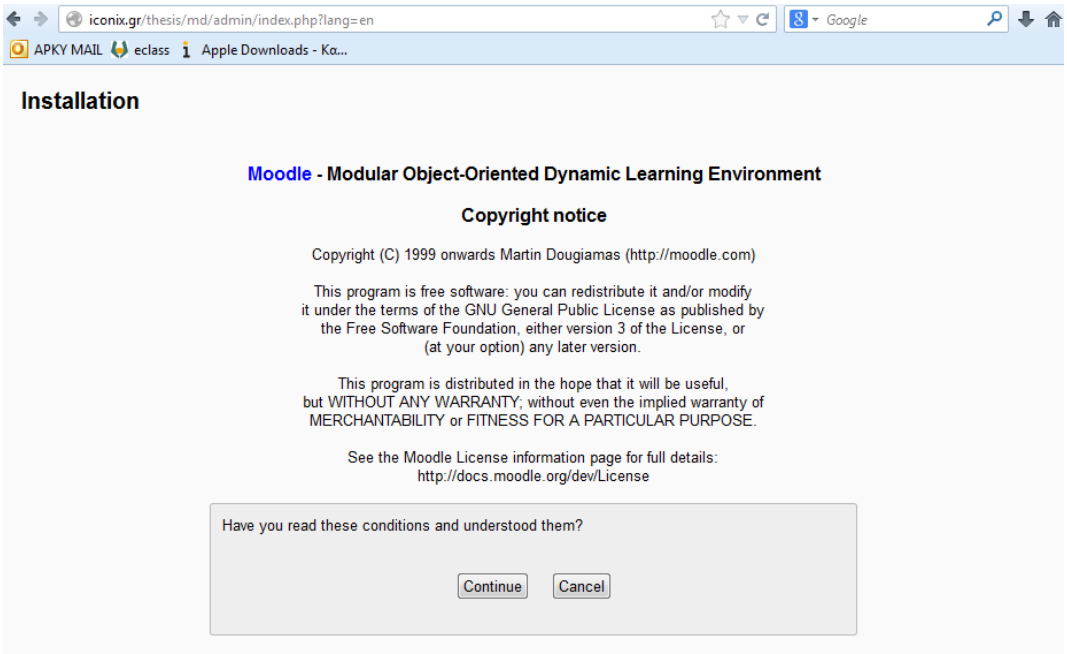
Εικ.2 Ορισμός διεύθυνσης πρόσβασης, και εγκατάστασης του Moodle



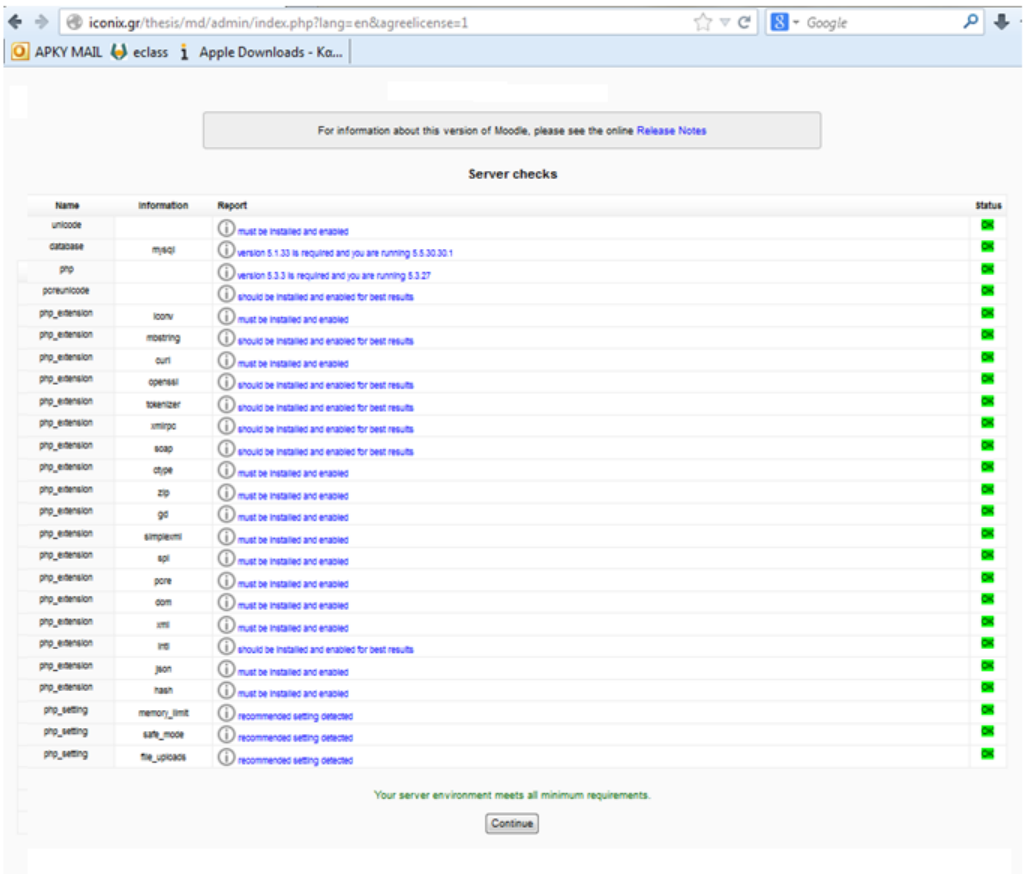
Εικ.3 Επιλογή του τύπου της Β.Δ που θα χρησιμοποιηθεί στο Moodle



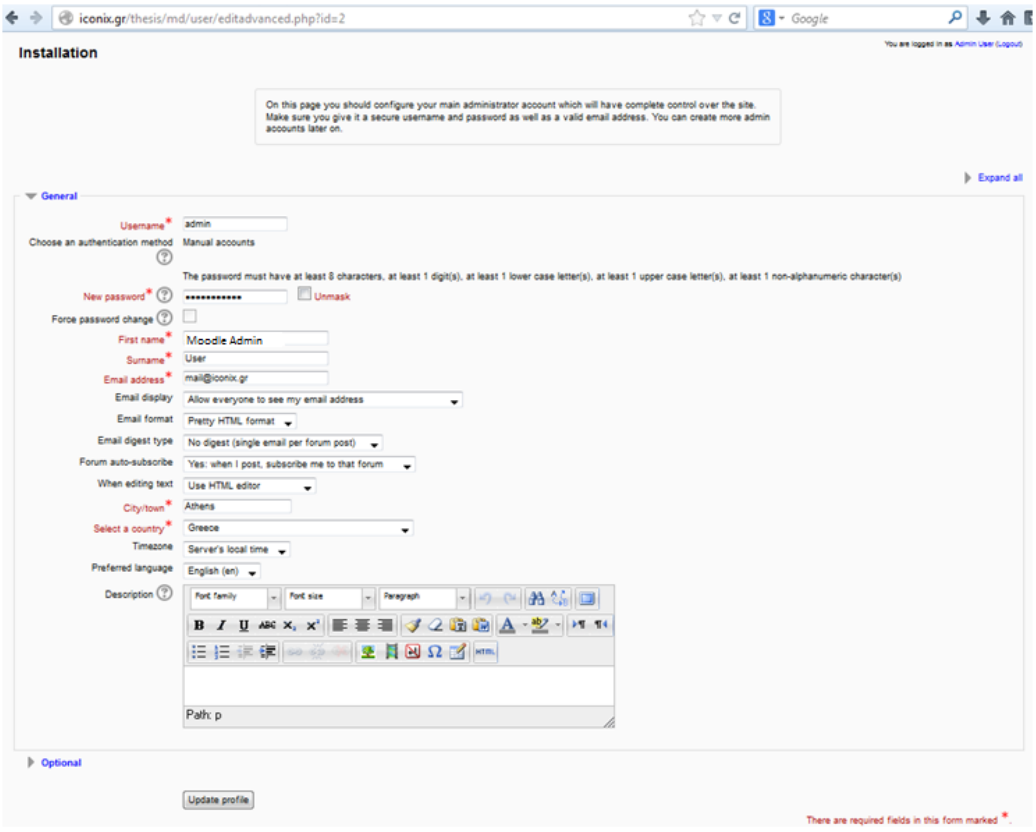
Εικ.4 Εισαγωγή στοιχείων για πρόσβαση στη Β.Δ. του Moodle



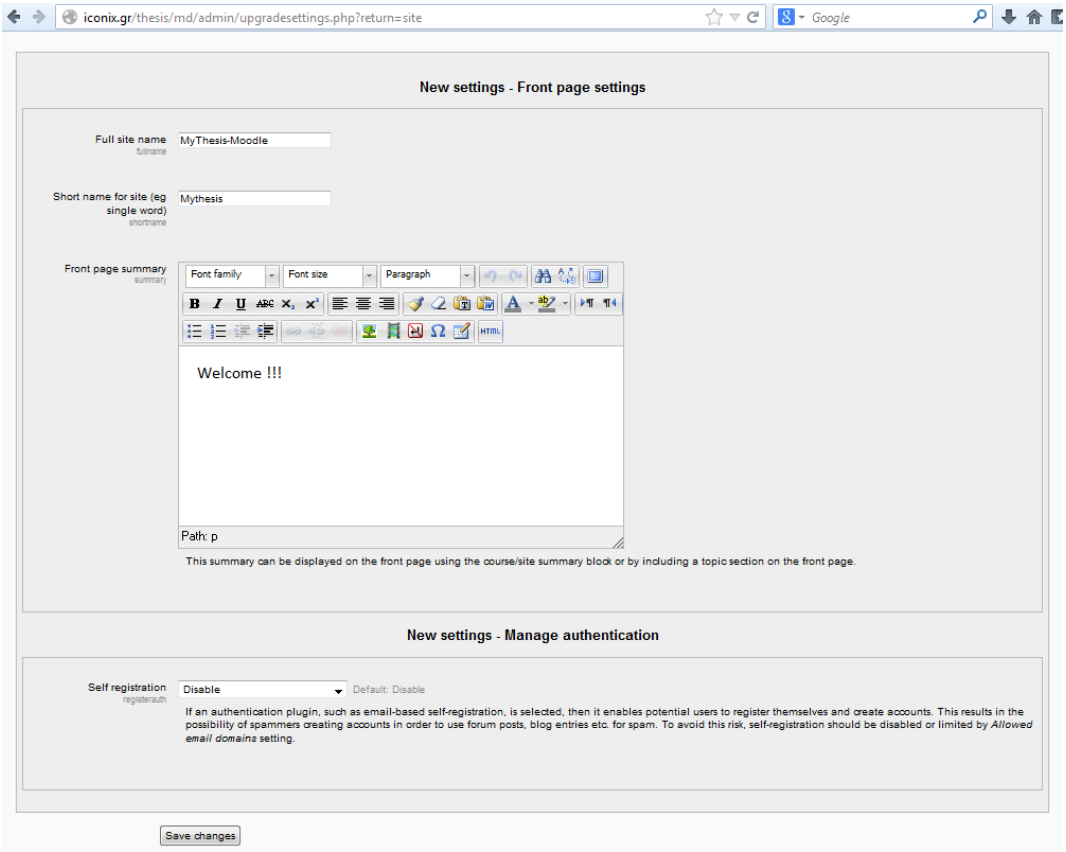
Εικ.5 Αποδοχή της εγκατάστασης του Moodle



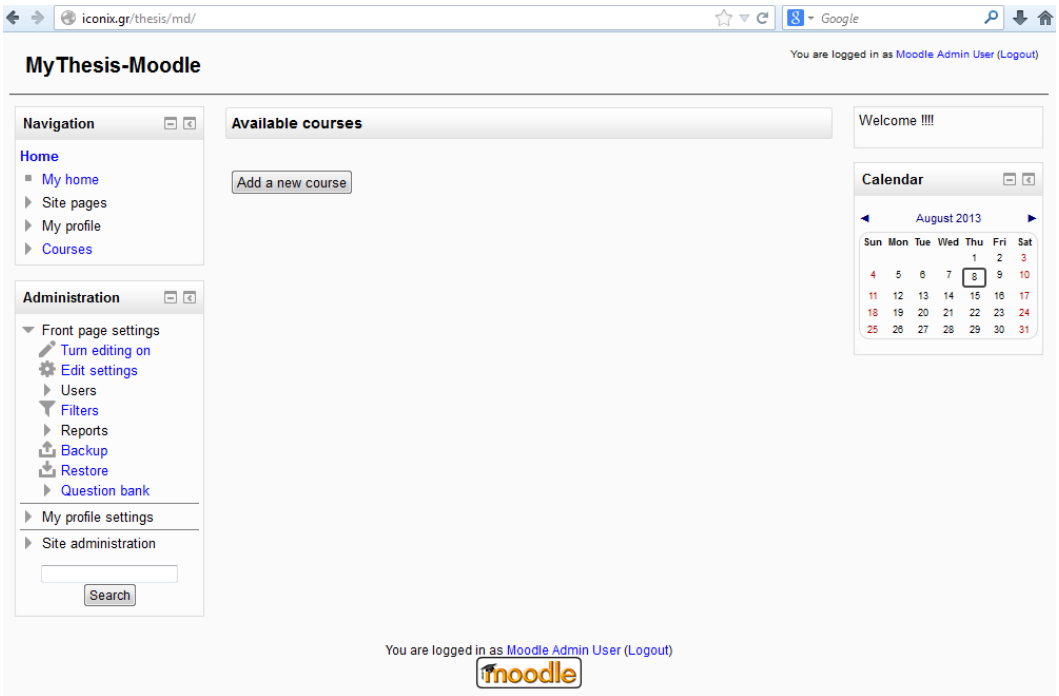
Εικ.6 Έλεγχος ύπαρξης ρουτίνων php στο Moodle



Εικ.7 Εισαγωγή στοιχείων χρήστη για τη δημιουργία λογαριασμού στο Moodle

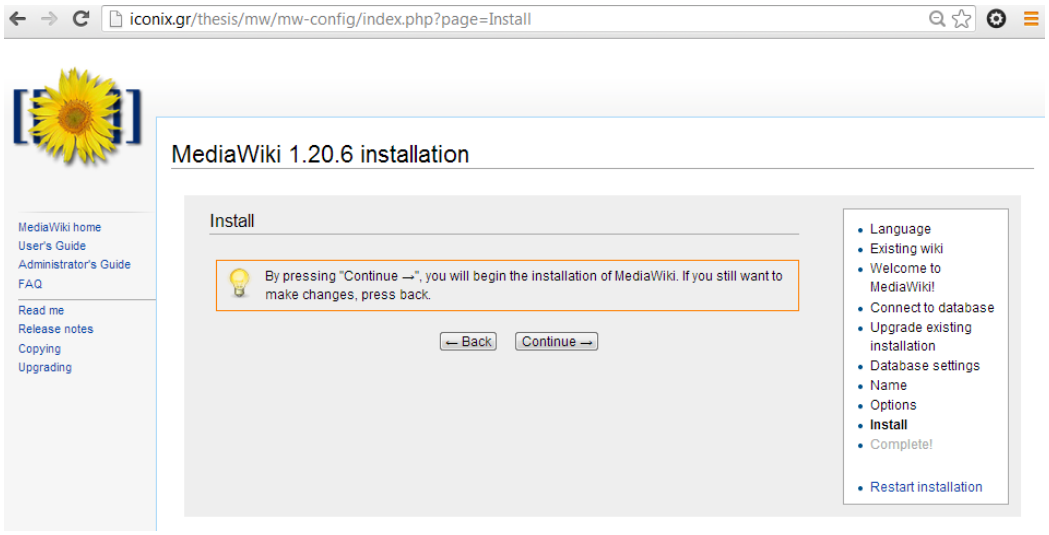


Εικ.8 Καθορισμός στοιχείων εμφάνισης της πρώτης σελίδας του Moodle

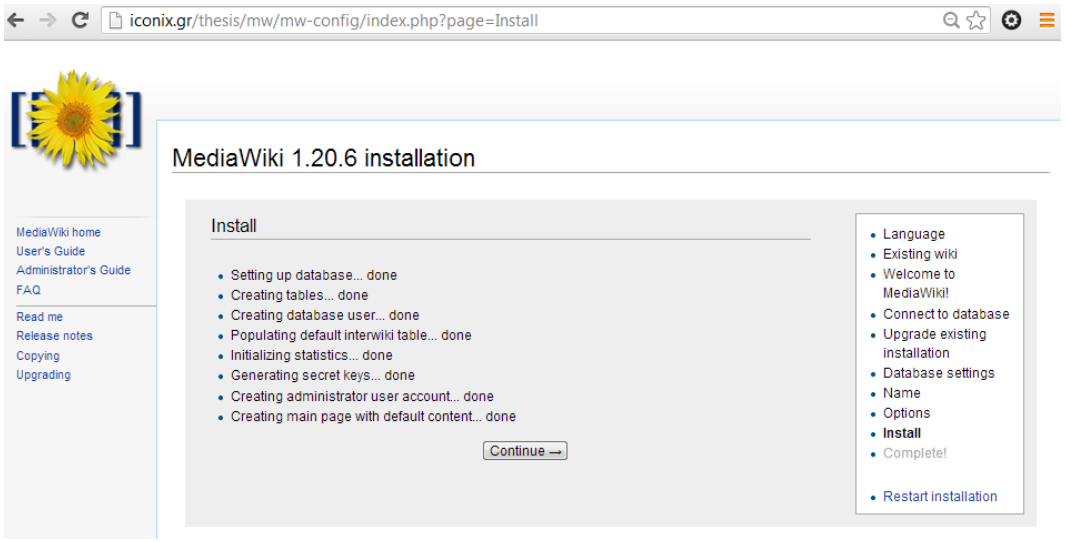


Εικ.9 Είσοδος στην αρχική σελίδα του Moodle

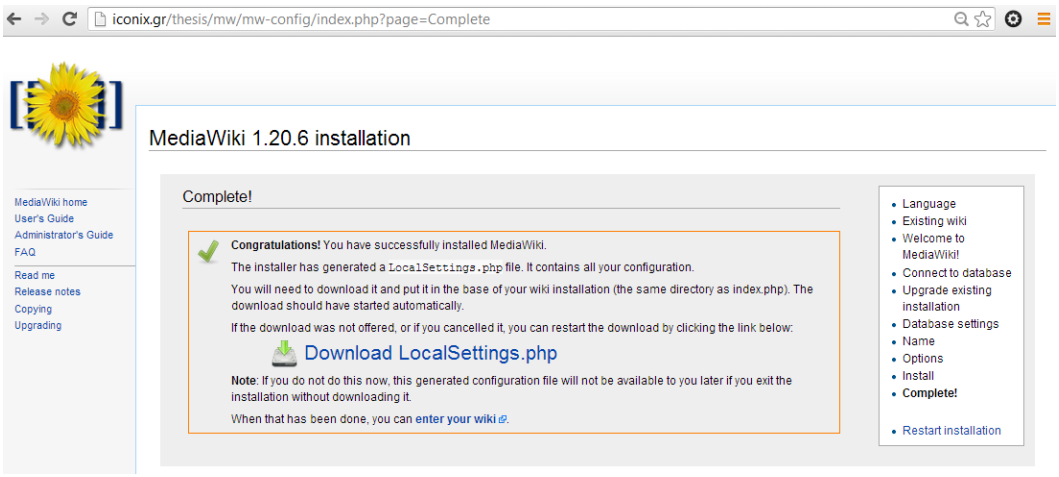
B.2 Εγκατάσταση MediaWiki



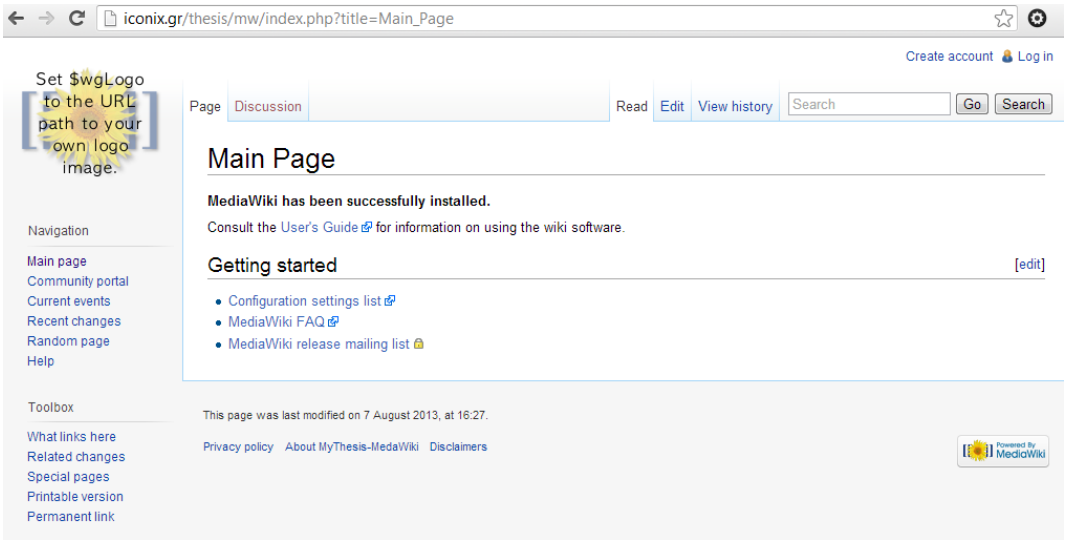
Εικ.10 Έναρξη Εγκατάστασης του MediaWiki



Εικ.11 Αναφορά κατάστασης στοιχείων που εγκαταστάθηκαν

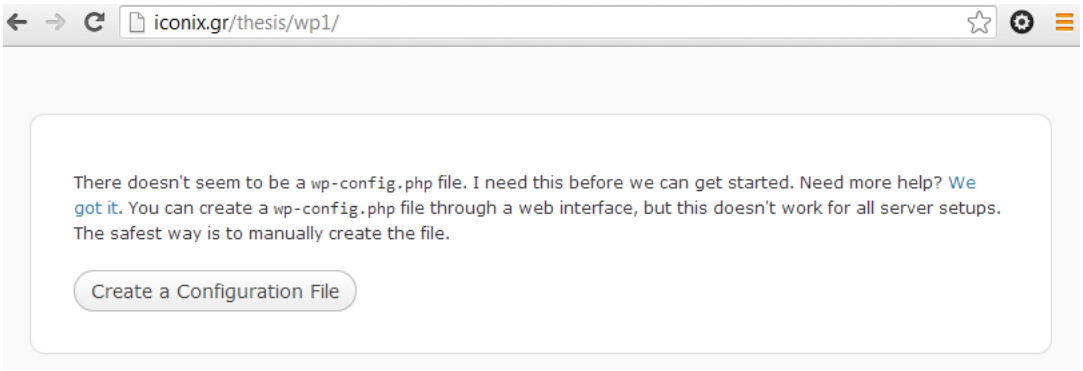


Εικ.12 Ολοκλήρωση Εγκατάστασης του MediaWiki

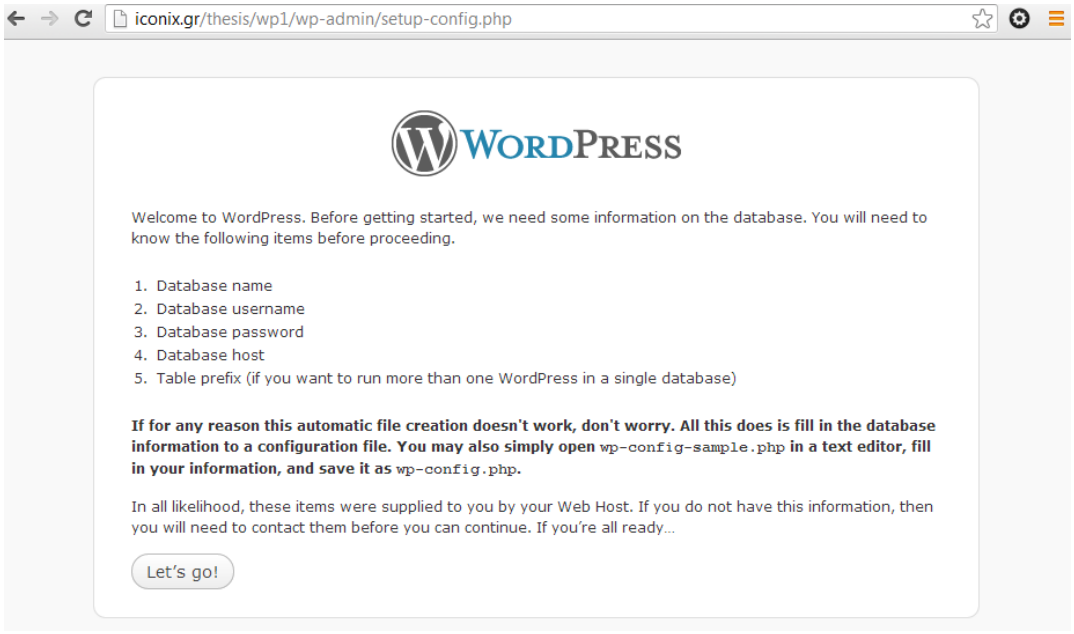


Εικ.13 Έμφάνιση της πρώτης σελίδας του MediaWiki

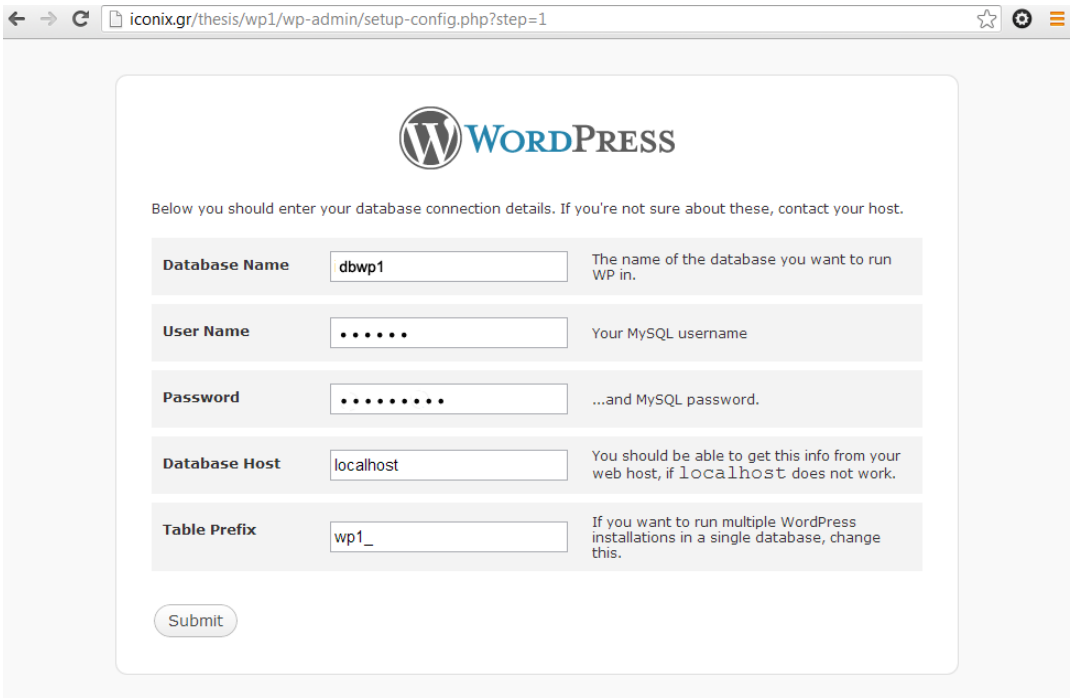
Β.3 Εγκατάσταση WordPress



Εικ.14 Έναρξη εγκατάστασης WordPress

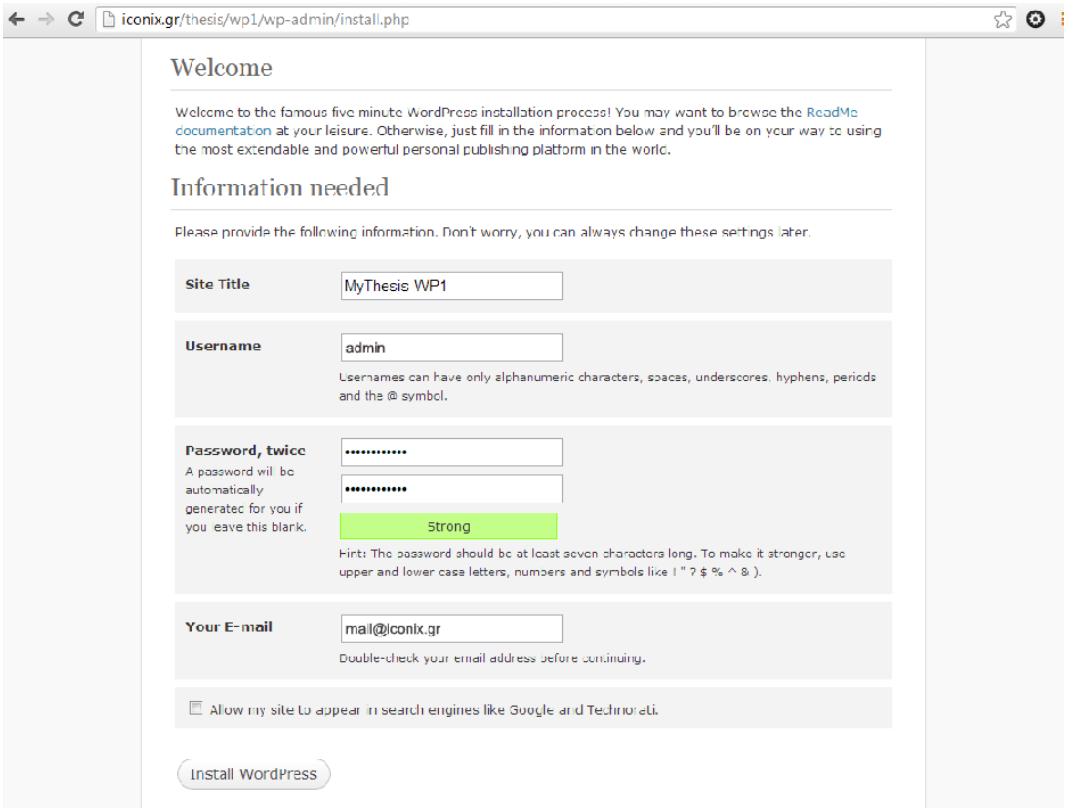


Εικ.14 Ενημερωτικό παράθυρο για τα στοιχεία της Β.Δ που θα ζητηθούν στο επόμενο βήμα της εγκατάστασης του WordPress

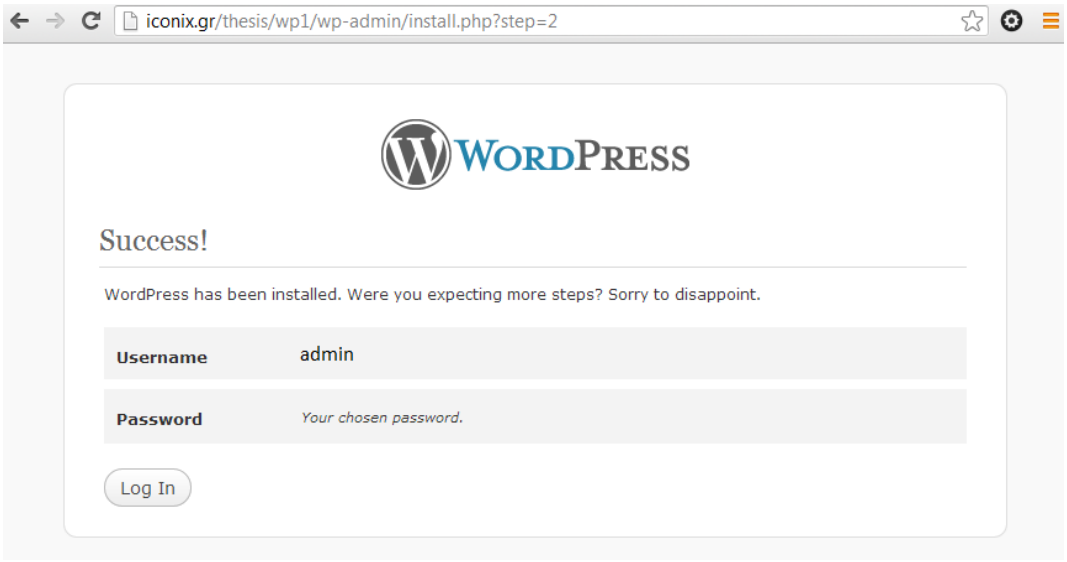


Εικ.15 Εισαγωγή στοιχείων για πρόσβαση στη Β.Δ του WordPress

Εικ.16 Εκτέλεση εγκατάστασης του WordPress



Εικ.17 Εισαγωγή στοιχείων χρήστη για τη πρόσβαση στο WordPress



Εικ.18 Ολοκλήρωση Εγκατάστασης του WordPress

Εικ.19 Αρχική σελίδα του WordPress

Οι πλατφόρμες που εγκαταστάθηκαν για την υλοποίηση της εφαρμογής είναι οι εξής :

- Moodle Έκδοση 2.0.1.0
- MediaWiki Έκδοση 1.16.1
- WordPress Έκδοση 3.0