ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ONLINE ΣΥΣΤΗΜΑΤΟΣ ΚΑΤΑΓΡΑΦΗΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΕΞΟΠΛΙΣΜΟΥ ΠΑΝΕΠΙΣΤΗΜΙΟΥ

Πτυχιακή εργασία

ΒΛΑΧΟΣ ΗΛΙΑΣ-ΣΠΥΡΙΔΩΝ

Αθήνα, 2017

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Δημητρακόπουλος Γεώργιος

Επίκουρος Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο

Νικολαΐδου Μάρα

Καθηγήτρια, Πληροφορική και Τηλεματική, Χαροκόπειο

Βαρλάμης Ηρακλής

Επίκουρος Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο

Ο/~~Η~~ ΒΛΑΧΟΣ ΗΛΙΑΣ-ΣΠΥΡΙΔΩΝ

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Αφιερώνω αυτή την πτυχιακή σε όλους αυτούς που προσπαθούν. Η ουσία είναι πόσο γεμάτος νιώθεις τώρα που τελειώνεις.

ΕΥΧΑΡΙΣΤΙΕΣ

Ένας κύκλος κλείνει, το φοιτητικό ταξίδι τελειώνει. Συνολικά ίσως η ωραιότερη περίοδος της έως τώρα ζωής μου. Για όλα αυτά που έζησα ως φοιτητής, ευχαριστώ την οικογένειά μου. Τους γονείς, την αδερφή μου, τη θεία και το θείο μου που ήταν και είναι εκεί για ότι χρειαστώ. Ευχαριστώ τους φίλους που έκανα ως φοιτητής για την βοήθεια καθόλη τη διάρκεια των σπουδών, αλλά κυρίως για τις στιγμές που ζούμε.

Ευχαριστώ το τμήμα Πληροφορικής και Τηλεματικής καθώς και το Χαροκόπειο, αφού από την ημέρα της εγγραφής φάνηκε ότι θα είναι διαφορετικό από τα υπόλοιπα εκπαιδευτικά ιδρύματα. Όλο το προσωπικό το οποίο απαρτίζει αυτό το τμήμα με έκανε να το αγαπήσω. Ελπίζω με τη βοήθεια του προσωπικού αλλά κυρίως των φοιτητών το τμήμα να διατηρήσει την ταυτότητα και τα χαρακτηριστικά που το κάνουν ξεχωριστό.

Περίληψη

Σήμερα η χρήση web-based εφαρμογών είναι τόσο διαδεδομένη ώστε σε πολλές περιπτώσεις να τείνουν να αντικαταστήσουν τις desktop-based. Η παρούσα πτυχιακή έχει ως αντικείμενο, τον σχεδιασμό και ανάπτυξη ενός συστήματος καταγραφής και διαχείρισης του ηλεκτρονικού εξοπλισμού του Πανεπιστημίου. Η web-based εφαρμογή που αναπτύχθηκε για το σκοπό αυτό, χωρίζεται σε τρία βασικά μέρη. Στη διαχείριση συσκευών, χρηστών και διευθύνσεων. Πρόσβαση σε αυτή θα έχουν οι διαχειριστές δικτύου και τα μέλη της ομάδας τεχνικής υποστήριξης του Πανεπιστημίου μας. Για το λόγο αυτό η εφαρμογή υποστηρίζει δύο διαφορετικούς ρόλους χρηστών μετά τη σύνδεση σε αυτή. Η εφαρμογή σχεδιάστηκε με γνώμονα τις ανάγκες ή ελλείψεις που διαπιστώθηκαν κατά την περίοδο συμμετοχής μου στην ομάδα τεχνικής υποστήριξης και ο κώδικας δομήθηκε με σκοπό να είναι κατανοητός και εύκολα επεκτάσιμος στο μέλλον. Επιπλέον έγινε προσπάθεια καταχώρησης σημαντικού όγκου δεδομένων ώστε να είναι πραγματικά χρήσιμη.

Abstract

Nowadays web applications are so popular that they tend to replace desktop applications. The purpose of this thesis is the design and development of a system for managing the computer equipment of our University. The web application developed for this purpose can be divided in to three main parts. Management of devices, users and ip addresses. This application will be available to network administrators and the dsupport team of our University. For this reason the app supports two different administrative roles. The application was designed to resolve problems that occurred during my time as a member of dsupport team. The code is structured in a way to be easily understandable and expandable in the future. Finally effort was made to insert as much data as possible into the database so the application is truly useful.

Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1: Εισαγωγή

1.1 Προσδιορισμός του προβλήματος

Σήμερα το λογισμικό αναλαμβάνει να διεκπεραιώσει πολλά καθήκοντα για τα οποία θα έπρεπε να ξοδέψουμε χρόνο και ενέργεια. Έτσι προτιμούμε την κατασκευή λογισμικού που θα μας απαλλάξει από χρονοβόρες διαδικασίες και ειδικά από διαδικασίες που απαιτούν το χειρισμό δεδομένων τα οποία δεν είμαστε σε θέση να κρατάμε στη μνήμη μας συνεχώς. Ένα τέτοιο παράδειγμα είναι το σύνολο του ηλεκτρονικού εξοπλισμού του πανεπιστημίου μας. Συγκεκριμένα ο ηλεκτρονικός εξοπλισμός του πανεπιστημίου μας, αποτελείται κυρίως από Ηλεκτρονικούς Υπολογιστές, σταθερούς και φορητούς, routers και hub, καθώς και εκτυπωτές. Ο εξοπλισμός αυτός, σε ποσοστό μεγαλύτερου του 97% διαθέτει σύνδεση σε κάποιο δίκτυο.

Αναμενόμενο είναι να μην μπορούμε να διατηρούμε στη μνήμη μας τη σχέση κάθε συσκευής με το πανεπιστήμιο, τα χαρακτηριστικά της και συγκεκριμένα την τοποθεσία εντός του πανεπιστημίου στην οποία βρίσκεται, καθώς και σε ποιον/ποια είναι καταχωρημένη. Επιπλέον η διατήρηση αρχείων με αυτή την πληροφορία θα ήταν μια σύνθετη διαδικασία, καθώς θα απαιτούταν η συνεργασία πολλών προσώπων με κοινή χρήση αρχείων. Βέβαια μερικώς θα μπορούσε να υπάρχει έστω σα διαδικασία με χρήση διαθέσιμων προϊόντων όπως Google Docs.

1.2 Κίνητρο

Στη διάρκεια της απασχόλησής μου, στο Χαροκόπειο πανεπιστήμιο, ως μέλος της ομάδας τεχνικής υποστήριξης από το 2014 έως το 2016, αντιμετωπίσαμε προβλήματα τα οποία οφείλονται στην έλλειψη πληροφοριών. Συγκεκριμένα πληροφοριών κυρίως για τη φυσική θέση των συσκευών, τους κατόχους τους και τις δικτυακές διευθύνσεις τους (IP διευθύνσεις). Πολλές φορές τα στοιχεία που διαθέταμε ήταν μη ενημερωμένα με αποτέλεσμα να δημιουργείται μπέρδεμα στην προσπάθεια επίλυσης προβλημάτων. Έτσι αυτή η πτυχιακή εργασία στοχεύει στην επίλυση αυτού του προβλήματος με την ανάπτυξη λογισμικού που θα ενσωματώνει τις απαιτούμενες πληροφορίες και λειτουργίες, σε ένα σύστημα.

1.3 Σκοπός

Ο σκοπός της παρούσας εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός συστήματος καταγραφής και διαχείρισης εξοπλισμού, το οποίο θα παρέχεται στον χρήστη μέσω του Browser. Ο χρήστης θα έχει τη δυνατότητα, προβολής και διαχείρισης του συστήματος, κάθε στιγμή, ανεξάρτητα από την τοποθεσία του. Πρόσβαση στο σύστημα αυτό θα έχουν τα μέλη του Χαροκοπείου Πανεπιστημίου έχοντας όμως διαφορετικές δυνατότητες, ανάλογα με τον ρόλο που διαθέτουν. Οι ρόλοι των χρηστών διακρίνονται σε: Χρήστης επιπέδου 1 (Admin), Χρήστης επιπέδου 2 (Expert) και Χρήστης επιπέδου 3 (User). Με αυτό τον τρόπο κάθε χρήστης

θα έχει πρόσβαση σε διαφορετικές πληροφορίες και λειτουργίες, με σκοπό την αποφυγή λαθών και την καλύτερη οργάνωση. Έμφαση θα δοθεί στο σχεδιασμό του συστήματος, με τέτοιο τρόπο ώστε να είναι εύχρηστο για τον απλό χρήστη, αλλά να παρέχει όλες τις λειτουργίες και πληροφορίες που χρειάζεται ο διαχειριστής.

1.4 Δομή

- Κεφάλαιο 2: Στο 2^{ο} κεφάλαιο γίνεται μια παρουσίαση των Πληροφοριακών Συστημάτων, ώστε να γίνουν κατανοητά τα βασικά χαρακτηριστικά τους.
- Κεφάλαιο 3: Στο 3^{ο} κεφάλαιο γίνεται ανάλυση των web εφαρμογών καθώς και των εννοιών και εργαλείων που απαιτούνται για την κατανόηση τους.

Κεφάλαιο 2: Πληροφοριακά Συστήματα

Στο κεφάλαιο 2 γίνεται μια παρουσίαση των Συστημάτων Καταγραφής Δεδομένων και Πληροφοριακών Συστημάτων. Επιπλέον γίνεται ιστορική ανασκόπηση των Πληροφοριακών Συστημάτων και του τρόπου αποθήκευσης δεδομένων, ενώ περιγράφεται ο τρόπος λειτουργίας τους σήμερα. Τέλος αναφέρονται παραδείγματα, τέτοιων συστημάτων στη χώρα μας.

2.1 Εισαγωγή στα Πληροφοριακά Συστήματα και Συστήματα Καταγραφής Δεδομένων

Ο όρος Πληροφοριακό Σύστημα, αναφέρεται σε ένα σύστημα το οποίο διαχειρίζεται και επεξεργάζεται πληροφορίες. Η πληροφορία δεν αναφέρεται αποκλειστικά στα δεδομένα, αλλά και στην ερμηνεία της, καθώς και στον τρόπο με τον οποίο αυτή χρησιμοποιείται. Έτσι ο όρος Πληροφοριακό Σύστημα αποκτά μια γενική έννοια και αφήνει περιθώρια για διαφορετικές ερμηνείες. Ένα Πληροφοριακό Σύστημα μπορεί να αποτελείται από λογισμικό, υλικό και ανθρώπους.

Η κατηγοριοποίηση των διαφόρων ειδών Πληροφοριακών Συστημάτων είναι πολύπλοκη διαδικασία, κυρίως λόγω του ότι:

- υπάρχουν διαφορετικοί τρόποι κατηγοριοποίησης
- ένα σύστημα μπορεί να ανήκει σε δύο κατηγορίες
- η εξέλιξη των συστημάτων μπορεί να αναιρέσει παλαιότερη κατηγοριοποίηση

Επομένως μπορούμε να προσδιορίσουμε μια κατηγοριοποίηση των Πληροφοριακών Συστημάτων σε υψηλό επίπεδο και να δημιουργήσουμε τρεις τάξεις Συστημάτων.

1^{η} Τάξη: Προσωπικά Πληροφοριακά Συστήματα

Συστήματα τα οποία διαχειρίζονται και αποθηκεύουν πληροφορίες για ιδιωτική χρήση ενός προσώπου. Ένα τέτοιο παράδειγμα είναι ο κατάλογος επαφών ενός προσώπου.

2^{η} Τάξη: Επιχειρησιακά – Οργανωτικά Πληροφοριακά Συστήματα

Συστήματα τα οποία έχουν ώς σκοπό την υποστήριξη ενός οργανισμού – μια επιχείρησης. Σε αυτή την τάξη μπορούμε να ξεχωρίσουμε δύο κύριες κατηγορίες. Η πρώτη κατηγορία περιλαμβάνει τα συστήματα τα οποία χρησιμοποιούνται από πληθώρα επιχειρήσεων, όπως είναι τα λογιστικά και τιμολογιακά συστήματα και τα συστήματα αποθήκης, ενώ η δεύτερη, τα συστήματα τα οποία είναι προσαρμοσμένα σε μια συγκεκριμένη επιχείρηση ή οργανισμό και δεν μπορούν να χρησιμοποιηθούν αυτούσια από κάποια άλλη. Αυτά τα συστήματα συχνά εξυπηρετούν εξειδικευμένους σκοπούς και διαδικασίες και για αυτό το λόγο

παραμετροποιούνται ανάλογα με τις εκάστοτε ανάγκες. Τέτοια παραδείγματα είναι συστήματα Νοσοκομείων και Στρατού.

3η Τάξη: Δημόσια Πληροφοριακά Συστήματα. Σε αντίθεση με τα προσωπικά, τα δημόσια Πληροφοριακά Συστήματα, μπορούν να διαχειριστούν και να αποθηκεύσουν πληροφορία στην οποία έχουν πρόσβαση τα μέλη μιας κοινότητας. Τέτοια παραδείγματα είναι συστήματα βιβλιοθηκών, μουσείων και εκπαιδευτικών ιδρυμάτων. [1]

2.2 Ιστορική ανασκόπηση

- **Αποθήκευση Δεδομένων (Υλικό):** Η πρώτη μέθοδος αποθήκευσης δεδομένων αναπτύχθηκε το 1928, από τον Γερμανό μηχανικό Fritz Pheumer. Πρόκειται για τη μαγνητική ταινία, ενώ ο πρώτος σκληρός δίσκος Hard Disk Drive κατασκευάστηκε το 1956 από την IBM. Με συνεχείς εξελίξεις ήρθε στη μορφή που τον ξέρουμε σήμερα, ενώ από τη στιγμή που κυκλοφόρησε, κυριάρχησε στην αγορά ως δευτερεύον μέσο αποθήκευσης. Η κυριαρχία του σκληρού δίσκου βασίστηκε στις επιδόσεις του και στη χωρητικότητά του. [2] Ένας σκληρός δίσκος αποτελείται από επιφάνειες με τροχιές, καλυμμένες με μαγνητικό υλικό, για την αποθήκευση των δεδομένων και ειδικές κεφαλές για ανάγνωση και εγγραφή δεδομένων. Οι επιφάνειες περιστρέφονται με μεγάλη ταχύτητα, ενώ η κεφαλή κινείται στις καθορισμένες τροχιές. Κάθε επιφάνεια έχει ξεχωριστή κεφαλή, ενώ όλες οι κεφαλές κινούνται συγχρονισμένα. [3][4] Οι σκληροί δίσκοι ακόμα και σήμερα λειτουργούν με τον ίδιο τρόπο.
- **Πληροφοριακά Συστήματα:** Τα πληροφοριακά συστήματα υπολογιστών ξεκίνησαν να χρησιμοποιούνται ευρέως το **1970**. Χρησιμοποιούσαν centralized αρχιτεκτονική για τους υπολογιστές και τα δεδομένα. Ήταν προγραμματισμένα σε COBOL (Completely absolute Business Oriented Language) και η χρήση τους ήταν αυστηρά συνδεδεμένη με περιορισμένο εύρος εταιρικών εφαρμογών. Χρησιμοποιούνταν κυρίως για λειτουργίες πληρωμών, αποθήκης, καθώς και χρέωσης. Το **1980** άρχισαν να υποστηρίζονται Πληροφοριακά συστήματα με τη χρήση προσωπικών υπολογιστών, καθώς ξεκίνησε η υποστήριξη βασικού networking μέσα σε μια εταιρία. Ο κύριος σκοπός των πληροφοριακών συστημάτων με προσωπικούς υπολογιστές ήταν να ανεξαρτητοποιηθούν τα τμήματα μια εταιρίας μεταξύ τους, με στόχο, ο τελικός χρήστης να ολοκληρώσει εργασίες στον προσωπικό του Η/Υ. Το **1990** τα πληροφοριακά συστήματα εξελίσσονται ώστε να συνδυάσουν το hardware με τα δεδομένα. Πλέον τα εταιρικά δίκτυα έχουν γίνει standard και ξεκινούν να χρησιμοποιούνται **Βάσεις Δεδομένων**. Η επόμενη και πιο πρόσφατη γενιά πληροφοριακών συστημάτων

ξεκίνησε από το **2000** και χαρακτηρίζεται από την ενσωμάτωση των πληροφοριακών συστημάτων σε δίκτυα. Ο κύριος σκοπός είναι ο κεντρικός έλεγχος, ενώ τα δίκτυα επεκτείνονται με τη χρήση του internet και δίνουν τη δυνατότητα σε διεθνείς εταιρίες και συνεργάτες να εργαστούν στο ίδιο πληροφοριακό σύστημα ή ακόμα και στο διαμοιρασμό δεδομένων ανάμεσα σε διαφορετικά πληροφοριακά συστήματα. Σήμερα τα πληροφοριακά συστήματα βασίζονται στο δίκτυο και προωθούν την ομαδικότητα. Ο κύριος σκοπός τους είναι η αποδοτικότητα και η ταχύτητα. [5]

- **Αποθήκευση Δεδομένων (Λογισμικό):** Τα πρώτα στοιχεία για την αποθήκευση δεδομένων σε βάσεις δεδομένων, χρονολογούνται πριν το 1970. Τότε ονομάζονταν Data Banks, δηλαδή τράπεζες δεδομένων. Το πιο γνωστό σύστημα διαχείρισης της εποχής ήταν το IMS της IBM. Το IMS είναι ιεραρχικό σύστημα διαχείρισης και ήταν εξαιρετικά διαδεδομένο. Στο IMS η διαχείριση των δεδομένων γινόταν σε χαμηλό επίπεδο. Έτσι μια εφαρμογή που χρησιμοποιούσε τη βάση δεδομένων ήταν άμεσα εξαρτώμενη από τη βάση και συγκεκριμένα από την υλοποίησή της. Επιπλέον στο IMS και γενικά στα ιεραρχικά συστήματα διαχείρισης βάσεων, χειριζόμαστε μια εγγραφή ανά στιγμή (record at a time). Τα ιεραρχικά συστήματα παρείχαν πολύ καλή απόδοση και αξιοπιστία, με αποτέλεσμα να κυριαρχήσουν. Όμως το 1970 ο E. F. Codd παρουσιάζει την εργασία του για το **σχεσιακό μοντέλο βάσεων δεδομένων**, η οποία θεωρείται η εργασία με τη μεγαλύτερη επιρροή στις βάσεις δεδομένων. Το σχεσιακό μοντέλο εισάγει τη δυνατότητα μαζικής επεξεργασίας δεδομένων (set at a time). Επιπλέον οι εφαρμογές είναι ανεξάρτητες από τη φυσική υλοποίηση της βάσης δεδομένων. Η εργασία του Codd για το σχεσιακό μοντέλο ήταν τόσο σημαντική που έλαβε το βραβείο Turing το 1981. Μέχρι το 1970 κυριαρχούσαν τα ιεραρχικά συστήματα βάσεων. Στα μέσα της δεκαετίας του ’70 αναπτύχθηκαν δύο σχεσιακά συστήματα τα οποία είναι πρόγονοι των συστημάτων που ξέρουμε σήμερα. Το πανεπιστήμιο της Καλιφόρνια ανέπτυξε το Ingres, ενώ το ερευνητικό κέντρο της IBM ανέπτυξε το System R. Στους απόγονους του Ingres περιλαμβάνεται και ο Microsoft SQL Server, ενώ το System R είναι πρόγονος διαφόρων σχεσιακών συστημάτων όπως της Oracle, του HP Allbase και του IBM SQL/DS. Στη δεκαετία του 1980 προτυποποιείται μια γλώσσα διαχείρισης και επερώτησης σχεσιακών βάσεων δεδομένων, η οποία ονομάζεται SQL (Structured Query Language). Τα σχεσιακά συστήματα κερδίζουν μερίδιο αγοράς καθώς είναι ευκολότερα υλοποιήσιμα, αλλά παρουσιάζουν σφάλματα. Από τότε μέχρι σήμερα γίνονται προσπάθειες ώστε η SQL να γίνει πιο ισχυρή και να συμπεριφέρεται σωστά.[6]

2.3 Σύγχρονα Πληροφοριακά Συστήματα

- **Enterprise Συστήματα:** Μετά την εμφάνιση του internet, ο στόχος των πληροφοριακών συστημάτων ήταν η βελτιστοποίησή τους ώστε οι εταιρίες να εκτελούν και να προγραμματίζουν τις εργασίες τους όσο το δυνατών πιο αποδοτικά. Το αποτέλεσμα ήταν να αυξηθούν οι ανάγκες των εταιριών για τη διαχείριση πόρων ενώ υπήρξε κατακόρυφη αύξηση του όγκου των δεδομένων τα οποία αποθήκευαν. Η χρήση των δεδομένων γινόταν δυσκολότερη και για να λυθούν τα προβλήματα που παρουσιάστηκαν, εμφανίστηκαν τα ERP συστήματα. Τα ERP (Enterprise Resource Planning) δεν αποτελούν ξεχωριστά πληροφοριακά συστήματα. Στην πραγματικότητα είναι ένα πλαίσιο λειτουργίας και συντονισμού ενός συνόλου πληροφοριακών συστημάτων. Ο λόγος δημιουργίας τους είναι οι αυτοματοποιημένες επιχειρησιακές διαδικασίες και ο βαθμός ολοκλήρωσης που προσφέρουν. Πλέον έχουν κατασκευαστεί επιχειρηματικά πλαίσια εφαρμογών ώστε να καλύψουν τις ανάγκες των εταιριών από άποψη πληροφορίας.
- **Cloud Computing:** Ο τρόπος χρήσης των Η/Υ σήμερα, αλλά και η κυκλοφορία πολλών διαφορετικών ειδών υπολογιστών, όπως Tablet, δημιουργεί μια νέα ανάγκη για τους χρήστες. Αυτή η ανάγκη είναι η δυνατότητα πρόσβασης στα δεδομένα τους, οποιαδήποτε στιγμή, από οποιαδήποτε συσκευή. Τα τελευταία χρόνια ο τομέα της πληροφορικής στρέφεται προς την ανάπτυξη συστημάτων τα οποία λειτουργούν και προσφέρονται στο χρήστη ως υπηρεσία. Η πλατφόρμα για την ανάπτυξη τέτοιων συστημάτων είναι το Υπολογιστικό Νέφος. Συγκεκριμένα στο υπολογιστικό νέφος διατίθενται πόροι, ενώ προσφέρει υψηλή αξιοπιστία και ευελιξία των διαθέσιμων πόρων. Ο χρήστης επιλέγει τους πόρους που επιθυμεί, παραδείγματος χάρη, χώρο στο σκληρό δίσκο ή επεξεργαστική ισχύ και τα αποκτά άμεσα με χαρακτηριστική ευκολία. Επομένως η μετάβαση των εφαρμογών και των συστημάτων στο υπολογιστικό νέφος είναι το σωστό βήμα για την παροχή λογισμικού στο οποίο η επεξεργασία των δεδομένων, καθώς και οποιαδήποτε χρήση υπηρεσίας, γίνεται διαδικτυακά. Σήμερα χρησιμοποιούμε καθημερινά εφαρμογές που βασίζονται στο υπολογιστικό νέφος. Τέτοια παραδείγματα είναι το ηλεκτρονικό ταχυδρομείο και φυσικά το cloud storage. Παρά το ότι το υπολογιστικό νέφος είναι μια έννοια η οποία εμφανίστηκε πρόσφατα, η ανάπτυξή του είναι τέτοια που έχουν δημιουργηθεί 4 τύποι υπολογιστικού νέφους. Οι 4 τύποι διαφέρουν μεταξύ τους ως προς τον τρόπο παροχής των υπηρεσιών.
    - **Public Cloud Computing: Το** Δημόσιο Υπολογιστικό νέφος, προσφέρεται σε κάθε χρήστη μέσω διαδικτύου. Οποιοσδήποτε έχει πρόσβαση σε αυτό ανεξάρτητα από την φυσική του θέση. Τέτοια παραδείγματα είναι το Dropbox, Google Drive, Microsoft Azure.

- **Private Cloud Computing:** Το ιδιωτικό υπολογιστικό νέφος προσφέρεται σε χρήστες εντός κάποιον ορίων. Π.χ. σε χρήστες οι οποίοι βρίσκονται στο δίκτυο ενός πανεπιστημίου.
- **Hybrid Cloud Computing:** Ο τρίτος τύπος αποτελεί έναν συνδυασμό δημόσιου και ιδιωτικού υπολογιστικού νέφους.
- **Cloud Computing Κοινότητας:** Στο υπολογιστικό νέφος μιας κοινότητας έχουν πρόσβαση συγκεκριμένοι χρήστες. Παράδειγμα οι χρήστες της υπηρεσίας okeanos πρέπει να είναι μέλη της ακαδημαϊκής κοινότητας.

Όπως αναφέραμε οι 4 τύποι υπολογιστικού νέφους διαφέρουν ως προς τον τρόπο παροχής των υπηρεσιών. Ο τρόπος λειτουργίας είναι ίδιος σε κάθε περίπτωση. Συνοπτικά το υπολογιστικό νέφος προσφέρει τα ακόλουθα πλεονεκτήματα. Απλό στη χρήση, εύκολο στην πρόσβαση, προσαρμογή πόρων, κοινή χρήση δεδομένων. Υπάρχουν 3 πρότυπα τα οποία διέπουν το υπολογιστικό νέφος και είναι τα ακόλουθα.

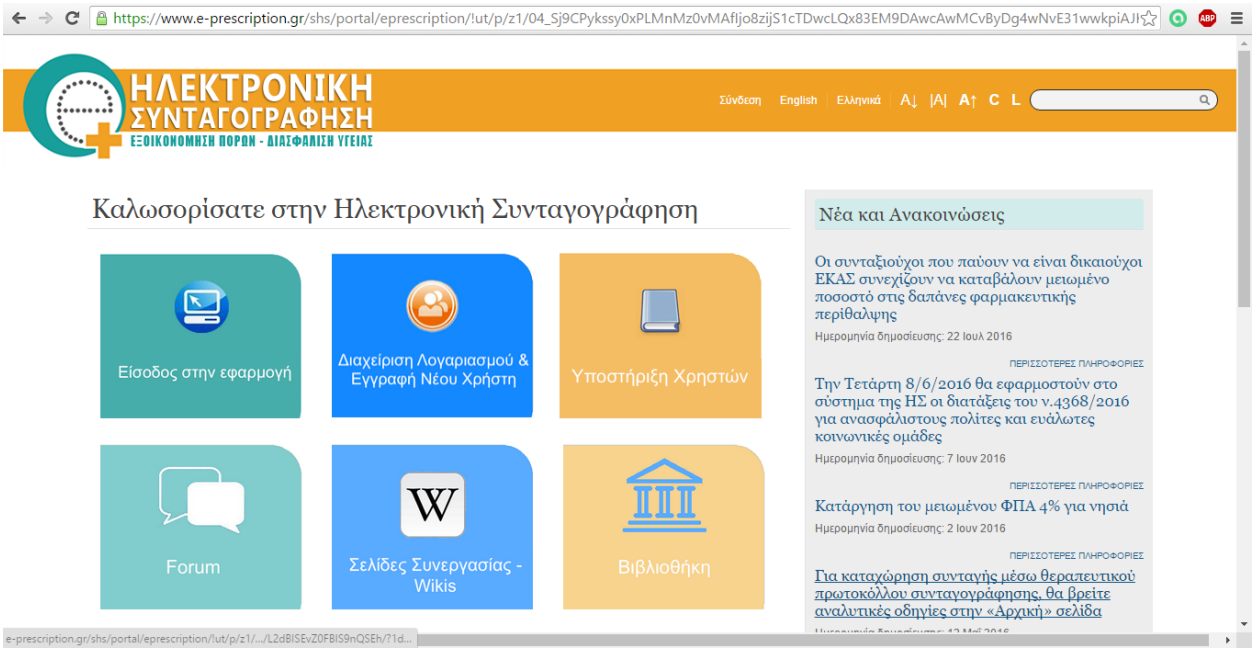
- Infrastructure as a Service (IaaS): Η υπηρεσία που παρέχεται είναι η χρήση του υλικού και κυρίως εξυπηρετητών. Τέτοιοι πάροχοι είναι η Amazon, η Microsoft κ.α.
- Platform as a Service (PaaS): Η υπηρεσία που παρέχεται είναι το υλικό αλλά και λογισμικό συστημάτων.
- Software as a Service (SaaS): Εδώ παρέχονται τα πάντα στο νέφος. Και το υλικό και ολόκληρο το λογισμικό παρέχεται στον χρήστη μέσω του νέφους. Με αυτό το πρότυπο είναι σχεδιασμένα τα σύγχρονα συστήματα ERP. Σήμερα όμως όλο και περισσότερα πληροφοριακά συστήματα είναι κατασκευασμένα με αυτό τον τρόπο. Έτσι ο τελικός χρήστης, έχει στη διάθεσή του ολόκληρη την εφαρμογή η το σύστημα, χωρίς να απαιτείται να εγκαταστήσει ξεχωριστό λογισμικό στον Η/Υ του. Ένα τέτοιο παράδειγμα εφαρμογής είναι το Google Docs. Επίσης προϊόντα όπως το SAP είναι ένα πλήρες πληροφοριακό σύστημα κατασκευασμένο με το πρότυπο SaaP.

[7]

2.4 Πληροφοριακά Συστήματα στην Ελλάδα

- **Πληροφοριακό Σύστημα Ηλεκτρονικής Συνταγογράφησης:** Στην Ελλάδα το ηλεκτρονικό σύστημα συνταγογράφησης ξεκίνησε τη λειτουργία του σταδιακά το 2011. Παρουσίασε προβλήματα και ελλείψεις στη λειτουργία του με αποτέλεσμα να ολοκληρωθεί το 2013 όπου τέθηκε σε λειτουργία η νέα εφαρμογή ηλεκτρονικής συνταγογράφησης και καταχώρησης της επίσκεψης στο γιατρό, καθώς και των εξετάσεων.. Σήμερα το σύστημα ηλεκτρονικής

συνταγογράφησης λειτουργεί υπό τη διεύθυνση www.e-prescription.gr και επιτρέπει τη σύνδεση στην εφαρμογή όλων των ιατρών της ελλάδος [8].



Εικ.1

Η διαδικασία καταχώρησης μια συνταγής φαρμάκων έχει ως εξής. Ο χρήστης – ιατρός συμπληρώνει το username και password που διαθέτει, για να κάνει είσοδο στο σύστημα. Έπειτα πρέπει να επιλέξει μονάδα συνταγογράφησης. Αφού επιλέξει τη μονάδα του, το σύστημα του εμφανίζει οθόνη με τα στοιχεία του. Για να δημιουργήσει μια νέα συνταγή, πρέπει να δημιουργήσει μια νέα επίσκεψη. Ο Ιατρός καταχωρεί τον ΑΜΚΑ του ασφαλισμένου, καθώς και τον λόγο της επίσκεψης και επιλέγει καταχώρηση. Αφού γίνει η καταχώρηση εμφανίζεται η επιλογή για καταχώρηση συνταγής. Ο γιατρός τώρα δημιουργεί τη συνταγή για τον ασθενή και όταν την ολοκληρώσει πρέπει να επιλέξει κλείσιμο επίσκεψης. Όταν επιλέγει κλείσιμο επίσκεψης το σύστημα του εμφανίζει μήνυμα για τον αν θα συμπεριληφθεί η επίσκεψη στο προκαθορισμένο όριο επισκέψεων που μπορεί να δεχθεί κάθε ιατρός. [9] Το σύστημα ηλεκτρονικής συνταγογράφησης χρησιμοποιήθηκε ώστε να μειωθεί το κόστος και παράλληλα να βελτιωθούν οι υπηρεσίες υγείας. Επιπλέον ένα τέτοιο σύστημα, βελτιώνει την ασφάλεια των ασθενών καθώς γίνεται καλύτερη διαχείριση φαρμάκων και μειώνονται τα σφάλματα στις συνταγογραφήσεις. Σε ένα πληροφοριακό σύστημα ηλεκτρονικής συνταγογράφησης είναι δυνατόν να ενσωματωθούν λειτουργίες οι οποίες παρέχουν βοήθεια στον ιατρό – χρήστη. Έτσι καθώς ο ιατρός γράφει τη συνταγή, το σύστημα μπορεί να τον

προειδοποιεί για πιθανή υπερδοσολογία ή βλαβερή αλληλεπίδραση του φαρμάκου με άλλο φάρμακο σε περίπτωση που ο ασθενής λαμβάνει περισσότερα φάρμακα. Δυστυχώς το σύστημα ηλεκτρονικής συνταγογράφησης στην Ελλάδα δεν παρέχει τη μέγιστη δυνατή υποστήριξη στους ιατρούς. Επιπλέον η χρήση του συστήματος απαιτούσε την υπέρβαση εμποδίων με σημαντικότερη την έλλειψη εξοπλισμού. Πλέον κάθε ιατρός χρειάζεται Η/Υ με σύνδεση στο ίντερνετ για να συνταγογραφήσει. Αυτό όμως ήταν δύσκολο κυρίως σε απομακρυσμένες περιοχές. Επιπλέον ο ιατρός δαπανούσε περισσότερη ώρα στο να ολοκληρώσει μια συνταγή, σε σχέση με τον απλό χειρόγραφο τρόπο. Έμμεσα όμως σήμαινε εξοικονόμηση χρόνου και ταλαιπωρίας για τον ασφαλισμένο αφού οποιαδήποτε λάθη είχαν προειδοποιηθεί ήδη από το σύστημα. Επίσης στις δυσκολίες χρήσης του συστήματος στη χώρα μας πρέπει να προσθέσουμε ότι αρχικά υπήρχε χαμηλό επίπεδο παρεχόμενων υπηρεσιών και συχνά πολλοί ιατροί αντιμετώπιζαν δυσκολίες με την εξοικείωσή τους με το σύστημα . Παρά όλα αυτά όμως το σύστημα της ηλεκτρονικής συνταγογράφησης συνέβαλε στην εξοικονόμηση σημαντικών πόρων καθώς και χρόνου, ενώ μελλοντικά η προσθήκη επιπλέον λειτουργιών θα το καταστήσουν εξυπνότερο και πιο χρήσιμο. [10]

- **Πληροφοριακό Σύστημα Διαχείρισης Φαρμάκων σε Δημόσιο Νοσοκομείο:** Το σύστημα διαχείρισης φαρμάκων ενός νοσοκομείου αποτελείται από τις διαδικασίες διάθεσης φαρμάκων στις κλινικές και την καταγραφή της αγωγής κάθε ασθενούς. Στόχος του συστήματος είναι να παρακολουθείται η πορεία κάθε φαρμάκου μέσα στο νοσοκομείο. Μελετώντας τη διαδικασία διακίνησης των φαρμάκων σε ένα νοσοκομείο, μπορούμε να αναπαραστήσουμε τη διακίνηση και άλλων αντικειμένων εκτός φαρμάκων σε ένα νοσοκομείο ή ακόμα και τη διακίνηση αντικειμένων σε ένα εκπαιδευτικό ίδρυμα. Για τη λειτουργία του συστήματος διαχείρισης φαρμάκων στο νοσοκομείο θεωρούμε ότι υπάρχει επαρκής ποσότητα φαρμάκων και υλικών στο νοσοκομείο και κάθε φάρμακο το οποίο μπαίνει στο νοσοκομείο, έχει καταγραφεί και εισαχθεί στο σύστημα. Επιπλέον για κάθε ασθενή μιας κλινικής καταγράφονται τα στοιχεία του και η φαρμακευτική του αγωγή. Επιπλέον καταγράφεται ο ιατρός που είναι υπεύθυνος για κάθε ασθενή. Με τη χρήση του συστήματος σε ένα νοσοκομείο γίνεται πιο εύκολη η λήψη αποφάσεων σχετικά με τις παραγγελίες φαρμάκων, μειώνονται τα λάθη και οι απώλειες, ενώ γίνεται ευκολότερη παρακολούθηση της αγωγής των ασθενών. [11]
- **Πληροφοριακό Σύστημα TAXIS:** Το κυριότερο και μεγαλύτερο πληροφοριακό σύστημα στην Ελλάδα είναι το TAXIS (Taxation Information System). Το Taxis λειτουργεί εδώ και μια δεκαετία με τη μορφή που έχει σήμερα (2016). Το Taxis

είναι το πληροφοριακό σύστημα της Ελλάδας για τη διαχείριση φορολογικών θεμάτων. Η πρώτη εφαρμογή του συστήματος ξεκίνησε το 2000 και με αναβαθμίσεις και επεκτάσεις έως το 2006 έφτασε στο σύστημα που χρησιμοποιούμε σήμερα.

Εικ.2

Το Taxis ως πληροφοριακό σύστημα έχει στόχο τη βελτίωση της εξυπηρέτησης των πολιτών, αφού μέσω διαδικτύου εκτελούν εργασίες για τις οποίες θα σπαταλούσαν περισσότερο χρόνο, την καλύτερη παρακολούθηση των φορολογικών εσόδων του κράτους και την αντιμετώπιση της φοροδιαφυγής. Η πιο διαδεδομένη εφαρμογή του Taxis είναι η εφαρμογή δήλωσης εισοδήματος Ε1. Οι πολίτες υποβάλουν τη δήλωσή τους ηλεκτρονικά μέσω browser, αφού συνδεθούν στο σύστημα. Το σύστημα ελέγχει τυχόν παραλείψεις στη δήλωση του χρήστη, του δίνει τη δυνατότητα να επεξεργαστεί τη δήλωση πριν την οριστική υποβολή και στο τέλος του προσφέρει πληροφόρηση για τον φόρο που καλείται να πληρώσει ή να του επιστραφεί (σπανίως). [12]

Κεφάλαιο 3: Web Εφαρμογές

Στο κεφάλαιο 3 γίνεται ανάλυση της λειτουργίας των web εφαρμογών, των χαρακτηριστικών τους, των τρόπων κατασκευής τους, καθώς και της αρχιτεκτονικής client server.

3.1 Εισαγωγή

Με τον όρο web application (διαδικτυακή εφαρμογή) αναφερόμαστε σε λογισμικό client-server στο οποίο η διεπαφή χρήστη παρέχεται μέσω web browser. Σήμερα ο διαχωρισμός ανάμεσα σε web applications και web sites, δεν είναι απόλυτα σαφής. Μπορούμε να τα διαχωρίσουμε τυπικά στο γεγονός ότι μια web εφαρμογή λαμβάνει είσοδο από το χρήστη, την επεξεργάζεται και επιστρέφει ένα αποτέλεσμα. Αντίθετα, τα web sites συνήθως παρουσιάζουν το ίδιο περιεχόμενο σε μεγάλο μέρος χρηστών. Οι web εφαρμογές τείνουν να προσφέρουν παρόμοια λειτουργικότητα με αυτή των desktop εφαρμογών. [13]

3.2 Ιστορία

Στα πρώτα μοντέλα Client-Server οι λειτουργίες μοιράζονταν ανάμεσα στον server και στον client, αφού απαιτούταν η εγκατάσταση εφαρμογής στον client, η οποία λειτουργούσε ως User Interface. Αυτό το μοντέλο παρουσίαζε σημαντικά προβλήματα, όπως στην ενημέρωση της εφαρμογής του Server. Κάθε εφαρμογή στους Client θα έπρεπε να ενημερωθεί και αυτή. Επιπλέον σε τέτοια μοντέλα υπάρχει εμφανής εξάρτηση της client εφαρμογής με το λειτουργικό σύστημα του client. [13]

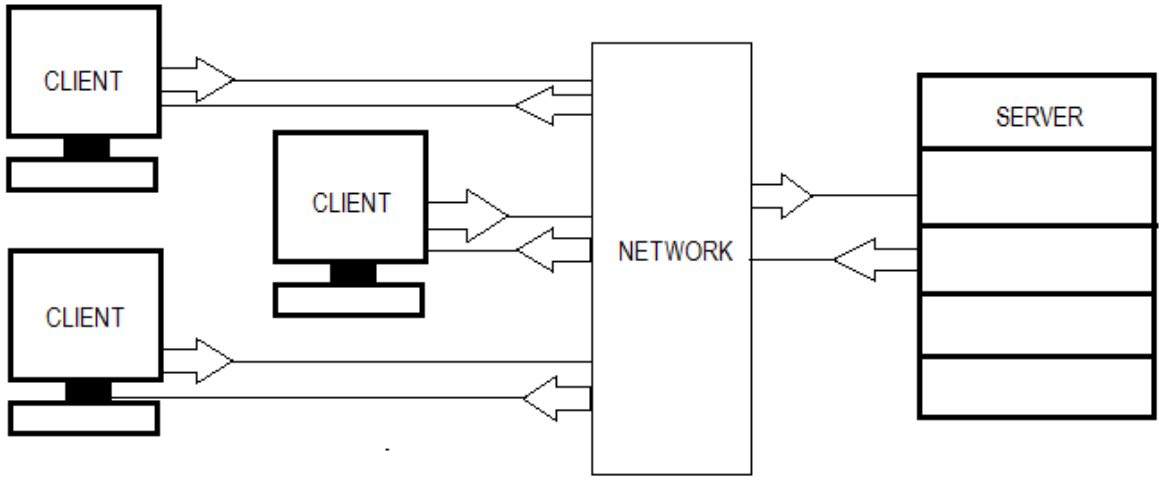
Παρά το γεγονός ότι ο Tim Berners Lee εφηύρε το World Wide Web από το 1989 και το 1990 κατασκεύασε τον πρώτο web browser, ως το 1995 ήταν γνωστή μόνο η μεταφορά στατικών δεδομένων. Δηλαδή ο Client λάμβανε ένα στατικό έγγραφο το οποίο ήταν η κάθε ιστοσελίδα την οποία ζητούσε από το Server. Έτσι δεν υπήρχε καμία πραγματική αλληλεπίδραση στον Client, παρά μόνο αν θεωρήσουμε ότι η μετάβαση από μια στατική σελίδα, σε μια άλλη, μπορεί να παρουσιαστεί ως αλληλεπίδραση. Αυτό άλλαξε το 1995 όταν η Netscape παρουσίασε μια γλώσσα προγραμματισμού η οποία έτρεχε στον Client. Αυτή ήταν η JavaScript, με την οποία μπορούσαν να υπάρχουν δυναμικά στοιχεία στη διεπαφή που έτρεχε στον Client. [13][14]

Η ανάπτυξη του www συνεχίστηκε και το 2005 εισήλθε ο όρος Ajax, ο οποίος δεν αποτελεί γλώσσα προγραμματισμού, αλλά ένα σύνολο τεχνικών που συνεργάζονται ώστε με τη χρήση κατάλληλων τεχνολογιών να δημιουργήσουν ασύγχρονες διαδικτυακές εφαρμογές. Ο όρος ασύγχρονες αναφέρεται στην ασύγχρονη επικοινωνία του Client με τον Server. Οι τεχνικές Ajax αναπτύχθηκαν κυρίως από τη Microsoft αρχικά και την Google αργότερα. Το

2006, ο οργανισμός W3C ανακοίνωσε την προσπάθειά του να δημιουργήσει ένα Standard που θα διέπει το XMLHttpRequest object [15][16].

3.3 Αρχιτεκτονική Client-Server

Η αρχιτεκτονική Client – Server βασίζεται στο διαμοιρασμό λειτουργιών ανάμεσα στον πάροχο μιας υπηρεσίας (Server) και σε αυτόν που τη ζητάει (Client). Ο διαμοιρασμός αυτός γίνεται μέσω δικτύου διασύνδεσης. Το δίκτυο διασύνδεσης μπορεί να είναι ένα τοπικό δίκτυο ενός οργανισμού, ένα δίκτυο ενός πανεπιστημίου κ.α.



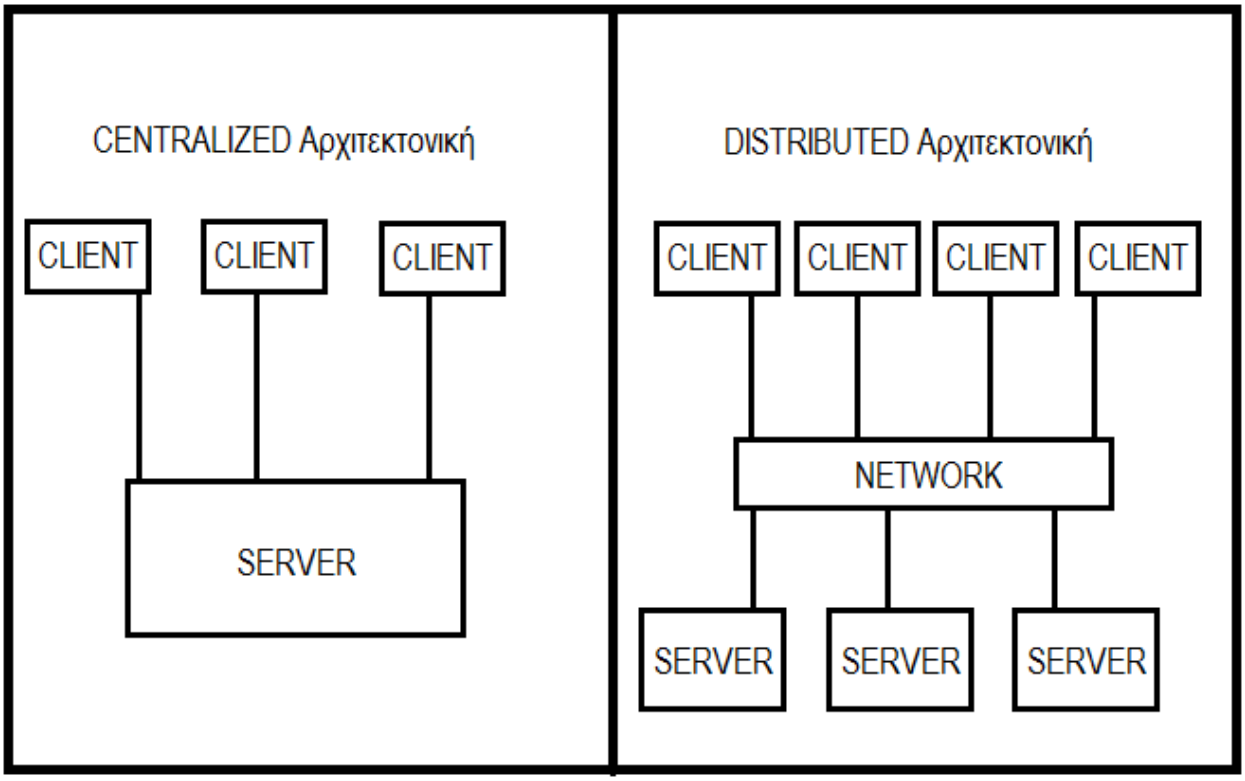
Εικ.3

Στη Centralized αρχιτεκτονική Client – Server όλη η επεξεργασία των δεδομένων πραγματοποιείται από έναν κεντρικό υπολογιστή. Διαχειρίζεται κάθε λειτουργία, κάθε είσοδο και έξοδο καθώς και αποθήκευση και ανάγνωση δεδομένων.

Στην κατανεμημένη αρχιτεκτονική Client – Server ο επεξεργαστικός φόρτος διαμοιράζεται ανάμεσα σε πολλούς υπολογιστές οι οποίοι είναι ξεχωριστοί από φυσικής άποψης και επικοινωνούν μέσω δικτύου.

3.3.1 Μοντέλο Client Server – Μεταβολή με την πάροδο των χρόνων.

- Την δεκαετία του 1970 το μοντέλο που κυριαρχούσε ήταν η Centralized αρχιτεκτονική. Δηλαδή ένας υπολογιστής διαχειρίζεται όλη την είσοδο, έξοδο, αποθήκευση και ανάγνωση δεδομένων και ανάγκες σε επεξεργαστική ισχύ.
- Σήμερα χρησιμοποιείται ευρέως η κατανεμημένη αρχιτεκτονική, δηλαδή πολλοί υπολογιστές καλύπτουν τις ανάγκες σε επεξεργαστική ισχύ. Οι υπολογιστές είναι κατανεμημένοι σε φυσικό επίπεδο και επικοινωνούν με το δίκτυο.
- Το μοντέλο Client – Server του μέλλοντος είναι η συνεργατική αρχιτεκτονική. Πολλοί υπολογιστές καλύπτουν τις ανάγκες σε επεξεργαστική ισχύ. Είναι κατανεμημένοι σε φυσικό επίπεδο και επικοινωνούν με το δίκτυο. Η διαφορά με τη κατανεμημένη αρχιτεκτονική είναι ότι οι πόροι διαμοιράζονται για να καλύψουν τις επεξεργαστικές ανάγκες. Οι λειτουργίες μια εφαρμογής μπορούν να εκτελούνται είτε στον Client, είτε στον Server. Έτσι η συνεργατική αρχιτεκτονική εκμεταλλεύεται στο έπακρο τα πλεονεκτήματα τις Client – Server αρχιτεκτονικής και του διαμοιρασμού δεδομένων. Ένα σύστημα που χρησιμοποιεί συνεργατική αρχιτεκτονική είναι αρκετά δύσκολο να στηθεί – ρυθμιστεί και να συντηρηθεί, αλλά παρέχει μεγάλο κέρδος στην παραγωγικότητα και αποδοτικότητα του δικτύου υπολογιστών. Οι ανάγκες σε επεξεργαστική ισχύ και ο διαμοιρασμός των δεδομένων εξαρτώνται από το είδος των εφαρμογών, τον εξοπλισμό, τον τύπο της βάσης δεδομένων καθώς και των αναγκών χρήσης του συστήματος. Όλα αυτά καθιστούν ένα σύστημα συνεργατικής αρχιτεκτονικής εξαιρετικά περίπλοκο. [17]

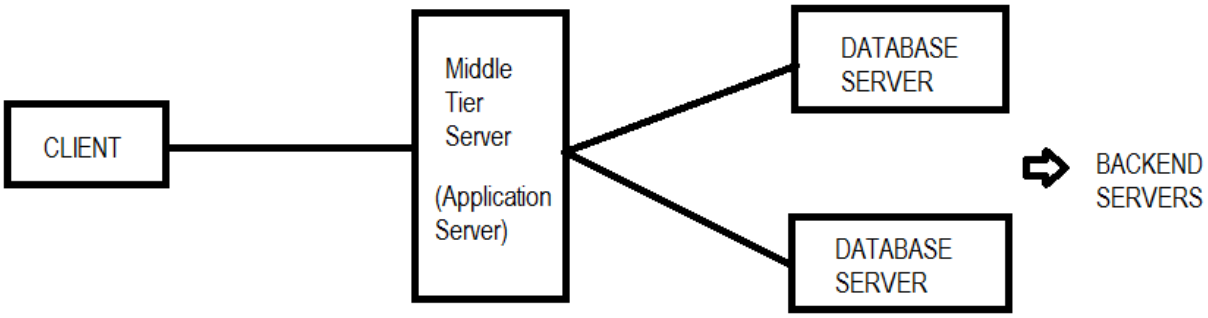


Εικ.4

3.3.2 Κλάσεις εφαρμογών Client – Server.

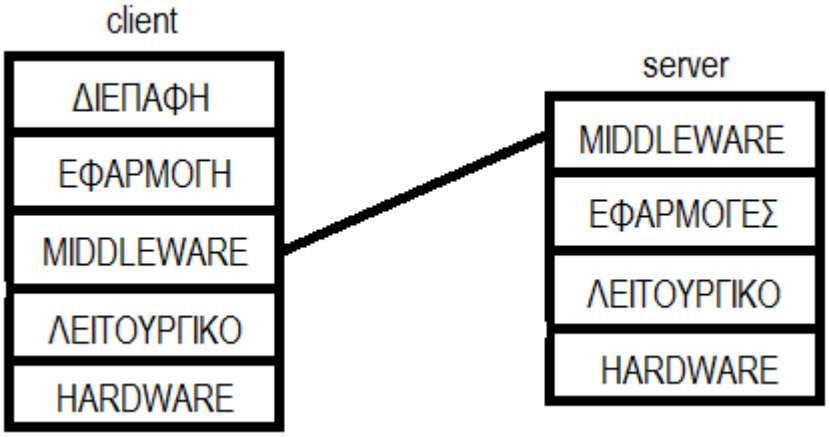
- **Server – Based εφαρμογές:** Σε Server – Based εφαρμογές, όλες οι λειτουργίες γίνονται στο Server. Ολόκληρο το Application Logic βρίσκεται στο server, ενώ ο client έχει μόνο τη διεπαφή χρήστη.
- **Client – Based εφαρμογές:** Στις client – based εφαρμογές όλες οι λειτουργίες και το Application Logic γίνονται στον client, ενώ ο Server αναλαμβάνει τις λειτουργίες τις βάσης δεδομένων και επαλήθευσης δεδομένων.
- **Cooperative εφαρμογές:** Η λειτουργία της εφαρμογής γίνεται με τον βέλτιστο τρόπο, αλλά ένα τέτοιο σύστημα είναι δύσκολο στην εγκατάσταση και τη συντήρηση.

Το λογισμικό των εφαρμογών κατανέμεται σε τρεις τύπους μηχανημάτων. Στο μηχάνημα χρήστη (client), στο μηχάνημα server μεσαίου στρώματος (πχ ένα gateway) και στον backend server.



Εικ.5

Ρόλος του Middleware στην αρχιτεκτονική Client – Server:

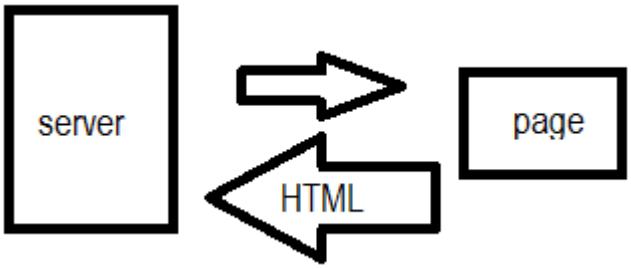


Εικ.6

1) Παρέχει μεθόδους για την πρόσβαση στους πόρους του συστήματος σε όλες τις πλατφόρμες.
2) Παρέχει τη δυνατότητα χρήσης της ίδια μεθόδου για την πρόσβαση στα δεδομένα. [18]

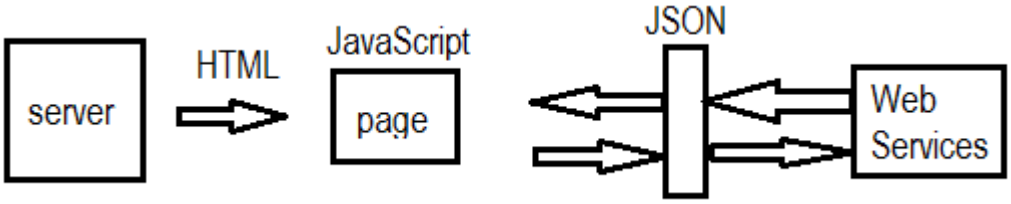
3.4 Τύποι αρχιτεκτονικών Web εφαρμογών

- **Server side HTML web application:** Είναι η πιο διαδεδομένη αρχιτεκτονική web εφαρμογών. Ο server παράγει html περιεχόμενο και το στέλνει στον client ως html σελίδα.



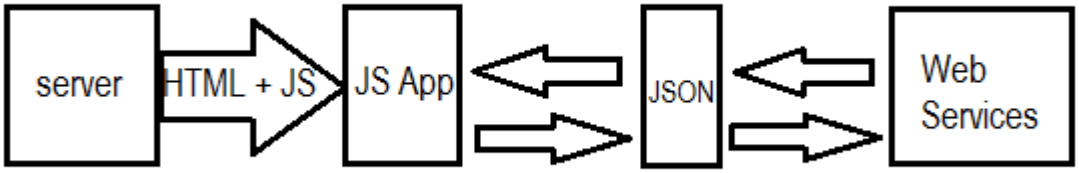
Αυτή η αρχιτεκτονική είναι η λιγότερο αποδοτική από άποψη χρόνου απόκρισης και χρηστικότητας, καθώς μεγάλος όγκος δεδομένων μεταφέρεται μεταξύ του server και του client. Δεν υπάρχει δυνατότητα αποστολής και λήψης δεδομένων real time με αποτέλεσμα για κάθε αλλαγή να απαιτείται η επαναφόρτωση ολόκληρης της σελίδας. Αντίθετα η αρχιτεκτονική αυτή προσφέρει υψηλό βαθμό συνδεσιμότητας και είναι εύκολα υλοποιήσιμη καθώς κάθε ένα URL λαμβάνει συγκεκριμένο και ξεχωριστό περιεχόμενο στο server. Η απόδοση όμως οριοθετείται από το μεγάλο όγκο δεδομένων, καθώς για κάθε συμβάν πρέπει να παραχθεί περιεχόμενο για ολόκληρη τη σελίδα και όχι μόνο για το μέρος το οποίο αλλάζει.

- **JS generation widgets (AJAX):** Η αρχιτεκτονικής αυτή είναι η εξελιγμένη μορφή της πρώτης αρχιτεκτονικής, στην οποία η σελίδα που προβάλλεται αποτελείται από ανεξάρτητα τμήματα.



Τα δεδομένα φορτώνονται σε αυτά τα τμήματα με AJAX queries είτε ως html είτε ως JSON και μετατρέπονται με τη χρήση JavaScript σε περιεχόμενο ορατό στη σελίδα. Το μεγάλο πλεονέκτημα της αρχιτεκτονικής αυτής είναι η δυνατότητα να μεταφερθούν δεδομένα από τον server, μόνο για το τμήμα της σελίδας που ζητά ο client. Τα τμήματα μια σελίδας μπορούν να είναι λειτουργικά ανεξάρτητα. Ο όγκος των δεδομένων που μεταφέρεται είναι πολύ μικρότερος σε σχέση με την προηγούμενη αρχιτεκτονική, με αποτέλεσμα να παρουσιάζει μειωμένο χρόνο απόκρισης. Κατά την πρώτη φόρτωση μιας σελίδας, ο χρόνος που απαιτείται θα είναι συγκρίσιμος με το χρόνο που θα απαιτούταν για τη φόρτωση της ίδιας σελίδας με την πρώτη αρχιτεκτονική. Η χρήση της σελίδας στη συνέχεια θα είναι γρηγορότερη καθώς θα φορτώνονται μόνο τα τμήματα που απαιτούνται. Επιπλέον είναι δυνατή η χρήση πριν από την ολοκληρωτική φόρτωση της σελίδας.

- **Service – Oriented single page web εφαρμογές (Web 2.0, HTML5 apps):** Η ουσία του τρόπου λειτουργίας της αρχιτεκτονικής αυτής περιέχεται στο ότι μια html σελίδα λαμβάνεται από τον server, η οποία λειτουργεί ως ένας container για τον JavaScript κώδικα. Ο κώδικας JavaScript είναι υπεύθυνος για ένα συγκεκριμένο web service και λαμβάνει τα δεδομένα που χρειάζεται. Με αυτά παράγει το html περιεχόμενο της σελίδας.



page

Η αρχιτεκτονική αυτή αποτελεί μια σύνθετη εφαρμογή JavaScript στην οποία μέρος της λειτουργικότητας βρίσκεται στον client. Έτσι μια τέτοια εφαρμογή μπορεί να λειτουργήσει ακόμα και αυτοδύναμα στον client. Επιπλέον προσφέρει το μικρότερο δυνατό χρόνο απόκρισης καθώς ο όγκος των δεδομένων που απαιτούνται είναι ο ελάχιστος. Σε ειδικές περιπτώσεις μεγάλου όγκου δεδομένων, η επεξεργασία θα πρέπει να μεταφερθεί στο web service. Τέλος οι εφαρμογές της αρχιτεκτονικής αυτής επηρεάζονται στον μικρότερο βαθμό από τον server, καθώς το μόνο που έχει να κάνει ο server είναι να παρέχει την JavaScript εφαρμογή στον browser. Η απόδοση του client και ο τύπος του browser έχουν το μεγαλύτερο ρόλο στην απόδοση της εφαρμογής.
[19]

Κεφάλαιο 4: Δικτυακός προγραμματισμός

Στο κεφάλαιο 4 γίνεται ανάλυση των δημοφιλέστερων τεχνικών, εργαλείων και γλωσσών προγραμματισμού για την ανάπτυξη web εφαρμογών.

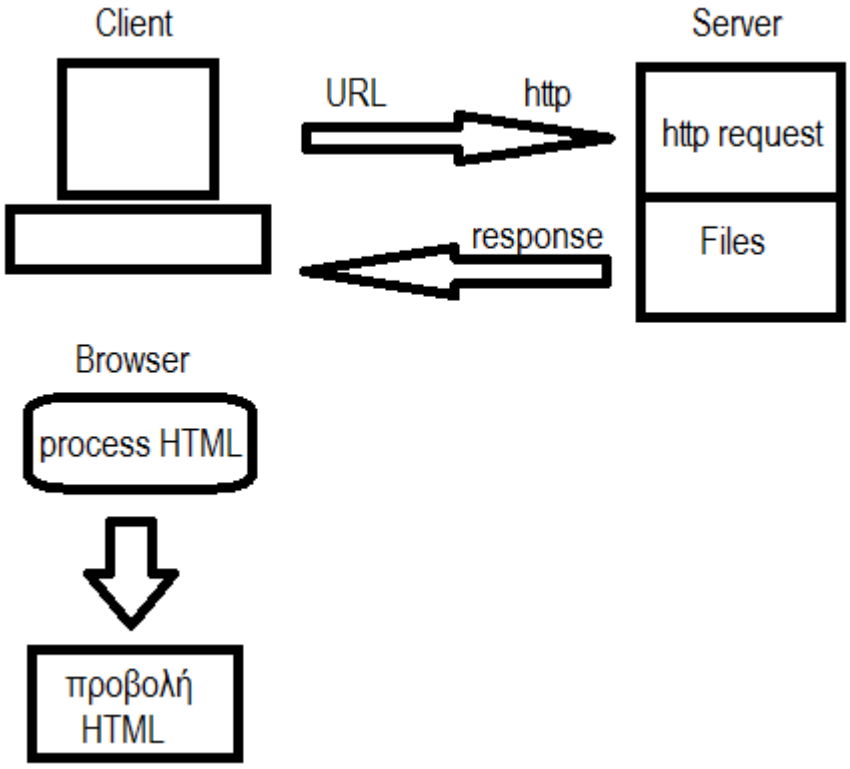
4.1 Εισαγωγή

Ο παγκόσμιος ιστός ή αλλιώς World Wide Web αποτελεί το μεγαλύτερο δίκτυο υπολογιστών του κόσμου. Η επικρατέστερη αρχιτεκτονική σε αυτό το δίκτυο είναι η αρχιτεκτονική Client - Server, ενώ η επικρατέστερη γλώσσα παρουσίασης είναι η HTML. Η HTML διαθέτει κατάλληλα tags για να περιγράφει τη δομή των HTML σελίδων.

Ο δικτυακός προγραμματισμός χωρίζεται σε δύο βασικές κατηγορίες.

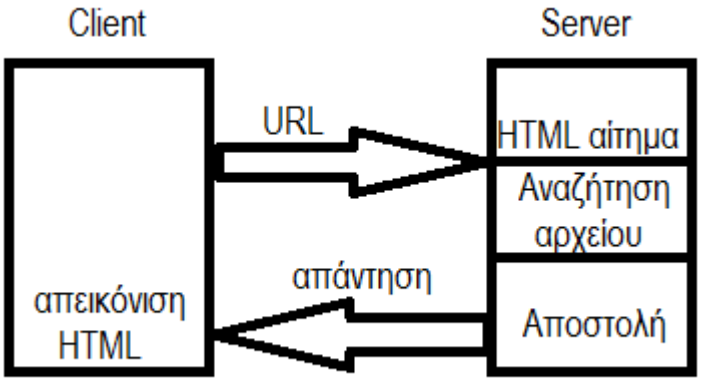
- Client side
- Server side

Στον client side προγραμματισμό η διαδικασία έχει ως εξής. Ο client αιτείται ένα αρχείο από τον server μέσω ενός URL. Ο Server λαμβάνει ένα http αίτημα και αναζητά το αρχείο. Ο Client λαμβάνει το αρχείο και με την HTML το παρουσιάζει κατάλληλα.



Η εκτέλεση του αρχείου γίνεται στον client μετά τη λήψη του. Προσφέρει μειωμένο φόρτο εργασίας στον server και προστατεύει τον server από σφάλματα.

Στον server side προγραμματισμό, τα βήματα που ακολουθούνται έχουν ως εξής. Ο client αιτείται μέσω ενός URL. Ο server λαμβάνει την http αίτηση και ανακτά το αρχείο που του ζητήθηκε. Όταν είναι έτοιμο, το στέλνει ως απάντηση στον client ο οποίος το απεικονίζει με την HTML



Η σελίδα δημιουργείται δυναμικά στο server και ο client τη λαμβάνει. Προσφέρει δυνατότητα σύνδεσης με βάση δεδομένων, καλύτερη συμβατότητα, συγχρονισμό, ανεξαρτησία από τον client και καλύτερη ασφάλεια.

AJAX: Asynchronous JavaScript and XML

Δεν πρόκειται για γλώσσα προγραμματισμού, αλλά για τεχνική προγραμματισμού με χρήση διαφόρων τεχνολογιών. Το Ajax μας επιτρέπει την ασύγχρονη επικοινωνία ανάμεσα σε client και server.

XML: eXtensive Markup Language

Η XML χρησιμοποιείται για τη δόμηση της πληροφορίας και την ανταλλαγή δεδομένων μεταξύ εφαρμογών.

JavaScript: Χρησιμοποιείται παράλληλα με την HTML. Προσθέτει λειτουργικότητα στις σελίδες στις οποίες ενσωματώνεται και είναι η βασική γλώσσα για τη χρήση AJAX.

[20]

4.2 HTML

Η HTML είναι η συνηθέστερη γλώσσα για την κατασκευή ιστοσελίδων. Τα αρχικά της προέρχονται από το Hyper Text Markup Language που σημαίνει ότι είναι μια γλώσσα περιγραφής και όχι προγραμματισμού. Η HTML χρησιμοποιεί tags τα οποία αντιπροσωπεύουν τα διάφορα elements. Οι browsers είναι φτιαγμένοι έτσι ώστε να χρησιμοποιούν τα html tags για να απεικονίσουν το περιεχόμενο της ιστοσελίδας. Μια σελίδα HTML χτίζεται με html elements. Τα html elements περιγράφονται από τα html tags. Οι browsers δεν προβάλουν τα html tags, αλλά τα χρησιμοποιούν ώστε να μεταφράσουν το περιεχόμενο της σελίδας. Η HTML συμπεριλαμβάνοντας την CSS διαμορφώνει την εμφάνιση του περιεχομένου. Επιπλέον η HTML μπορεί να ενσωματώσει προγράμματα γραμμένα σε scripting γλώσσα προγραμματισμού, όπως είναι η JavaScript. Με την JavaScript μπορούμε να καθορίσουμε πως συμπεριφέρεται το περιεχόμενο τις σελίδας κατά την αλληλεπίδραση του χρήστη.

Μια σελίδα HTML απαιτεί ορισμένα tags ώστε να αναγνωριστεί σωστά από έναν browser. Παρακάτω βλέπουμε τη βασική μορφή μιας HTML σελίδας.

<!DOCTYPE html> δηλώνει ότι η σελίδα είναι HTML5

<html> είναι το αρχικό element κάθε html σελίδας

<head> περιέχει πληροφορίες και meta δεδομένα

<title>This is the title</title>

</head>

<body>

<h1></h1> τα tags είναι τα ονόματα των element μέσα σε brackets

</body>

</html>

Εκδόσεις της HTML:

Το 1989 ο φυσικός Tim Berners-Lee σε ένα έγγραφό του περιέγραφε τη λειτουργία ενός internet-based συστήματος για την ανταλλαγή εγγράφων στο CERN. ΤΟ 1990 κατασκεύασε το λογισμικό για ένα τέτοιο σύστημα. Η πρώτη δημόσια διαθέσιμη περιγραφή της HTML ήταν ένα έγγραφο με την ονομασία HTML Tags από τον Tim Berners-Lee το **1991**. Το **1995** δημοσιεύεται η **HTML 2.0** με την επίσημη ονομασία RFC 1866 και μέχρι το 1997 δημοσιεύτηκαν

συμπληρωματικά RFC τα οποία πρόσθεταν δυνατότητες. Το **1997** δημοσιεύεται η **HTML 3.2** η οποία ήταν η πρώτη έκδοση που αναπτύχθηκε και δημοσιεύτηκε από τον οργανισμό **W3C.** Το 1999 δημοσιεύεται η HTML 4.0.1 και πάλι από το W3C. Το 2000 δημοσιεύεται η XHTML ως ISO/IEC πρότυπο. Μέχρι το 2014 δεν υπήρχε νέα έκδοση HTML. Η HTML5 δημοσιεύτηκε το 2014 από το W3C και το 2016 η HTML 5.1 η οποία είναι κ η τελευταία έκδοση έως σήμερα.

[21]

Τα html elements μπορούν να διαθέτουν attributes. Τα attributes παρέχουν επιπλέον πληροφορίες για ένα element. Ένα δημοφιλές attribute είναι το href. Στην html δηλώνουμε τους συνδέσμους (links) με το <a> tag, ενώ στο attribute href δηλώνουμε τη διεύθυνση. Με το id attribute δηλώνουμε ένα μοναδικό αναγνωριστικό id σε ένα element, ενώ με το style attribute δηλώνουμε στυλ εμφάνισης για το element. Αυτά είναι μόνο λίγα από τα συνολικά html elements τα οποία έχουν καθοριστεί από το W3C (World Wide Web Consortium).

Το W3C είναι μια διεθνής κοινότητα στην οποία συμμετέχουν οργανισμοί μέλη, σταθερό προσωπικό και το κοινό, με σκοπό να αναπτύξουν τα web standards. Του W3C ηγείται ο Tim Berners-Lee, δημιουργός του internet. Βασική δουλειά του W3C είναι η ανάπτυξη πρωτοκόλλων και κατευθυντήριων γραμμών ώστε να εξασφαλίσει την συνεχή ανάπτυξη του world wide web. Αυτό συμβαίνει με τον καθορισμό προτύπων με τα οποία λειτουργεί το word wide web. [22]

4.3 JavaScript

Η JavaScript είναι μια από τις βασικές γλώσσες προγραμματισμού στο διαδίκτυο. Δημιουργήθηκε το 1995 από τον Brendan Eich ο οποίος ήταν μηχανικός λογισμκού στην εταιρεία Netscape. Η JavaScript κυκλοφόρησε για πρώτη φορά το 1996 με τον Netscape 2.0, το δημοφιλέστερο browser της εποχής. Λίγο αργότερα η Microsoft παρουσίασε τη δική της εκδοχή της JavaScript, στον Internet Explorer 3, με την ονομασία JScript. Η Jscript είχε παρόμοιες λειτουργίες και ήταν συμβατή με την Javascript. Εκείνη την περίοδο η Netscape υπέβαλε την JavaScript στον ευρωπαϊκό οργανισμό προτυποποίησης, Ecma International και το ίδιο έτος δημιουργήθηκε η πρώτη έκδοση του προτύπου ECMAScript, το οποίο είναι η επίσημη ονομασία της JavaScript. Η μεγαλύτερη αναβάθμιση του προτύπου ήρθε το 1999 ε την έκδοση ECMAScript 3 και από τότε μέχρι σήμερα παρέμεινε σχεδόν σταθερό. Στο τέλος του 2009 δημοσιεύθηκε η 5η έκδοση, ενώ η έκδοση ECMAScript 6 δημοσιεύθυηκε στα μέσα του 2015.

Η JavaScript ακριβώς επειδή είναι σχεδιασμένη να τρέχει ως scripting γλώσσα στο

περιβάλλον που φιλοξενείται, δε διαθέτει δικούς της μηχανισμούς για είσοδο και έξοδο. Βασίζεται αποκλειστικά στο περιβάλλον φιλοξενίας για να επικοινωνεί με τον έξω κόσμο. Το πιο συνηθισμένο περιβάλλον φιλοξενίας της JavaScript είναι ο browser, αλλά μεταφραστές JavaScript συναντώνται σε προϊόντα όπως το Adobe Photoshop και σε server side περιβάλλοντα όπως το Node.js, σε ενσωματωμένους υπολογιστές, ακόμα και στο γραφικό περιβάλλον GNOME ενός από τα δημοφιλέστερα GUI του Linux. Η JavaScript είναι μια δυναμική γλώσσα προγραμματισμού με αντικείμενα και μεθόδους και υποστηρίζει αντικειμενοστρεφή προγραμματισμό.

Μια μέθοδος της JavaScript είναι η getElementById(). Με αυτή τη μέθοδο μπορούμε να επεξεργαστούμε το HTML περιεχόμενο της σελίδας, δίνοντάς της το id του HTML element.

document.getElementById(“element_id”).innerHTML = “This is JavaScript”;

Επιπλέον με την getElementById μπορούμε να παρέμβουμε στο attribute ενός element καθώς και στο style. Με την JavaScript μπορούμε να κρύωουμε και να εμφανίσουμε elements.

document.getElementById(“element_id”).style.display = “none”;

Η JavaScript μπορεί να εισαχθεί κατευθείαν σε ένα html αρχείο. Αρκεί ο κώδικας JavaScript να περικλείεται σε <script> tag.

<script> JavaScript code </script>

Επίσης μπορούμε να χρησιμοποιήσουμε JavaScript από εξωτερικά αρχεία βάζοντας το path του εξωτερικού αρχείου στο src attribute του <script> tag.

<script src=”path/to/file.js”> JavaScript code </script>

Τα αρχεία JavaScript πρέπει να έχουν την κατάληξη .js

Με την JavaScript μπορούμε να προβάλουμε δεδομένα με διάφορους τρόπους.

- Γράφοντας σε ένα html element χρησιμοποιώντας την innerHTML
- Τυπώνοντας στην HTML έξοδο χρησιμοποιώντας την document.write()
- Δημιουργώντας ένα alert box με την window.alert()
- Τυπώνοντας στην κονσόλα του browser με την console.log().

Οι μέθοδοι document.write() και console.log() θα πρέπει να χρησιμοποιούνται μόνο για testing και debug καθώς η document.write θα διαγράψει το html περιεχόμενο και η console.log τυώνει μόνο στην consola, πράγμα το οποίο δεν είναι καθόλου εύχρηστο για τον τελικό χρήστη.

4.3.1 Σύνταξη JavaScript

Όπως κάθε γλώσσα προγραμματισμού, έτσι και η JavaScript διαθέιτει statements,

μεταβλητές, operators, τιμές, keywords, expressions κ.α.

Για να αποθηκεύσουμε τιμές δεδομένων χρησιμοποιούμε τις μεταβλητές. Για να δηλώσουμε μια μεταβλητή χρησιμοποιούμε τη σύνταξη var userName; και userName = “name” για να αναθέσουμε την τιμή name στη μεταβλητή userName.

Ο “equal to” operator γράφεται ως “==”

Οι μεταβλητές που δημιουργούμε και δεν τους αναθέτουμε τιμή, έχουν την τιμή undefined. Αν προσθέσουμε μεταβλητές τύπου string, το αποτέλεσμα θα είναι τα 2 string concatenated.

Συναρτήσεις

Μια JavaScript συνάρτηση είναι ένα τμήμα κώδικα το οποίο εκτελεί κάποια ενέργεια. Η συνάρτηση εκτελείται όταν κάτι την καλέσει. Η σύνταξη είναι η ακόλουθη.

function function_name(param1, param2){

JavaScript code

}

Η συνάρτηση JavaScript καλείται με έναν από τους ακόλουθους τρόπους.

- Με ένα event (ο χρήστης έκανε κλικ σε κάποιο element)
- Κλήση από τον JavaScript κώδικα
- Αυτόματα (self invoked)

Η δημιουργία των συναρτήσεων μας επιτρέπει επαναχρησιμοποίηση κώδικα. Μια συνάρτηση JavaScript μπορεί να χρησιμοποιηθεί ως τιμή μιας μεταβλητής.

var checker;

checker = function_check(param);

Αντικείμενα

Τα αντικείμενα της JavaScript μπορούμε να τα παρομοιάσουμε με μεταβλητές οι οποίες διαθέτουν πολλαπλές τιμές.

πχ var phone = {type: “smartphone”, brand: “LG”, model: “G5”};

Οι τιμές γράφονται ως ζεύγη name: value δηλαδή το όνομα της τιμής και η τιμή της με : ανάμεσά τους. Το ζεύγος name: value ονομάζεται property. Για να έχουμε πρόσβαση στα properties ενός αντικειμένου αρκεί να γράψουμε objectName.propertyName ή objectName[“propertyName”]. πχ phone.type

Επίσης μπορούμε να αποκτήσουμε πρόσβαση στις μεθόδους ενός αντικειμένου με τον

ακόλουθο τρόπο: objectName.methodName();

[23][27]

4.4 JSON (JavaScript Object Notation)

Το JSON είναι μια μορφή για ανταλλαγή δεδομένων ή μηνυμάτων, κατανοητή από τους ανθρώπους και εύκολα δημιουργείται από τους υπολογιστές. Το JSON είναι ένα παράγωγο της JavaScript, έχει μορφή κειμένου και είναι ανεξάρτητο γλώσσας προγραμματισμού, με αποτέλεσμα να είναι ιδανικό μέσο για ανταλλαγή δεδομένων.

Δομή JSON

Το JSON δομείται με δύο τρόπους.

- Με συλλογή ζευγών όνομα/τιμή. Στις περισσότερες γλώσσες προγραμματισμού, αυτό ονομάζεται αντικείμενο ή struct.
- Με ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό ονομάζεται πίνακας ή λίστα.

Σχεδόν όλες οι γλώσσες προγραμματισμού υποστηρίζουν αυτές τις δομές με τον ένα ή τον άλλο τρόπο. Έτσι είναι απόλυτα λογικό μια μορφή δεδομένων που είναι σχεδιασμένη για ανταλλαγή, να βασίζεται σε αυτές τις δομές.

Ένα JSON αντικείμενο έχει της ακόλουθη μορφή.

αντικείμενο { name1: value1, name2: value2 }

Ένας JSON πίνακας αποτελέιται από μια ταξινομημένη λίστα τιμών και έχει την ακόλουθη μορφή.

array [value1, value2]

Ανταλλαγή δεδομένων

Κατά την ανταλλαγή δεδομένων μεταξύ του client και του server, τα δεδομένα πρέπει να είναι σε text μορφή. Μπορούμε να μετατρέψουμε οποιοδήποτε JavaScript αντικείμενο σε JSON και να στείλουμε το JSON στο server. Αντίστοιχα μπορούμε να μετατρέψουμε το JSON που λάβαμε από το server σε JavaScript αντικέιμενο ώστε να επεξεργαζόμαστε τα δεδομένα ως JavaScript χωρίς πολύπλοκες μετατροπές.

πχ var jsobj = {“name”: “John”, “age”: 25, “city”:”Athens”};

var JSONobj = JSON.stringify(jsobj);

Έτσι μετατρέψαμε ένα JavaScript αντικέιμενο σε JSON.

Για να μετατρέψουμε ένα JSON που λάβαμε από το server, σε JavaScript αντικέιμενο χρησιμοποιούμε την JSON.parse()

var myjson = '{"name": "John", "age": 25}';

var obj = JSON.parse(myjson);

Αποθήκευση δεδομένων

Η αποθήκευση δεδομένων JSON συνήθως γίνεται σε μορφή text. Με την JSON μπορούμε να αποθηκεύσουμε JavaScript αντικείμενα στο local storage. πχ

myObj = {"name": "John", "age": 25, "city": "Athens"}

myJSON = JSON.stringify(myObj);

localStorage.setItem("JSONtest", myJSON);

Για να ανακτήσουμε τα δεδομένα που είναι αποθηκευμένα, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση της JavaScript την JSON.parse() και να τα μετατρέψουμε σε JavaScript αντικείμενα.

text = localStorage.getItem("JSONtest");

obj = JSON.parse(text);

name = obj.name;

Όπως φαίνεται η JSON έχει όλα εκείνα τα στοιχεία που την κάκνουν ιδανική για ανταλλαγή δεδομένων. Χρησιμοποιεί JavaScript σ΄΄νταξη, αλλά τα JSON δεδομένα είναι σε μορφή text πάντα. Έτσι μπορούν να διαβαστούν από όλες σχεδόν τις γλώσσες προγραμματισμού.

Αποστολή αιτήματος

Μπορούμε να στείλουμε ένα AJAX αίτημα στο server για να λάβουμε το JSON. Στη συνέχεια με την JSON.parse() θα μετατρέψουμε το JSON σε JavaScript αντικείμενο.

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function(){

if (this.readyState == 4 && this.status == 200){

```
        myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
        }
    };
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

Όταν χρησιμοποιούμε την JSON.parse σε JSON το οποίο περιέχει πίνακα, η μέθοδος θα επιστρέψει έναν JavaScript πίνακα αντί για αντικείμενο.

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function(){
    if (this.readyState == 4 && this.status == 200){
        myArr = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myArr[0];
        }
    };
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

Η μέθοδος JSON.parse υποστηρίζεται σχεδόν σε όλους σύγχρονους browsers και έχει προτυποποιηθεί στην τελευταία έκδοση της JavaScript.

Με την JSON μπορούμε να στείλουμε δεδομένα στο server, αρκεί να είναι σε text μορφή. Για να μετατρέψουμε ένα JavaScript αντικείμενο σε JSON χρησιμοποιούμε την JSON.stringify()

Χρήση

Έχουμε ένα JavaScript αντικείμενο

```
var obj = {"name": "John", "age": 30, "city":"Athens"};
```

Για να το μετατρέψουμε σε JSON string αρκεί να κάνουμε

var converted = JSON.stringify(obj);

Η μέθοδος JSON.stringify υποστηρίζεται σχεδόν σε όλους τους σύγχρονους browsers και έχει προτυποποιηθεί στην τελευταία έκδοση της JavaScript.

4.4.1 JSON με PHP

Η JSON μορφή μπορεί να χρησιμοποιηθεί από πληθώρα γλωσσών προγραμματισμού. Συγκεκριμένα η PHP διαθέτει μεθόδους για τη χρήση JSON. Η μέθοδος json_encode() μετατρέπει τα php αντικείμενα σε JSON μορφή. Έτσι μπορούμε εύκολα να παράγουμε JSON σε ένα web server. Ο client με την JSON.parse() της JavaScript θα διαβάσει το JSON που έλαβε από τον web server. Επίσης με την json_encode() μπορούμε να μετατρέψουμε και τους php πίνακες σε JSON.

Μετατροπή PHP αντικειμένου σε JSON

Έστω το αντικείμενο car

```
<?php

$car -> brand = “ford”;

$car -> model = “docus”;

$car -> fuel = “diesel”;
```

Η json_encode() θα μετατέψει το php αντικείμενο σε JSON.

```
$myJSON = json_encode($car);

echo $myJSON;

?>
```

Κλήση PHP αρχείου

Για να καλέσουμε αυτό το php αρχείο θα στείλουμε ένα AJAX αίτημα από τον client στο server

```
<script>

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

        if (this.readyState == 4 && this.status == 200) {
```

```
        //μετατροπή του JSON σε JavaScript αντικείμενο
        var myObj = JSON.parse(this.responseText);
        }
    };
xmlhttp.open("GET", "demo_file.php", true);
xmlhttp.send();
</script>
```

Η PHP διαθέτει τη μέθοδο json_decode() με την οποία μετατρέπουμε ένα JSON string σε php αντικείμενο ή μεταβλητή. Έτσι έχουμε τη δυαντότητα να λάβουμε JSON σε ένα web server και να το χρησιμοποιήσουμε στην PHP. Για παράδειγμα μπορούμε να στείλουμε ένα AJAX αίτημα στον server το οποίο θα μας επιστρέψει δεδομένα από μια βάση δεδομένων. Θα χρησιμοποιήσουμε το JSON ως παράμετρο για το query στη βάση δεδομένων.

```
<script>
obj = {"table": "cars", "limit": 10};
dbparam = JSON.stringify(obj); //Μετατρέπουμε το αντικείμενο σε JSON string για να το στείλουμε στο server.
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function(){
    if (this.readyState == 4 && this.status == 200){
        var response = this.responseText;
        }
    };
xmlhttp.open("GET", "json_demo_db.php?x="+dbparam, true);
xmlhttp.send();
</script>
```

Στέλνουμε το αίτημα στο αρχείο php του server με το JSON string ως παράμετρο. Το αποτέλεσμα επιστρέφεται ως JSON. Στο αρχείο PHP το οποίο βρίσκεται στο server θα

χρησιμοποιήσουμε την παράμετρο που εστάλη στο αίτημα, αφού πρώτα τη μετατρέψουμε σε php αντικείμενο με την json_decode()

```
<?php

header (“Content-Type: application/json; charset = UTF-8”);

$obj = json_decode($_GET[“x”], false); //μετατρέπουμε το JSON string που περιέχεται στην παράμετρο X, σε php αντικείμενο.

$conn = new mysqli(“Server”, “userdb”, “passworddb”, “database”);

$result = $conn -> query(“SELECT BRAND FROM”.$obj->table.“LIMIT”.$obj->limit);

$output = array();

$output = $result -> fetch_all(MYSQL_ASSOC); //Βάζουμε τον πίνακα σε αντικείμενο το οποίο στη συνέχεια θα επιστραφεί ως JSON.

echo json_encode($output); //Το php αρχείο επιστρέφει το αποτέλεσμα του select σε JSON το οποίο λαμβάνει ο client.
```

Στον client θα μετατρέψουμε το αποτέλεσμα που λάβαμε από το php αρχείο, σε JavaScript αντικείμενο. Στη συγκεκριμένη περίπτωση σε πίνακα.

```
<script>

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200){
                myObj = JSON.parse(this.responseText);
                for (x in myObj) {
                        txt += myObj[x].name +”<br>”;
                        }
                document.getElementById(“element_id”).innerHTML = txt;
                }
        };
xmlhttp.open(“GET”, “json_demo_db.php?x=”+dbparam, true);
```

```
xmlhttp.send();
</script>
```

[24],[25],[26]

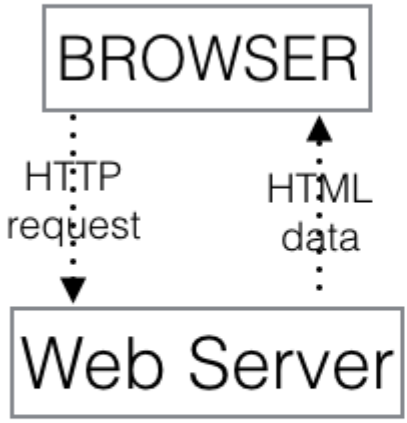
4.5 AJAX

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, η AJAX δεν αποτελεί γλώσσα προγραμματισμού ή μια νέα τεχνολογία. Πρόκειται για ένα σύνολο τεχνικών οι οποίες ενσωματώνουν:

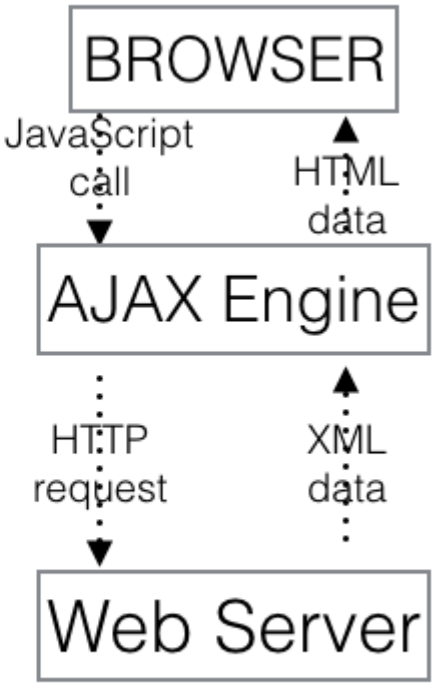
- Απλή προβολή με HTML και CSS
- Δυναμική προβολή και αλληλεπίδραση με το Document Object Model (DOM)
- Ανταλλαγή δεδομένων με XML
- Ασύγχρονη λήψη δεδομένων με το XMLHttpRequest
- JavaScript για να συνδέσει όλα τα παραπάνω.

Στο κλασικό μοντέλο web εφαρμογών, οι ενέργειες του χρήστη προκαλούν την αποστολή ενός HTTP αιτήματος στο server. Ο server μετά την απαραίτητη επεξεργασία, επιστρέφει μια HTML σελίδα στον client. ΤΟ πρόβλημα με αυτό το μοντέλο είναι ότι για το χρονικό διάστημα κατά το οποίο ο server εκτελεί κάποια ενέργεια, ο client είναι αναγκασμένος να περιμένει. Μια AJAX εφαρμογή όμως, εξαλείφει την ανάγκη να περιμένει ο χρήστης για κάθε ενέργεια που προκαλεί στο server. ΓΙα να το πετύχει αυτό εισάγει ένα επιπλέον μέσο ανάμεσα στο χρήστη και το server. ΤΟ μέσο αυτό ονομάζεται AJAX engine. Στην αρχή ενός session σε μια AJAX εφαρμογή, ο browser φορτώνει την AJAX engine η οποία είναι γραμμένη σε JavaScript. Η AJAX engine απεικονίζει το user interface και αναλαμβάνει την επικοινωνία με το server. Δίνει τη δυνατότητα η αλληλεπίδραση του χρήστη με την εφαρμογή ναι γίνεται ασύγχρονα και ανεξάρτητα από την επικοινωνία με το server. Έτσι ο χρήστης δε θα χρειαστεί να περιμένει όσο ο server εκτελεί λειτουργίες.

Στο παρακάτω σχήμα βλέπουμε τον τρόπο λειτουργίας των κλασικών και των AJAX εφαρμογών.



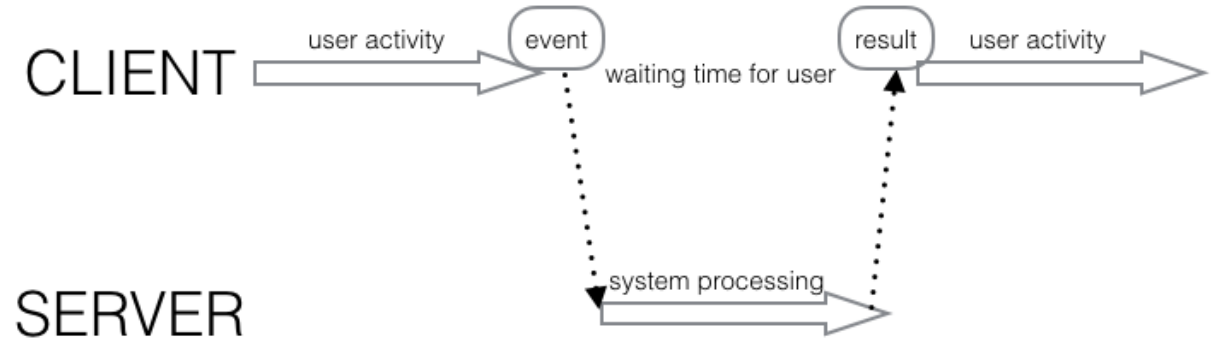
κλασικό μοντέλο web εφαρμογών

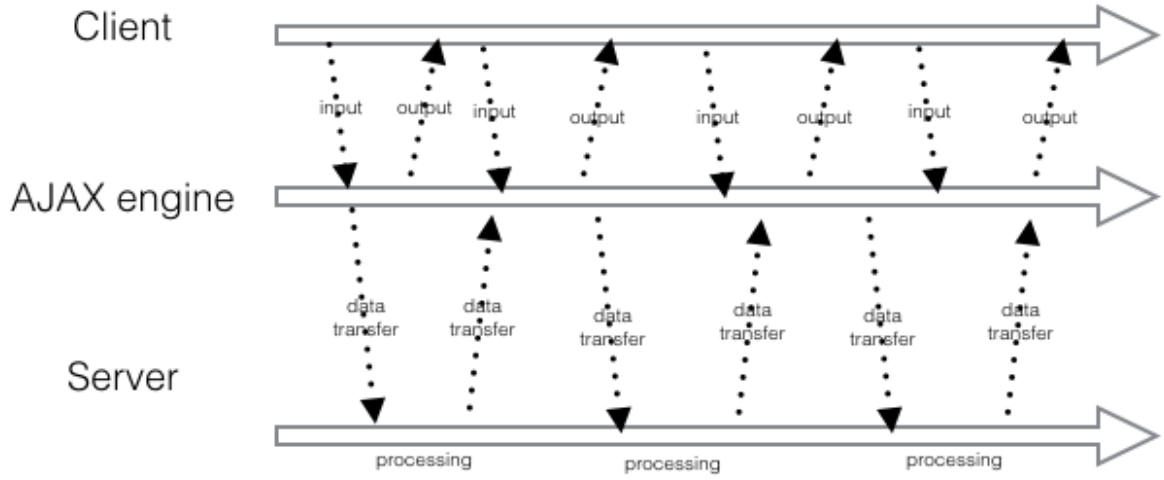


AJAX μοντέλο web εφαρμογών

Σε τι όμως διαφέρει πραγματικά μια AJAX εφαρμογή από τις κλασικές; Η διαφορά τους συνοψίζεται στην ασύγχρονη μετάδοση δεδομένων. Πιο συγκεκριμένα σε μια web εφαρμογή ο χρήστης προκαλεί κάποιο event, πχ έστο στο δικό μας σενάριο, click για αποστολή μιας φόρμας. Ο browser του χρήστη θα στείλει ένα αίτημα στο server στέλνοντας τα δεδομένα της φόρμας. Για όσο χρόνο στέλνονται τα δεδομένα και ο server τα επεξεργάζεται, πχ για validation, ο χρήστης δεν μπορεί να αλληλεπιδράσει με την εφαρμογή, παρά μόνο όταν ολοκληρωθεί η διαδικασία και ο χρήστης μεταφερθεί στην επόμενη σελίδα που θα τον ενημερώνει για επιτυχημένη αποστολή της φόρμας. Χρειάστηκε δηλαδή να περιμένει να φορτωθεί η νέα σελίδα για να πληροφορηθεί για το εάν η φόρμα του εστάλη επιτυχώς. Σε μια AJAX εφαρμογή όμως, ο client μπορεί να λάβει απάντηση από το server για το validation της φόρμας, καθώς ο χρήστης την συμπληρώνει. Μπορεί δηλαδή να αλληλεπιδρά με την εφαρμογή, ενώ στο παρασκήνιο ο client στέλνει και λαμβάνει δεδομένα από το server. Στα παρακάτω σχήματα φαίνεται η παραπάνω διαδικασία.

Στο πρώτο σχήμα βλέπουμε τη σύγχρονη επικοινωνία μεταξύ client και server, ενώ στο δεύτερο την ασύγχρονη.





Στην ασύγχρονη επικοινωνία βλέπουμε ακριβώς ότι ο χρήστης μπορεί να αλληλεπιδράσει με την εφαρμογή πριν ο server ολοκληρώσει τις όποιες εργασίες του.

Σήμερα οι περισσότερες εφαρμογές που χρησιμοποιούμε στο ίντερνετ είναι εφαρμογές AJAX. Μερικές από τις δημοφιλέστερες είναι το Gmail, οι χάρτες Google Maps, το Google suggest στην αναζήτηση της Google και πολλές άλλες.

[28]

XMLHttpRequest object

Η βάση του AJAX είναι το αντικείμενο XMLHttpRequest. Χρησιμοποιείται για την ανταλλαγή δεδομένων με το server στο παρασκήνιο. Με αυτόν τον τρόπο είναι δυνατόν να

ανανεωθούν τμήματα μιας σελίδας, χωρίς να χρειάζεται να επαναφορτωθεί ολόκληρη η σελίδα. Όλοι σχεδόν οι σύγχρονοι browsers υποστηρίζουν το XMLHttpRequest object. Η σύνταξή του έχει ως εξής.

var xmlhttp = new XMLHttpRequest();

Με τη new HMLHttpRequest() δημιουργείται ένα νέο XMLHttpRequest αντικείμενο, αλλά για ένα ολοκληρωμένο AJAX αίτημα πρέπει να χρησιμοποιήσουμε και άλλες μεθόδους του αντικειμένου.

Η open() παίρνει ως παραμέτρους, τη μέθοδο του αιτήματος δηλαδή GET η POST, το url δηλαδή τη διεύθυνση στην οποία βρίσκεται το αρχείο το οποίο χρησιμοποιούμε στο αίτημα, true ή false για το αν το αίτημα θα πραγματοποιηθεί με ασύγχρονο ή σύγχρονο τρόπο και προαιρετικές παραμέτρους, username & password.

open(method, ulr, async, username, password)

Η send() στέλνει το αίτημα στο server. Χρησιμοποιείται όταν το αίτημα γίνεται με GET. Όταν το αίτημα γίνεται με POST χρησιμοποιούμε την send(string)

Οι παραπάνω μέθοδοι είναι απαραίτητες για ένα AJAX αίτημα, αλλά το XMLHttpRequest διαθέτει και άλλες. Με τα properties του αντικειμένου μπορούμε να το επεξεργαστούμε. Η onreadystatechange θα καλέσει μια συνάρτηση όταν η ιδιότητα readystate αλλάξει. Η readystate αποθηκεύει την κατάσταση του XMLHttpRequest.

Οι καταστάσεις του XMLHttpRequest μπορεί να είναι:

- 0: όταν το request δεν έχει αρχικοποιηθεί
- 1: Όταν εδραιώνεται η σύνδεση με τον server
- 2: Όταν ο server λαμβάνει το αίτημα
- 3: Όταν το αίτημα επεξεργάζεται στο server
- 4: ΤΟ αίτημα ολοκληρώθηκε και η απάντηση είναι έτοιμη.

Το responseText επιστρέφει τα δεδομένα της απάντησης ως string.

Το responseXML επιστρέφει τα δεδομένα της απάντησης ως XML.

Το status επιστρέφει έναν αριθμό κατάστασης του αιτήματος. Όταν ο αριθμός είναι 200 σημαίνει ‘ΟΚ’. Το 403 σημαίνει απαγορευμένο και το 404 σημαίνει δε βρέθηκε. Επίσης υπάρχει και το statusText το οποίο επιστρέφει ‘ΟΚ’ ή ‘Not Found’.

Για να στείλουμε το αίτημα στο server χρησιμοποιούμε τις μεθόδους open() και send() του XMLHttpRequest object.

xmlhttp.open(“GET”,”ajax_file.txt”, true);

xmlhttp.send();

Μέθοδοι GET & POST

Σε ένα απλό αίτημα με την GET

xmlhttp.open(“GET”,”file.php”, true);

xmlhttp.send();

είναι πιθανό να πάρουμε ως απάντηση ένα cashed αποτέλεσμα. Για να το αποφύγουμε αυτό μπορούμε να προσθέσουμε ένα μοναδικό ID στο URL του αρχείου.

xmlhttp.open(“GET”,”file.php?t=”+Math.random(), true);

xmlhttp.send();

Επίσης μπορύμε να στείλουμε πληροφορίες στο url.

xmlhttp.open(“GET”,”file.php?name=John&age=25”, true);

xmlhttp.send();

Ένα απλό αίτημα με την POST μέθοδο γίνεται ως εξής.

xmlhttp.open(“POST”,”file.php”, true);

xmlhttp.send();

Σε ένα POST αίτημα για να στείλουμς δεδομένα ώπος θα στέλναμε σε μια HTML φόρμα, θα πρέπει να ορίσουμε έναν HTTP header καθώς και τα δεδομένα που θέλουμε, στη μέθοδο send().

xmlhttp.open(“POST”,”file.php”, true);

//ορίζουμε http header

xmlhttp.setRequestHeader(“Content-type”,”application/x-www-form-urlencoded”);

xmlhttp.send(“name=John&age=25”);

Asynchronous - True or False

Για να στείλουμε ένα αίτημα ασύγχρονα, η παράμετρος async της μεθόδου open()

πρέπει να είναι true. Το να στέλνουμε αιτήματα ασύγχρονα ήταν μια τρομερή καινοτομία για τον δικτυακό προγραμματισμό. Χωρίς αυτή τη δυνατότητα, η εφαρμογή θα πάγωνε. Με το ασύγχρονο αίτημα, η JavaScript δε χρειάζεται να περιμένει την απάντηση του server, αλλά μπορεί να εκτελεί άλλες λειτουργίες όσο χρόνο χρειάζεται για να λάβει απάντηση και να την επεξεργαστεί όταν θα είναι έτοιμη. Όταν λοιπόν η παράμετρος async είναι true, ορίζουμε μια συνάρτηση που θα κληθεί όταν προκύωει onreadystatechange event.

```
xmlhttp.onreadystatechange = function() {
        κώδικας
    }
```

Εάν δε θέλουμε ασύγχρονη απαστολή του αιτήματος, ορίζουμε την async παράμετρο ως false. Σε αυτή την περίπτωση η JavaScript θα σταματήσει να εκτελείται μέχρι να λάβει απάντηση από τον server. Έτσι αν ο server είναι απασχολημένος, η εφαρμογή θα κολλήσει. Όταν χρησιμοποιούμε async = false δεν πρέπει να ορίσουμε συνάρτηση στην onreadystatechange. Ο κώδικας θα συνεχιστεί μετά τη send().

onreadystatechange

Η ιδιότητα readyState περιέχει την κατάσταση του XMLHttpRequest. Η onreadystatechange ορίζει τη συνάρτηση που θα εκτελεστεί όταν το readyState αλλάξει. Τα status και statusText περιέχουν την κατάσταση του XMLHttpRequest αντικκειμένου. Η συναρτηση στην onreadystatechange καλείται κάθε φορά που αλλάζει το readyState. Η απάντηση είναι έτοιμη όταν το readyState γίνει 4 και το status 200.

```
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        //η απάντηση είναι έτοιμη
        //do something
    }
};
```

Επομένως σε κάθε αίτημα προκαλείται onreadystatechange event 4 φορές. Η ιδιότητα responseText επιστρέφει την απάντηση του server ως JavaScript string.

```
var response = xmlhttp.responseText;
```

Η ιδιότητα respnseXML επιστρέφει την απάντηση του server ως XML DOM αντικείμενο

```
xmlfile = xmlhttp.responseXML;
txt = " ";
x = xmlhttp.getElementByTagName("CARS");
for (i=0; i<x.length; i++) {
        txt += x[i].childNodes[0].nodeValue + "<br>";
        }
document.getElementVyId("demo").innerHTML = txt;
xmlhttp.open("GET", "file.xml", true);
xmlhttp.send();
```

4.6 PHP: Hypertext Preprocessor

Η γλώσσα προγραμματισμού PHP είναι ο ακρογωνιαίος λίθος για την ανάπτυξη του δυναμικού προγραμματισμού στο διαδίκτυο. Με την PHP ένας web server μπορεί να παράξει δυναμικό περιεχόμενο, δύναται να είναι διαφορετικό κάθε φορά που ένας browser αιτείται μια σελίδα. Η PHP είναι μια scripting γλώσσα εύκολα κατανοητή, αλλά με τεράστιες δυαντότητες.

Τα PHP αρχεία έχουν την κατάληξη .php επομένως όταν ο server συναντήσει αυτή την κατάληξη σε ένα αρχείο που του ζητήθηκε, αυτόματα το περνά στον php processor. Το PHP πρόγραμμα είναι υπεύθυνο να επιστρέψει ένα κατάλληλο αρχείο για προβολή σε web browser. Για να ενεργοποιηθούν οι PHP εντολές στον web server χρησιμοποιούμε το php tag. Μπορούμε να έχουμε ένα ολόκληρο πρόγραμμα PHP μέσα σε php tags.

```
<?php //tag ανοίγματος
        //php κώδικας
?> // tag κλεισίματος
```

4.6.1 Δομή PHP

- Στην PHP μπορούμε να δηλώσουμε ένα τμήμα κώδικα σε σχόλια με δύο τρόπους.
  // σχόλιο μιας γραμμής

  /* σχόλια

πολλαπλών

γραμμών */

- Οι εντολές στην PHP πρέπει να έχουν ; στο τέλος τους.
- Οι μεταβλητές δηλώνονται με το σύμβολο $ και το όνομα της μεταβλητής.
  $name = “ilias”;

  Όλοι οι τύποι μεταβλητών της PHP δηλώνονται με το σύμβολο $.

  $myname = “ilias”;

  $myage = 25;

  $myarray = array ($myname, $myage);

- Τα ονόματα των μεταβλητών πρέπει να ξεκινάνε με χαρακτήρα του αλφαβήτου ή underscore (_). Μπορούν να περιέχουν χαρακτήρες a-z, A-Z, αριθμούς 0-9 και underscore(_). Τα ονόματα είναι case sensitive, δηλαδή η μεταβλητή $name είναι διαφορετική από την $Name. Απαγορεύεται να περιέχουν κενό - space.
- Οι operators στην PHP είναι αρκετά απλοί. Η PHP διαθέτει τους ακόλουθους αριθμητικούς operators.

+	ΠΡΌΣΘΕΣΗ
-	ΑΦΑΙΡΕΣΗ
*	ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ
/	ΔΙΑΙΡΕΣΗ
%	ΥΠΟΛΟΙΠΟ
++	increment
--	decrement

- Η ανάθεση γίνεται με τον ακόλουθο τρόπο: variable = value;
  Ο operator = αναθέτει την τιμή στα δεξιά του, στη μεταβλητή στα αριστερά του. Ο operator += προσθέτει τη μεταβλητή στα δεξιά του, στη μεταβλητή στα αριστερά του.

  $i = 3;

  $i += 1;

Το $i θα έχει την τιμή 4

- Οι μεταβλητές τύπου string έχουν το δικό τους operator, την περίοδο (.) το οποίο χρησιμοποιείται για string concatenation και θα αναλυθεί αργότερα.
- Οι operators σύγκρισης χρησιμοποιούνται όταν θέλουμε να συγκρίνουμε μεταβλητές όπως σε μια if συνθήκη.
  Οι operators σύγκρισης είναι οι ακόλουθοι.

==	είναι ίσο με
!=	δεν είναι ίσο με
>	είναι μεγαλύτερο από
<	είναι μικρότερο από
>=	είναι μεγαλύτερο ή ίσο
<=	είναι μικρότερο ή ίσο

- Η PHP διαθέτει και logical operators τους οποίους μπορούμε να χρησιμοποιήσουμε σε μια συνθήκη. Οι logical operators είναι οι ακόλουθοι.

&&	And
and	and χαμηλής προτεραιότητας
||	or
or	or χαμηλής προτεραιότητας
!	Not πχ if (!($i==0))
xor	exclusive or

πχ. if ($age > 18 && $age < 30) η συνθήκη είναι αληθής για $age μεταξύ 19 και 29.

- **String Concatenation**
  Για τη συνένωση δύο string χρησιμοποιούμε την (.)

  echo “Your age is ” .$age. “and is under the age limit”;

Μπορούμε να ενώσουμε δύο μεταβλητές τύπου string με τον ακόλουθο τρόπο.

$str_1 .= $str_2;

Έστω ότι το $str_1 έχει την τιμή string1 και το $str_2 έχει την τιμή string2. Με τη συνένωση το $str_1 θα έχει την τιμή string1string2.

Στην PHP αν έχουμε ένα string σε μονά αυτάκια ‘ ’ τότε δθατηρείται ως έχει.

πχ $str = ‘variable $age should not be null’;

echo $str; //τυπώνει variable $age should not be null

‘Εστω τώρα $str = “variable $age should not be null”;

echo $str;

Ενώ στην πρώτη περίπτωση θα τυπωθούν ακριβώς οι χαρακτήρες του $str, στη δεύτερη η PHP θα προσπαθήσει να τυπώσει το $age ως μεταβλητή, δηλαδή να τυπώσει την τιμή της. Για να πούμε στην PHP να χειρισθεί ορισμένους χαρακτήρες κυριολεκτικά και να μη τους μεταφράσει, χρησιμοποιούμε το (\) backslash.

πχ $str = ‘my car’s color is black’;

Η απόστροφος στο car’s θα δημιουργήσει πρόβλημα καθώς ο PHP processor θα νομίζει ότι το string τελειώνει εκεί. Για να μη γίνει αυτό, βάζουμε $str = ‘my car\’s color is black’;

echo $str; και θα τυπωθεί κανονικά το my car’s color is black.

- **Συναρτήσεις:**
  Όπως και στις άλλες γλώσσες προγραμματισμού, έτσι και στην PHP η δημιουργία συναρτήσεων μας προσφέρει καλύτερη οργάνωση και επαναχρησιμοποίηση κώδικα. Γίνεται πιο εύκολα συντηρήσιμος και σε μερικές περιπτώσεις προσφέρει έξτρα λειτουργικότητα. Για να δημιουργήσουμε μια php συνάρτηση τη δηλώνουμε με τον ακόλουθο τρόπο:

  <?php

  function myFunction($param) {

  function code

  }

  Οι μεταβλητές της PHP έχουν scope μέσα στη συνάρτηση στην οποία δηλώνονται. Αυτές λέγονται τοπικές μεταβλητές. Αν όμως θέλουμε μια μεταβλητή να έχει scope σε

ολόκληρο το php πρόγραμμα τη δηλώνουμε με το keyword global.

global $username;

Αυτές οι μεταβλητές ονομάζονται global μεταβλητές. Οι τοπικές μεταβλητές καταστρέφονται αφού η συνάρτηση ολοκληρώσει την εκτέλεσή της. Η PHP μας δίνει τη δυνατότα να διατηρήσουμε την τιμή μιας τοπικής μεταβλητής δηλώνοντάς την ως static.

static $counter;

Έτσι στην επόμενη εκτέλεση της συνάρτησης, η μεταβλητή θα έχει αρχικά την τιμή που είχε στην προηγούμενη εκτέλεση. Ένα σενάριο στο οποίο χρειαζόμαστε αυτή τη δυνατότητα είναι όταν θέλουμε να κρατάμε τον αριθμό των εκτελέσεων μιας συνάρτησης. Δηλώνοντας τη μεταβλητή ως static και όχι ως global, έχει local scope αλλά διατηρεί την τιμή της.

Η εκδόσεις PHP 4.1 και μετά, διαθέτουν super global μεταβήτές. Αυτές οι μεταβλητές παρέχονται από την php και είναι διαθέσιμες σε όλο το πρόγραμμα. Έχουμε πρόσβαση στης super global μεταβλητές από παντού. Παρέχουν χρήσιμες πληροφορίες για το πρόγραμμα που τρέχει και δομούνται σε πίνακες. Οι super global μεταβλητές είναι οι ακόλουθες.

$GLOBALS	Περιέχει όλες τις μεταβλητές που είναι σε global scope. Τα ονόματα των μεταβλητών χρησιμοποιούνται ως κλειδιά για τον πίνακα.
$_SERVER	Περιέχει πληροφορίες όπως headers, paths και τοποθεσίες. Ο server δημιουργεί αυτές τις πληροφορίες, αλλά δεν είναι σίγουρο ότι όλοι οι server θα παρέχουν τον ίδιο αριθμό πληροφοριών.
$_GET	Περιέχει τις μεταβλητές που εστάλησαν με HTTP GET μέθοδο
$_POST	Περιέχει τις μεταβλητές που εστάλησαν με HTTP POST μέθοδο
$_FILES	Περιέχει τα στοιχεία που φορτώθηκαν με HTTP POST μέθοδο
$_COOKIE	Περιέχει τις μεταβλητές που εστάλησαν με HTTP cookies
$_SESSION	περιέχει τις μεταβλητές session διαθέσιμες στο php πρόγραμμα
$_REQUEST	Περιέχει πληροφορίες που εστάλησαν στο server από τον browser. Από default είναι $_GET, $_POST, $_COOKIE.

$_ENV	Περιέχει μεταβλητές που εστάλησαν στο πρόγραμμα με την environment μέθοδο

4.6.2 Αντικείμενα

Ενώ οι συναρτήσεις ήδη όπως είδαμε προσφέρουν ευκολίες στον προγραμματισμό, ο αντικειμενοστρεφής προγραμματισμός πολλαπλασιάζει τη χρηστικότητά τους. Δηλαδή η ομαδοποίηση των συναρτήσεων και των δεδομένων τους σε αντικείμενα, διευκολύνει σε μεγάλο βαθμό την οργάνωση, συντήρηση και επαναχρησιμοποίηση του κώδικα.

Για να δημιουργήσουμε ένα πρόγραμμα στο οποίο θα χρησιμοποιούμε αντικείμενα, πρέπει να οργανώσουμε μια σύνθεση δεδομένων και συναρτήσεων που ονομάζεται κλάση. Κάθε αντικείμενο που βασίζεται στην κλάση ονομάζεται instance της κλάσης. Τα δεδομένα τα οποία σχετίζονται με ένα αντικείμενο ονομάζονται ιδιότητες (properties) και οι συνερτήσεις του, μέθοδοι (methods). Στον αντικειμενοστρεφή προγραμματισμό όταν δημιουργούμε αντικείμενα, η σωστή πρακτική είναι να χρησιμοποιούμε ενθυλάκωση (encapsulation). Δηλαδή να φτιάχνουμε την κλάση με τέτοιο τρόπο ώστε μόνο οι μέθοδοι της κλάσης αυτής να έχουν πρόσβαση στις ιδιότητες του αντικειμένου. Αυτή η πρακτική προσφέρει εύκολη αποσφαλμάτωση του κώδικα, καθώς ολόκληρος ο κώδικας του αντικειμένου θα βρίσκεται μόνο στην κλάση.

Στην περίπτωση που έχουμε δημιουργήσει μια κλάση και χρειαζόμαστε άλλη μια η οποία μοιάζει με την προηγούμενη μπορούμε να δημιουργήσουμε τη νέα κλάση χρησιμοποιώντας κληρονομικότητα (inheritance). Έτσι η νέα κλάση θα διαθέτει όλες τις ιδιότητες της κλάσης από την οποία κληρονόμησε. Η αρχική κλάση ονομάζεται superclass και η νέα, subclass.

Δήλωση κλάσης

Για να δηλώσουμε μια κλάση χρησιμοποιούμε το keyword class και το όνομα της κλασης το οποίο είναι case sensitive. Η δήλωση της κλάσης περιλαμβάνει εκτός από το όνομα, τις ιδιότητες και τις μεθόδους της.

```
class Car

{

        public $brand, $model;

        function create_car()
```

```
    {
        code to create a car
    }
}
```

Για να δημιουργήσουμε ένα αντικείμενο της παραπάνω κλάσης χρησιμοποιούμε το keyword new.

$car1 = new Car;

Αποκτάμε πρόσβαση στις ιδιότητες του αντικειμένου με τον παρακάτω τρόπο

$object->property() δηλαδή $car1->brand = “Ford”;

Αποκτάμε πρόσβαση στις μεθόδους το αντικειμένου με τον παρακάτω τρόπο

$object->method() δηλαδή $car1->create_car();

Constractor

Όταν δημιουργούμε ένα αντικείμενο μπορούμε να περάσουμε παραμέτρους στην κλάση του. Στην κλάση υπάρχει μια ειδική μέθοδος η οποία ονομάζεται κατασκευαστής (constractor) και αρχικοποιεί τις ιδιότητες του αντικειμένου. Σε παλαιότερες εκδόσεις PHP η μέθοδος κατασκευαστής δηλωνόταν με το όνομα της κλάσης, δηλαδή:

```
class Car
{
    public function Car(param1, param2)
    {
    constractor code
    $brand = “Ford”;
    }
}
```

Η PHP 5 ορίζει διαφορετική δήλωση του κατασκευαστή.

```
class Car
{
```

```
        public function __construct(param1, param2)
        {
        constractor code
        $brand = “Ford”;
        }
}
```

$this

Με την ειδική μεταβλητή $this έχουμε πρόσβαση στις ιδιότητες του αντικειμένου στο οποίο βρισκόμαστε.

```
class Car
{
        public $brand, $model;
        function create_car()
        {
                return $this->model;
        }
}
```

Η μέθοδος create_car() χρησιμοποιεί τη μεταβλητή $this για να αποκτήσει πρόσβαση στο αντικείμενο και να επιστρέψει την τιμή της ιδιότητας model του αντικειμένου. Σε αυτή την περίπτωση το σύμβολο $ παραλείπεται από την ιδιότητα model όταν χρησιμοποιούμε τον -> operator.

Αρχεία PHP: include & require

Κατά την κατασκευή προγραμμάτων PHP είναι πιθανό να χρειαστούμε συναρτήσεις οι οποίες έχουν ήδη κατασκευαστεί, επομένως δε χρειάζεται να τις ξαναγράψουμε στον κώδικά μας. Μπορούμε να τις φορτώσουμε από εξωτερικά αρχεία και να τις χρησιμοποιήσουμε με τον ίδιο τρόπο όπως αν τις είχαμε αντιγράψει στον κώδικά μας. Αυτό επιτυγχάνεται με τις δηλώσεις include και require.

Με τη χρήση της include δηλώνουμε στην PHP να φορτώσει όλα τα δεδομένα ενός αρχείου. Η εντολή γράφεται ως εξής:

```
<?php

include “external_file.php”;

code to follow

?>
```

Κάθε φορά που περιέχουμε την εντολή include στον κώδικά μας, το αρχείο φορτώνεται ξανά ακόμα και αν το έχουμε ήδη φορτώσει πιο πάνω. Για παράδειγμα εάν έχουμε κάνει include το αρχείο file1.php και στη συνέχεια του προγράμματός μας κάνουμε include το αρχείο file2.php το οποίο όμως περιέχει με τη σειρά του το file1.php. Άθελά μας κάναμε include το file1.php δύο φορές, κάτι το οποίο μπορεί να δημιουργήσει προβλήματα καθώς θα υπάρχουν δηλώσεις της ίδια συνάρτησης παραπάνω από μία φορές. Η λύση σε αυτό το πρόβλημα είναι η δήλωση include_once.

```
<?php

include_once “external_file.php”;

code to follow

?>
```

Με αυτή τη δήλωση όταν η PHP συναντήσει include ή include_once για το ίδιο αρχείο ξανά, θα αναγνωρίσει ότι το έχει ήδη φορτώσει και θα το αγνοήσει.

Με τις δηλώσεις include & include_once η PHP θα προσπαθήσει να φορτώσει το αρχείο που ζητάμε. Η εκτέλεση του προγράμματος θα συνεχιστεί ακόμα και αν το αρχείο που ζητήθηκε δε βρέθηκε. Για αυτό όταν είναι απαραίτητη η χρήση του αρχείου πρέπει να χρησιμοποιήσουμε την require αντί για την include. Με την require δηλώνουμε ότι το αρχείο είναι απαραίτητο για τη συνέχιση της εκτέλεσης του προγράματος. Όπως και με την include_once έτσι και εδώ μπορούμε να χρησιμοποιήσουμε τη require_once για να αποφύγουμε πολλαπλές φορτώσεις του ίδιου αρχείου.

Scope in PHP 5

Για να ελέγξουμε το πεδίο εφαρμογής (scope) των μεθόδων και ιδιοτήτων η PHP μας παρέχει 3 keywords.

public: Όταν δηλώνουμε μια μεταβλητή ή μέθοδο στην PHP, είναι από προεπιλογή public.

protected: Οι ιδιότητες και μέθοδοι που είναι protected μπορούν να χρησιμοποιηθούν μόνο από τις μεθόδους της κλάσης και υποκλάσεων του αντικειμένου.

private: Οι ιδιότητες και μέθοδοι που είναι private μπορούν να χρησιμοποιηθούν μόνο από τις μεθόδους της ίδιας κλάσης.

```
<?php

class Car

{

        public $brand = “Ford”; //public property

        var $model = “focus” //public property but deprocated;

        protected $fuel = “deisel”;

        private function edit_car() //private method

        {

        code here

        }

}

?>
```

Κληρονομικότητα

Αφού έχουμε φτιάξει μια κλάση, μπορούμε να δημιουργήσουμε υποκλάσεις με την έννοια της κληρονομικότητας. Αυτό μας δίνει τη δυνατότητα αν έχουμε μια κλάση παρόμοια με αυτή που χρειαζόμαστε, να την επεκτείνουμε σε υποκλάση και να αλλάξουμε μόνο τα τμήματα κώδικα που χρειάζεται να είναι διαφορετικά. Η κληρονομικότητα πετυχαίνεται με τον extends operator.

```
<?php

class Car

{

public $brand, $model;

function create_car()
```

```
{
code here
}
}
class Insurance extends Car
{
public $driver, $phone;
function display()
{
echo “Owner: ” .$this->driver .”</br>”;
echo “Owner Phone: ” .$this->phone .”</br>”;
echo “Car Brand: ” .$this->brand .”</br>”;
echo “Car Model: ” .$this->model .”</br>”;
}
}
?>
```

Η κλάση Car έχει δύο ιδιότητες, τις brand & model και μια μέθοδο την create_car. Η υποκλάση Insurance επεκτείνει την Car προσθέτοντας δύο ιδιότητες, τις driver & phone και μια μέθοδο για την προβολή των δεδομένων μιας ασφάλειας χρησιμοποιώντας την ειδική μεταβλητή $this η οποία αναφέρεται στις τιμές του αντικειμένου.

4.7 MySQL

Με πάνω από δέκα εκατομμύρια installations, η MySQL αποτελεί το πιο διαδεδομένο σύστημα διαχείρησης βάσης δεδομένων για web servers. Σχεδιασμένη στα μέσα της δεκαετίας του 90 , αποτελεί μια ώριμη τεχνολογία που τροφοδοτεί πολλούς από τους σημερινούς πιο πολυσύχναστους διαδικτυακούς προορισμούς.

Ένας από τους λόγους της επιτυχίας της είναι το γεγονός ότι, όπως η PHP, είναι δωρεάν. Είναι όμως απίστευτα δυνατή και εξαιρετικά γρήγορη - μπορεί να τρέχει πάνω στο πιο βασικό/απλό hardware χωρίς να επηρεάζει σημαντικά τους πόρους του συστήματος (system resources)
Η mySQL είναι επίσης πολύ προσαρμοστική, δηλαδή μπορεί να εξελιχθεί μαζί με μια ιστοσελιδα. Για την ακρίβεια σε σύγκριση με άλλες βάσεις δεδομένων η mySQL και η Oracle είχαν την καλύτερη επίδοση και επεκτασιμότητα.

4.7.1 MySQL Basics

Η βάση δεδομένων αποτελεί μια δομημένη συλλογή αρχείων ή δεδομένων που είναι αποθηκευμένα στο σύστημα ενός υπολογιστή και είναι οργανωμένα με τέτοιο τρόπο ώστε να μπορούν να αναζητούνται γρήγορα και οι πληροφορίες να μπορούν να ανακτηθούν γρήγορα. Το SQL στο όνομα mySQL αποτελέι ακρωνύμιο του Structured Query Language (δομεμημένη γλώσσα ερωτήματος). Η γλώσσα αυτή είναι βασισμένη στα αγγλικά και χρησιμοποιείται και σε άλλες βάσεις δεδομένων όπως η Oracle και η Microsoft SQL Server. Είναι σχεδιασμένη να επιστρέφει απλά αιτήματα (requests) από την βάση δεδομένων μέσω απλών εντολών όπως:

SELECT title FROM publications WHERE author=’Charles Dickens’;

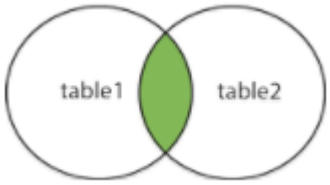
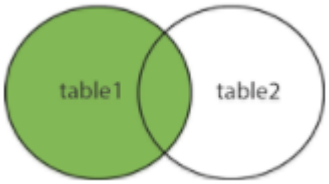
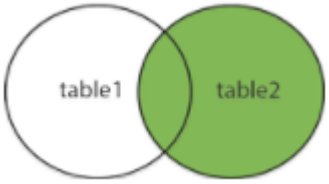
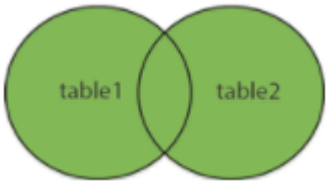
Η βάση δεδομένων mySQL περιέχει έναν ή περισσότερους πίνακες, καθένας από τους οποίους περιέχει δεδομένα ή γραμμές (rows). Οι γραμμές αυτές περιλαμβάνουν πλήθος στηλών (columns) ή πεδίων (fields) που περιέχουν την καθεαυτή πληροφορία. Ο παρακάτω πίνακας απεικονίζει τα περιεχόμενα ενός πίνακα με 5 δημοσιεύσεις (publications) με τα πεδία συγγραφέας (author) ,τίτλος (title) , τύπος (type) και χρονολογία δημοσίευσης(year of publication).

Author	Title	Type	Year
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Non-Fiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

Βασικές εντολές MySQL

Περιγραφή	Εντολή
Δημιουργία πίνακα	CREATE TABLE [table name] (firstname VARCHAR(20), lastname VARCHAR(35),officeid VARCHAR(10),userid VARCHAR(15),username VARCHAR(8),email VARCHAR(35),phone VARCHAR(25),datestamp DATE,timestamp time);
Εισαγωγή στήλης σε πίνακα	alter table [table name] add column [new column name] varchar (20);
Πως κάνουμε μια στήλη μεγαλύτερη	alter table [table name] modify [column name] VARCHAR(30);
Αλλαγή ονόματος στήλης	alter table [table name] change [old column name] [new column name] varchar (50);
Διαγραφή στήλης	alter table [table name] drop column [column name];
Διαγραφή γραμμής/ων από ένα πίνακα.	DELETE from [table name] where [field name] = 'whatever';
Εμφάνιση των databases στον sql server.	show databases;
Επιλογή συγκεκριμένης database.	use [db name];
Εμφάνιση όλων των πινάκων στην database.	show tables;
Εμφάνιση του format ενός πίνακα.	describe [table name];
Διαγραφή database.	drop database [database name];
Διαγραφή πίνακα.	drop table [table name];
Εμφάνιση όλων των στοιχείων ενός πίνακα	SELECT * FROM [table name];
Εμφάνιση στηλών και του περιεχομένου τους από ενα συγκεκριμένο πίνακα .	show columns from [table name];
Απαρίθμηση σειρών .	SELECT COUNT(*) FROM [table name];

Πως κάνουμε μια στήλη “μοναδική” έτσι ώστε να μην υπάρχουν διπλότυπα	alter table [table name] add unique ([column name]);
Εμφάνιση των εγγραφών με όνομα "Bob" και τηλέφωνο '1234567'.	SELECT * FROM [table name] WHERE name = "Bob" AND phone_number = '1234567';
Εμφάνιση των εγγραφών με που **δεν** έχουν όνομα "Bob" και τηλέφωνο '1234567' οργανωμένα με βάση το τηλέφωνο.	SELECT * FROM [table name] WHERE name != "Bob" AND phone_number = '1234567' order by phone_number;
Εμφάνιση των εγγραφών όπου το όνομα ξεκινάει με τα γράμματα "Mar" .	SELECT * FROM [table name] WHERE name like "Mar%" ';
Εμφάνιση μοναδικών εγγραφών.	SELECT DISTINCT [column name] FROM [table name];
Εμφάνιση επιλεγμένων εγγραφών σε αύξουσα ή φθίνουσα σειρά	SELECT [col1],[col2] FROM [table name] ORDER BY [col2] DESC;
Υπολογισμός του συνολικού αριθμολυ δεδομένων ενός πίνακα:	SELECT SUM([column]) FROM [table];
Επιλογή μέγιστης τιμής μια στήλης ενός πίνακα	SELECT MAX([column]) FROM [table];
Επιλογή ελάχιστης τιμής μια στήλης ενός πίνακα	SELECT MIN([column]) FROM [table];
Υπολογισμός μέσου όρου μια στήλης ενός πίνακα	SELECT AVG([column]) FROM [table];
Ενημέρωση πληροφοριών ενός ήδη υπάρχων πίνακα.	UPDATE [table name] SET Select_priv = 'Y',Insert_priv = 'Y',Update_priv = 'Y' where [field name] = 'user';
Ενημέρωση δικαιωμάτων βάσης δεδομένων.	FLUSH PRIVILEGES;
	SELECT *column_name(s)* FROM *table1* UNION SELECT *column_name(s)* FROM *table2*;

Inner Join 	SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate FROM Orders INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
Left Join 	SELECT *column_name(s)* FROM *table1* LEFT JOIN *table2* ON *table1.column_name* = *table2.column_name*;
Right Join 	SELECT *column_name(s)* FROM *table1* RIGHT JOIN *table2* ON *table1.column_name* = *table2.column_name*;
Full outer Join 	SELECT *column_name(s)* FROM *table1* FULL OUTER JOIN *table2* ON *table1.column_name = table2.column_name*;

[29]

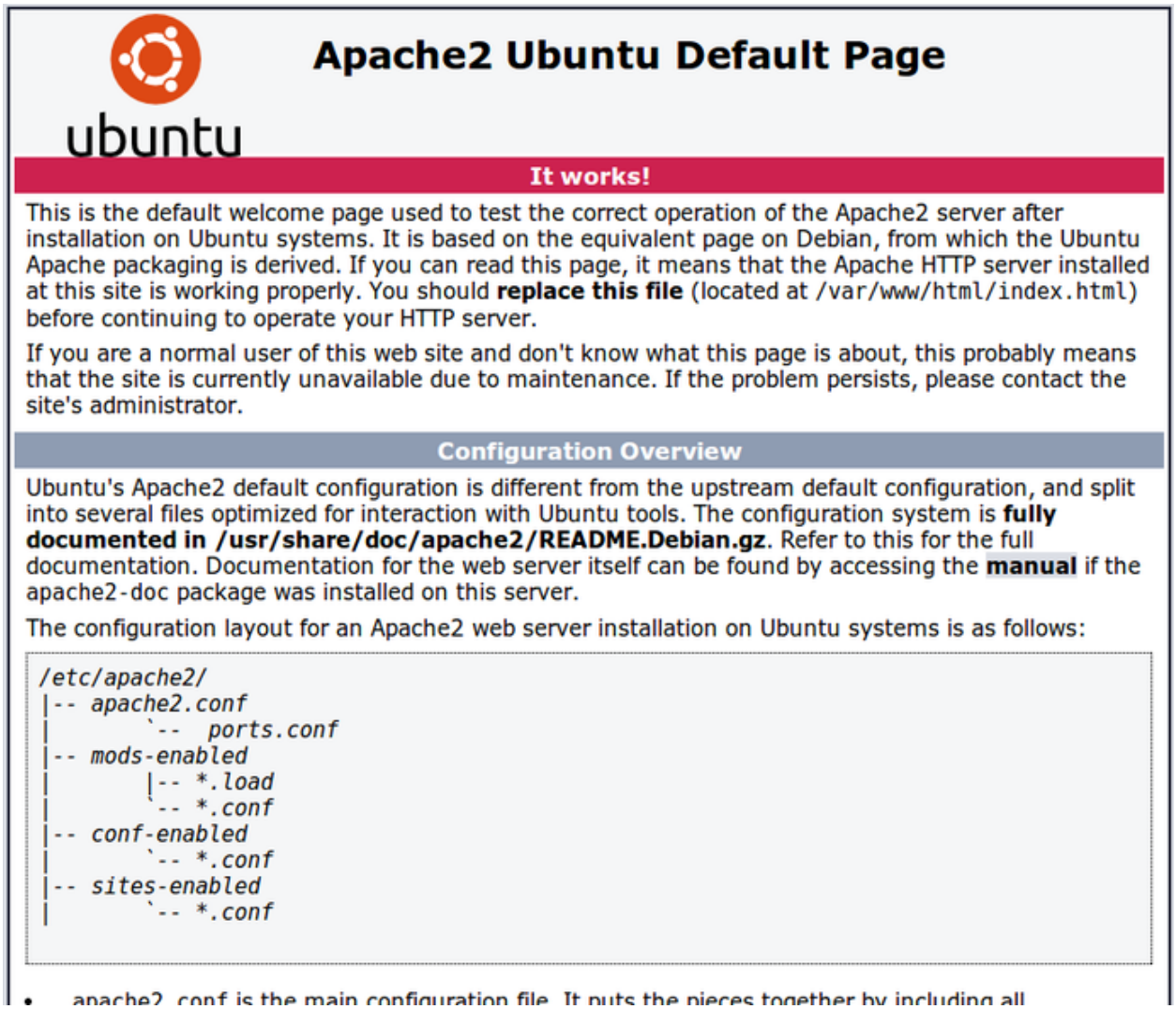
Web Server - Εγκατάσταση MySQL

Για να εγκαταστήσουμε την MySQL χρειαζόμαστε έναν web server. Θα χρησιμοποιήσουμε τον Apache web server σε linux μηχάνημα. Παρακάτω περιγράφεται η διαδικασία εγκατάστασης του Apache σε μηχάνημα Ubuntu 14.04.

Η εγκατάσταση του apache γίνεται με την παρακάτω εντολή.

sudo apt-get install apache2

Πριν από αυτή, είναι προτεινόμενη η εκτέλεση της εντολής sudo apt-get update για ενημέρωση του συστήματος. Έτσι στο μηχάνημά μας τώρα τρέχει ο apache web server και αν επισκεφτούμε τη διεύθυνση του μηχανήματός στο διαδίκτυο, θα δούμε την default σελίδα του apache2.



Εικ. 5

Τώρα μπορούμε να συνεχίσουμε με την εγκατάσταση της MySQL. Η MySQL είναι ένα σύστημα διαχείρησης βάσεων (DBMS). Για την εγκατάστασή του θα χρειαστούμε βοηθητικά πακέτα.

sudo apt-get install mysql-server php5-mysql

Κατά την εγκατάσταση θα μας ζητηθεί να ορίσουμε password για τον root user. Θα είναι ο admin χρήστης της MySQL. Στη συνέχεια πρέπει να πούμε στην MySQL να ορίσει τη δομή των βάσεων στις οποίες θα αποθηκεύουμε δεδομένα. Αυτό γίνεται με την εντολή:
sudo mysql_install_db

Σημαντικό βήμα είναι η εκτέλεση ενός διαδραστικού script το οποίο θα ασφαλήσει το DBMS μας. Εκτελούμε την εντολή:
sudo mysql_secure_installation

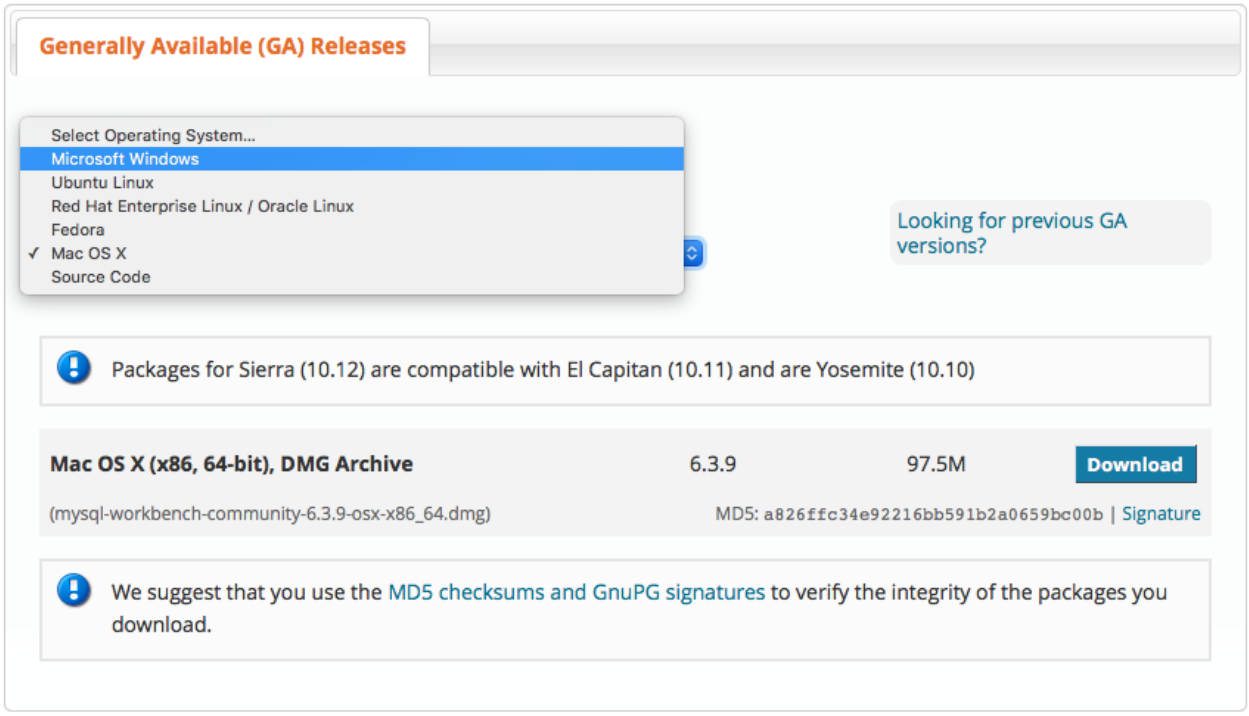
Θα μας ζητηθεί το password του root user και θα μας δωθεί η επιλογή να το αλλάξουμε. Στις επόμενες ερωτήσεις δεχόμαστε τις προεπιλεγμένες απαντήσεις και το σύστημά μας είναι έτοιμο.

Για να έχουμε τη δυνατότητα να τρέχουμε τις εφαρμογές μας στον web server που δημιουργήσαμε, θα χρειαστούμε την PHP. Η PHP είναι το στοιχείο εκείνο που θα επεξεργάζεται τον κώδικά μας, θα παράγει δυναμικό περιεχόμενο, θα συνδέεται με τη βάση και θα ανακτά δεδομένα για προβολή, με λίγα λόγια είναι το κύριο συστατικό για τη λειτουργία της web εφαρμογής.
Για την εγατάσταση της php και των βοηθητικών πακέτων εκτελούμε την εντολή:
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt

Εκτελούμε επανεκκίνηση του server για να λάβουν χώρα οι αλλαγές που πραγματοποιήσαμε:
sude service apache2 restart.
Από αυτή τη στιγμή ο web server είναι έτοιμος να τρέξει τη web εφαρμογή μας.

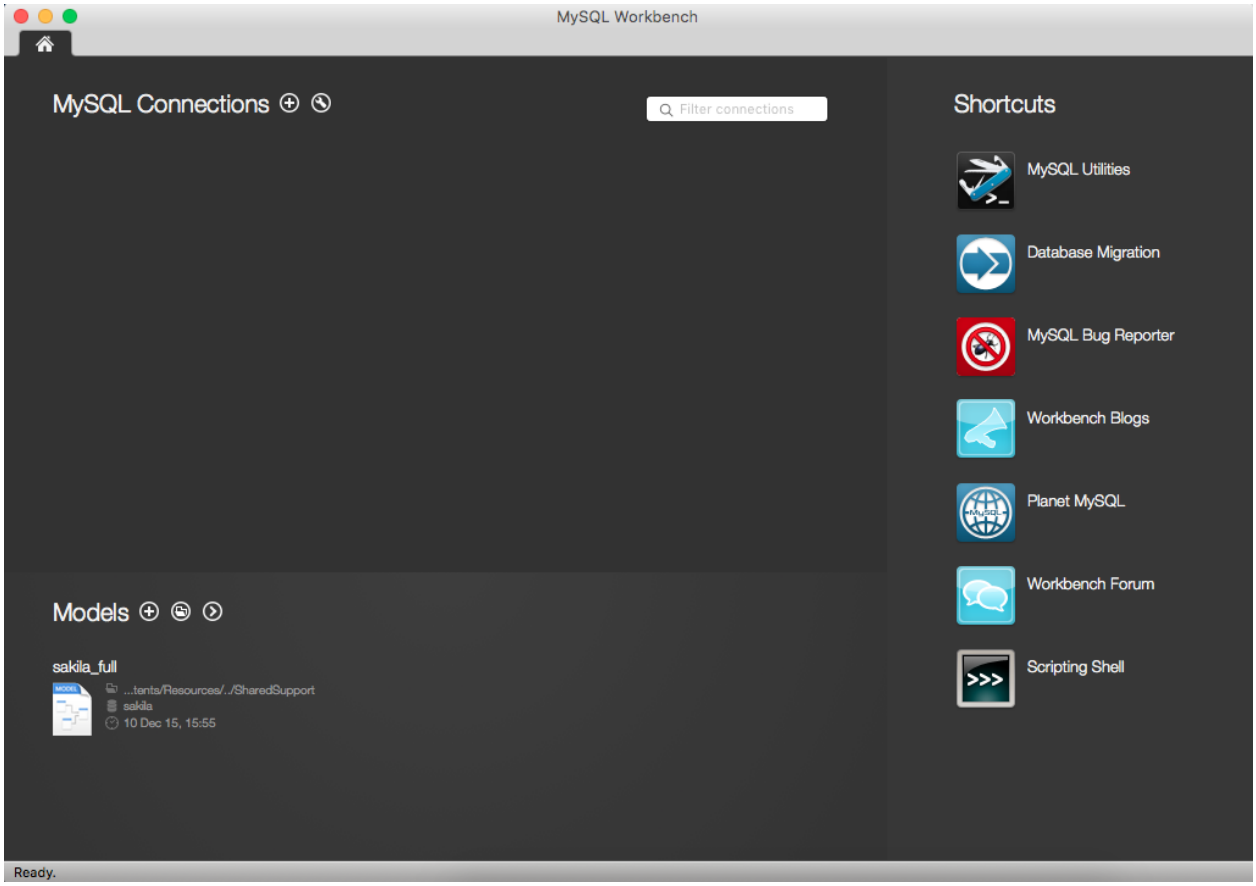
Εργαλεία διαχείρισης MySQL βάσης
Για τη διαχείριση της MySQL βάσης μας, υπάρχουν εργαλεία που μας διευκολύνουν, παρέχοντας γραφικό περιβάλλον για εκτέλεση πληθώρας λειτουργιών. Τα δύο δημοφιλέστερα εργαλεία είναι το MySQL Workbench και το phpMyAdmin.

MySQL Workbench: Είναι δωρεάν software διαθέσιμο για Windows, Linux & OS X. Ενδεικτικά γιια την εγκατάσταση της τελευταίας έκδοσης σε windows μηχάνημα, απαιτείται Microsoft .Net 4.5, Visual C++ 2015 και windows 7 η νεότερο λειτουργικό. Για τη λήψη του MySQL Workbench επισκεφτείτε το http://dev.mysql.com/downloads/workbench/
και επιλέξτε την πλατφόρμα εγκατάστασης.

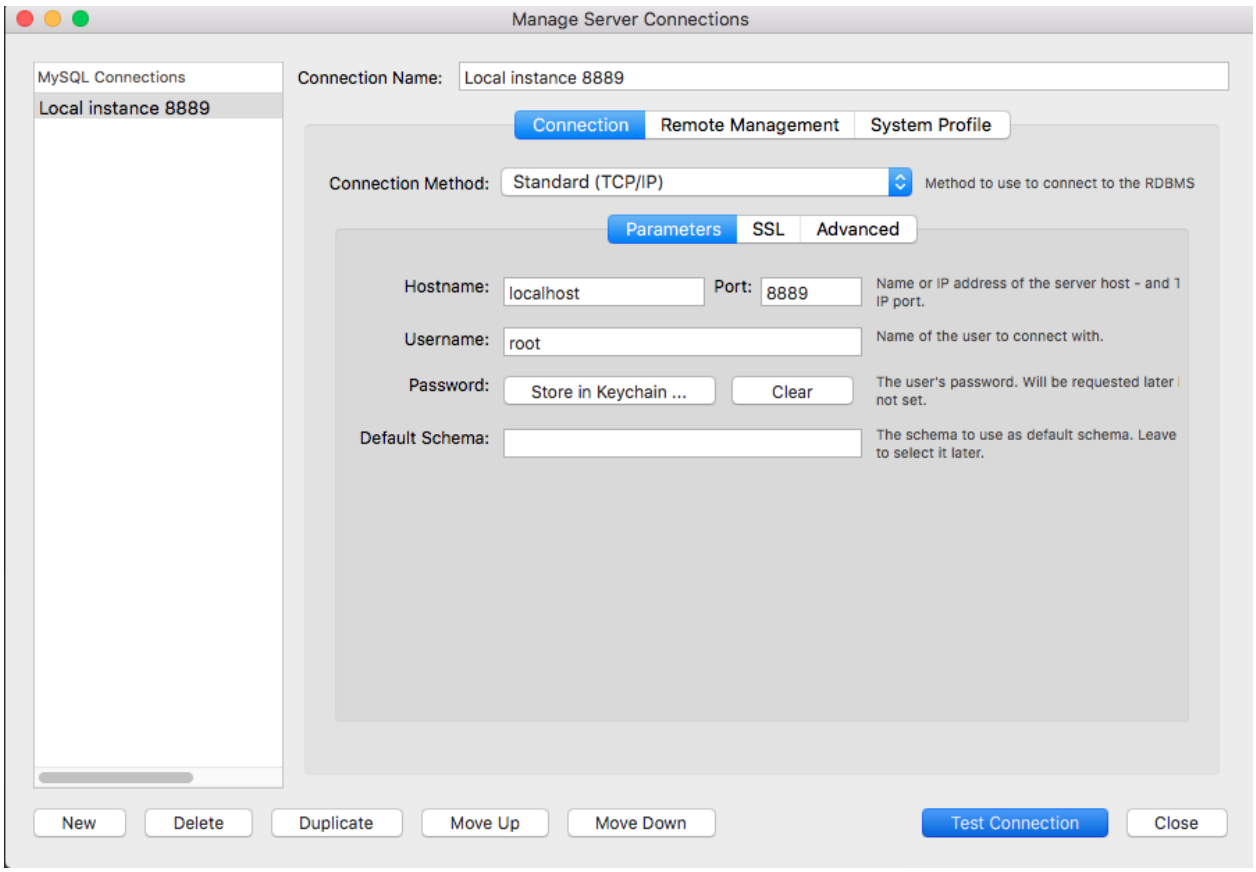


[32]

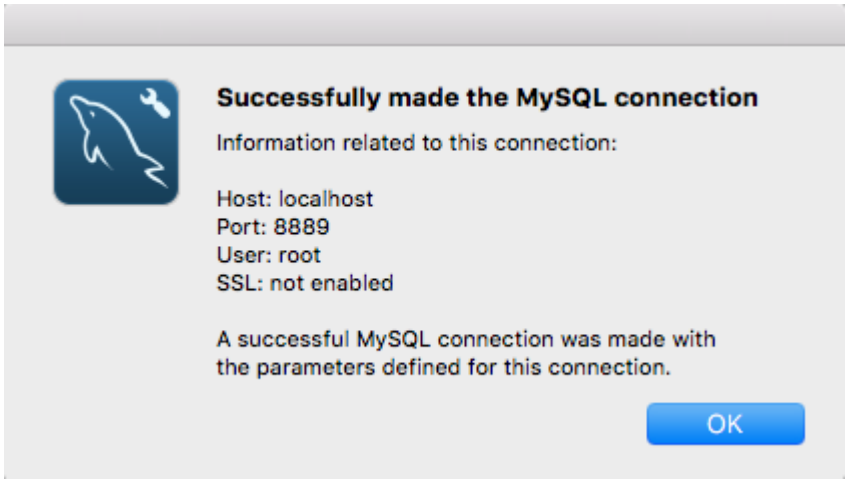
Ανοίγοντας στο MySQL Workbench συναντάμε την αρχική οθόνη από την οποία μπορούμε να δημιουργήσουμε μια νέα σύνδεση.



Πατώντας το + στο MySQL Connections ανοίγει το παράθυρο διαλόγου για νέα σύνδεση όπου εισάγουμε τα στοιχεία.



hostname: Η διεύθυνση του web server στο οποίο είναι εγκατεστημένη η MySQL
Port: Η πόρτα στην οποία ακούει η MySQL. Η default είναι η 3306
Username: Το username του χρήστη που επιλέξαμε κατά την εγκατάσταση της MySQL
Επιλέγουμε Test Connection και εμφανίζεται το ακόλουθο παράθυρο.



Η σύνδεση ήταν επιτυχείς και μπορούμε πλέον διαχειριστούμε τις βάσεις μέσα από γραφικό περιβάλλον η να τρέξουμε τα ερωτήματά μας κατευθείαν στο MySQL Workbench.
Το επόμενο εργαλείο για τη διαχείριση MySQL βάσεων είναι το phpMyAdmin. Είναι δωρεάν λογισμικό γραμμένο σε PHP και είναι σχεδιασμένο για διαχείριση βάσεων στο web μέσω web browser. Είναι δηλαδή μια εφαρμογή που παρέχεται στο χρήστη μέσω web browser. Όπως και στο MySQL Workbench, έτσι και εδώ έχουμε τη δυνατότητα να επεξεργαστούμε τις βάσεις με γραφικό περιβάλλον ή να τρέξουμε SQL ερωτήματα. Για να εγκαταστήσουμε την τελευταία έκδοση του phpMyAdmin απαιτείται να έχουμε έναν web server (Apache, nginx κ.α), εγκατστημένη PHP 5.5 η νεότερη, MySQL 5.5 ή νεότερη και έναν browser που να υποστηρίζει jQuery 2.0 για να το χρησιμοποιούμε στο client μηχάνημά μας. Για την εγκατάσταση του phpMyAdmin σε Apache web server σε ubuntu Linux μηχάνημα, τρέχουμε την εντολή:

sudo apt-get install phpmyadmin apache2-utils

Κατά την εγκατάσταση θα εμφανιστούν παράθυρα διαλόγου στα οποία θα χρειαστεί να επιλέξουμε password για τη σύνδεσή μας στο phpMyAdmin.

Μετά την ολοκλήρωση της εκγατάστασης, με έναν text editor ανοίγουμε το αρχείο /etc/apache2/apache2.conf και γράφουμε την παρκάτω εντολή:

Include /etc/phpmyadmin/apache.conf

Τέλος επανεκκινούμε τον apache

sudo service apache2 restart

Για να ανοίξουμε το phpMyAdmin στον web browser πηγαίνουμε στη διεύθυνση

server_ip/phpmyadmin

Μετά τη σύνδεση είμαστε έτοιμοι να επεξργαστούμε τις βάσεις και να τρέξουμε sql ερωτήματα μέα στο γραφικό περιβάλλον του phpMyAdmin.

[33][34][35]

MySQL με PHP

Παρακάτω θα δούμε πως πραγματοποιούνται βασικά tasks όπως είναι η σύνδεση σε MySQL βάση, καθώς και η εκτέλεση βασικών ερωτημάτων, με PHP κώδικα.

Σύνδεση σε MySQL με PHP

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Παράμετροι:
$servername = “localhost”;
$username //Το username του χρήστη που δημιουργήσαμε κατά την εγκατάσταση της MySQL
$password //Το password του χρήστη που δημιουργήσαμε κατά την εγκατάσταση της MySQL

Δημιουργία Βάσης
Για τη δημιουργία βάσης εκτελούμε το ερώτημα “CREATE DATABASE db_name”

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

Αφού δημιουργήθηκε η βάση, στο επόμενο βήμα θα δημιουργήσουμε ένα πίνακα με το ερώτημα

CREATE TABLE table_name (
col_name1 col_type AUTO INCREMENT PRIMARY KEY,
col_name2 col_type,
col_name3 col_type,
col_name4 col_type);

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Προσθήκη εγγραφής στον πίνακα

Το ερώτημα για την εισαγωγή στη βάση είναι το ακόλουθο:

INSERT INTO table_name (field1, field2, field3) VALUES (‘value1’, ‘value2’, ‘value3’);

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Προσθήκη πολλαπλών εγγραφών

Με την php μπορούμε να προσθέσουμε περισσότερες από μία εγγραφές, τρέχοντας το ερώτημα μια φορά

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Ανάγνωση εγγραφής

Το ερώτημα για την ανάγνωση εγγραφών είναι το

SELECT field_name FROM table_name;

Μπορούμε να επιλέξουμε ένα πεδίο, περισσότερα, ή όλα με το SELECT * FROM table

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

Συνθήκες

Για την ανάγνωση, ενημέρωση η διαγραφή δεδομένων υπό συνθήκη, χρησιμοποιούμε τον όρο WHERE

SELECT * FROM table_name WHERE field = ‘value’;

Οι ακόλουθοι operators μπορούν να χρησιμοποιηθούν στον WHERE όρο

=	Equal
!=	Not Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

BETWEEN	Ανάμεσα σε ένα εύρος
LIKE	Αναζήτηση μοτίβου
IN	Ορισμός πολλαπλών πιθανών τιμών για μια στήλη

Διαγραφή

Για τη διαγραφή μιας εγγραφής θα χρησιμοποιήσουμε τον όρο WHERE

DELETE FROM table_name WHERE field = ‘value’;

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

Ενημέρωση εγγραφής

Με παρόμοιο τρόπο όπως και στη διαγραφή θα χρησιμοποιήσουμε συνθήκη

UPDATE table_name SET field = ‘value’ WHERE field = ‘value’;

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

Ξένο κλειδι

Τα ξένα κλειδιά στις βάσεις δεδομένων μας επιτρέπουν να διατηρούμε μια σχέση - σύνδεση μεταξύ των πινάκων. Μια στήλη σε ένα πίνακα, αναφέρεται σε μια στήλη σε άλλο πίνακα, με σκοπού να διατηρείται η ακεραιότητα των δεδομένων. Η στήλη αναφοράς πρέπει να είναι πρωτεύον κλειδί στον πίνακα αναφοράς. Με τα ξένα κλειδιά μπορούμε να ορίσουμε κανόνες ανάμεσα στους δύο πίνακες.

Έστω οι πίνακες της βάσης μας:

Πίνακας CUSTOMERS	Πίνακας ORDERS
C_ID	ORDER_ID
SURNAME	CUSTOMER_ID

Στον πίνακα ORDERS, το πεδίο CUSTOMER_ID είναι ξένο κλειδί και αναφέρεται στο C_ID του πίνακα CUSTOMERS. Έτσι δεν μπορούμε να έχουμε order χωρίς customer_id. Μπορούμε να

ορίσουμε κανόνες για το τι θα συμβεί εάν αλάξουμε το C_ID ενός πελάτη. Για παράδειγμα έχουμε τη δυνατότητα να πούμε στη βάση ότι δεν επιτρέπεται αλλαγή στο CUSTOMER_ID. Τότε αν πάμε να αλλάξουμε το ID ενός πελάτη, η MySQL θα μας προειδοποιήσει για παραβίαση περιορισμού ξένου κλειδιού. Από την άλλη μπορούμε να εκμεταλλευθούμε τον κανόνα, αν αλλάξει το C_ID να αλλάξει και το CUSTOMER_ID. Αυτό θα γίνει αυτόματα και θα μας γλιτώσει από περισσότερα ερωτήματα. Το ίδιο συμβαίνει και αν θελήσουμε να διαγράψουμε έναν πελάτη. Ορίζουμε αν θα διαγραφούν και οι παραγγελίες του ή όχι (να γίνει NULL το CUSTOMER_ID), ή αν δε θα επιτρέπεται τέτοια διαγραφή.

Το storage engine της MySQL που υποστηρίζει ξένα κλειδιά είναι το InnoDB

Δημιουργία ξένου κλειδιού

```
CREATE TABLE ORDERS(
ORDER_ID int not null auto_increment primary key,
CUSTOMER_ID int not null,
FOREIGN KEY cid_fk(CUSTOMER_ID)
REFERENCES CUSTOMERS(C_ID)
ON UPDATE CASCADE
ON DELETE RESTRICT
)
```

Αν ο πίνακας ORDERS έχει ήδη δημιουργηθεί, προσθέτουμε ξένο κλειδί με τον ακόλουθο τρόπο.

```
ALTER TABLE ORDERS
ADD FOREIGN KEY cid_fk(CUSTOMER_ID)
REFERENCES CUSTOMERS(C_ID)
ON UPDATE CASCADE
ON DELETE RESTRICT
```

Κεφάλαιο 5. Ανάπτυξη εφαρμογής καταγραφής και διαχείρισης εξοπλισμού

Στο 5ο κεφάλαιο περιγράφεται αναλυτικά ο τρόπος ανάπτυξης του συστήματος καθώς και όλα τα εργαλεία που χρησιμοποιήθηκαν.

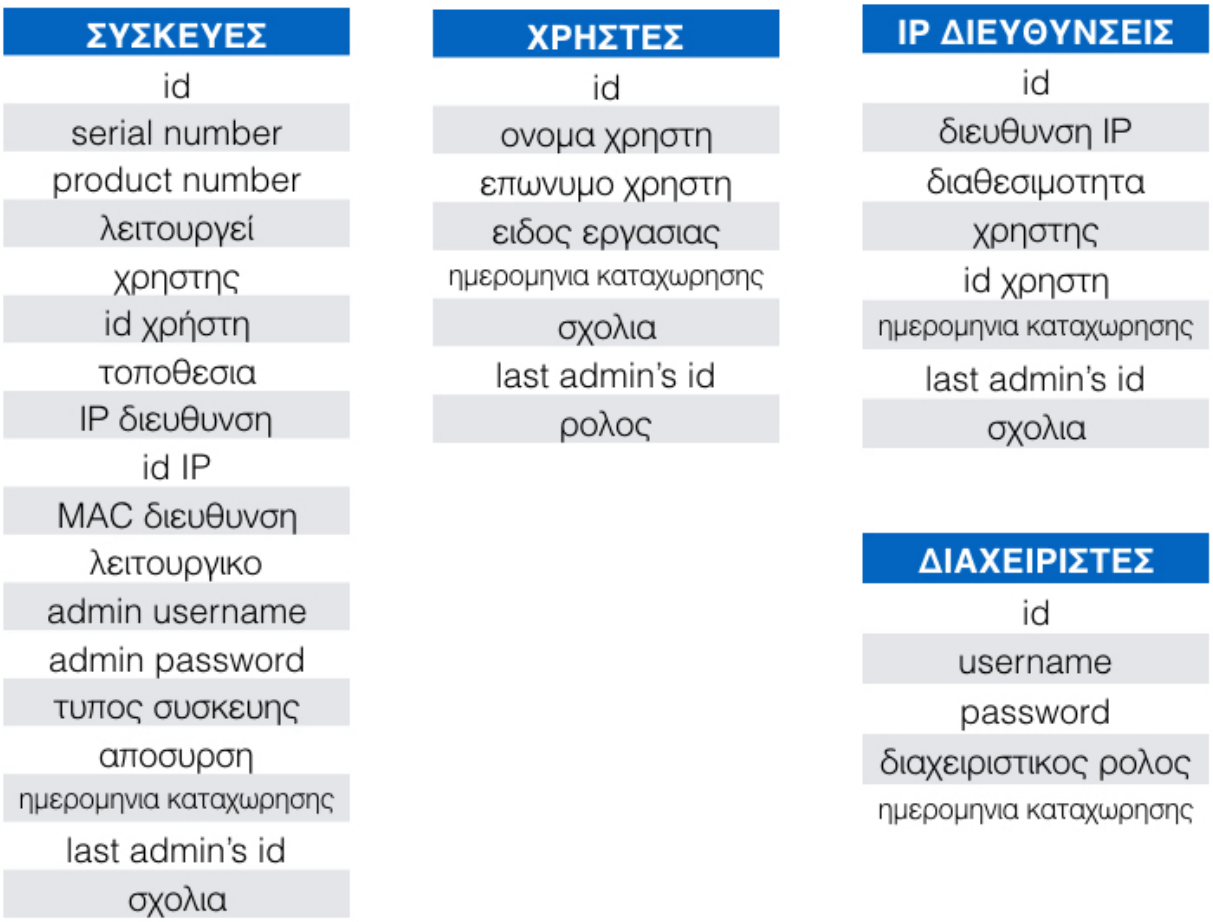
5.1 Περιγραφή βάσης δεδομένων

Το σύστημα καταγραφής του εξοπλισμού του πανεπιστημίου χωρίστηκε σε 3 βασικά μέρη. Στις συσκευές (H/Y, printers, routers), τους χρήστες (προσωπικό πανεπιστημίου) και τις IP διευθύνσεις.

- Συσκευές: Η καταγραφή των συσκευών είναι το θέμα και βασικός λόγος που οδήγησε στην ανάπτυξη του συτήματος.
- Χρήστες: Για την καλύτερη σχεδίαση και διαχείριση των συσκευών κρίθηκε απαραίτητο ο διαχωρισμός των χρηστών, δηλαδή του προσωπικού του Πανεπιστημίου σε ξεχωριστή οντότητα. Με αυτόν τον τρόπο είναι δυνατή η ανάπτυξη λειτουργιών ανέθεσης συσκευών σε χρήστες και η εύκολη διαχείρισή τους.
- Διευθύνσεις IP: Καθώς σχεδόν το σύνολο των συσκευών που μας ενδιαφέρουν, λειτουργούν συνδεδεμένες στο δίκτυο του πανεπιστημίου και χρειάζεται μια μοναδική διεύθυνση IP η κάθε μία, ήταν αναγκαία η διαχείρισή τους ως ξεωριστή οντότητα.

Επομένως το σύστημα βασίζεται σε μια MySQL βάση δεδομένων με 3 οντότητες. Επειδή σχεδιάστηκε ώστε να χρησιμοποιηθεί από το κέντρο δικτύων και κυρίως από την ομάδα τεχνικής υποστήριξης, λογικό είναι να υποστηρίζει περισσότερους από έναν ρόλους χρηστών, για να έχουν διαφορετικά δικαιώματα. Αυτή η λειτουργία κρίθηκε αναγκαία για τη σωστή οργάνωση του συστήματος. Έτσι έχομε μια τέταρτη οντότητα, τους διαχειριστές.

Στην παρακάτω εικόνα φαίνεται η δομή της βάσης δεδομένων



Οι συσκευές όπου είναι και η κύρια οντότητα συνδέεται με ξένα κλειδιά με τους χρήστες, τις IP καθώς και τους διαχειριστές. Ο πίνακας των συσκευών, διαθέτει ένα πεδίο στο οποίο καταχωρείται το id του χρήστη στον οποίο ανατέθηκε η συσκευή. Επίσης διαθέτει ένα περίο στο οποίο καταχωρείται το id της IP της συσκευής αν της αναθέσουμε IP διεύθυνση. Τέλος διαθέτει ένα πεδίο στο οποίο καταχωρείται το id του διαχειριστή που τη δημιούργησε ή την επεξεργάστηκε τελευταίος.

Ο πίνακας των χρηστών έχει ως πρωτεύον κλειδί το id του χρήστη το οποίο είναι ξένο κλειδί στις συσκευές.

Ο πίνακας των IP έχει ως πρωτεύον κλειδί το id της IP το οποίο είναι ξένο κλειδί στις συσκευές. Επιπλέον έχει ένα πεδίο id χρήστη στο οποίο καταχωρείται το id του χρήστη στον οποίο ανατέθηκε η συσκευή.

Ο πίνακας των διαχειριστών έχει ως πρωτεύον κλειδεί το id του διαχειριστή το οποίο είναι

ξένο κλειδί στον πίνακα των συσκευών, των χρηστών και των IP

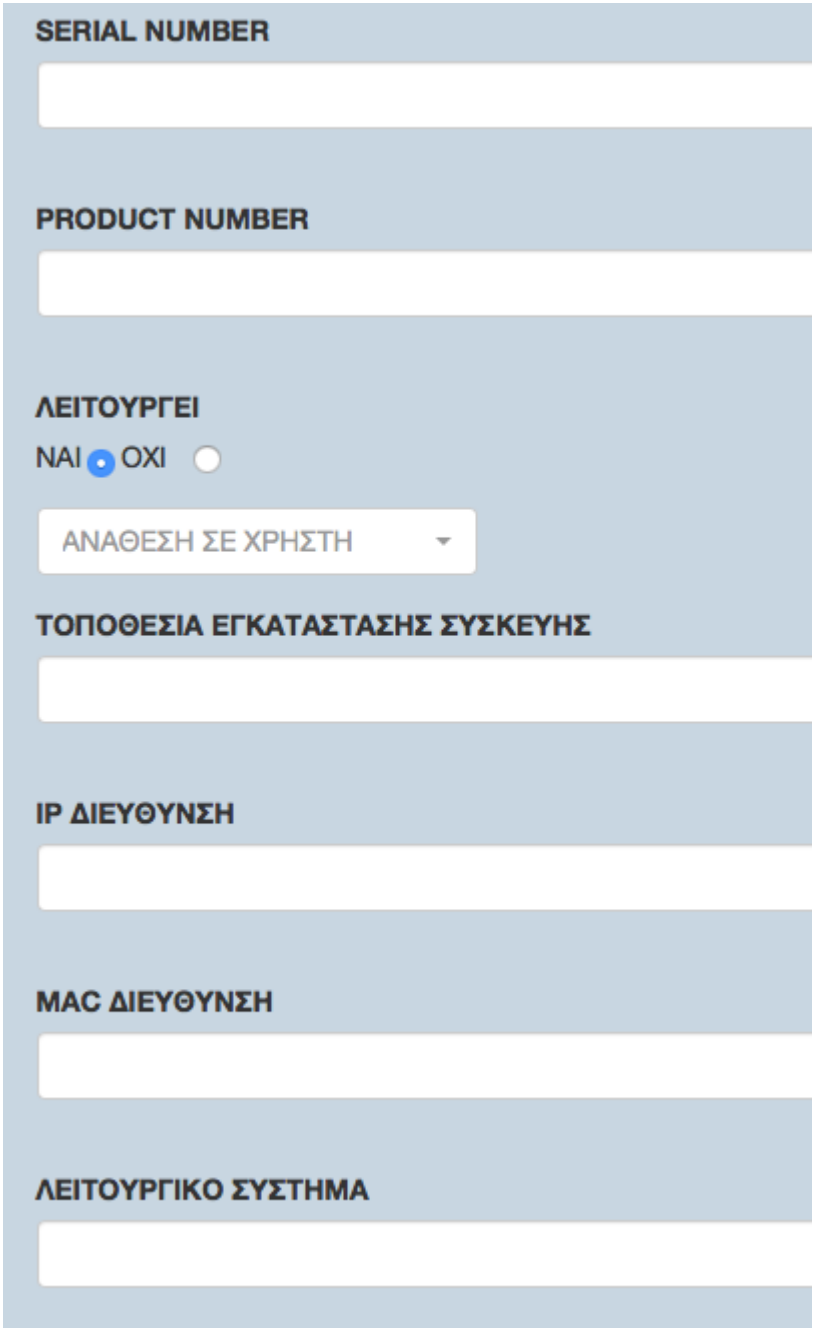
5.2 Λειτουργίες

Οι βασικές λειτουργίες που υποστηρίζει το σύστημα είναι οι εξής:

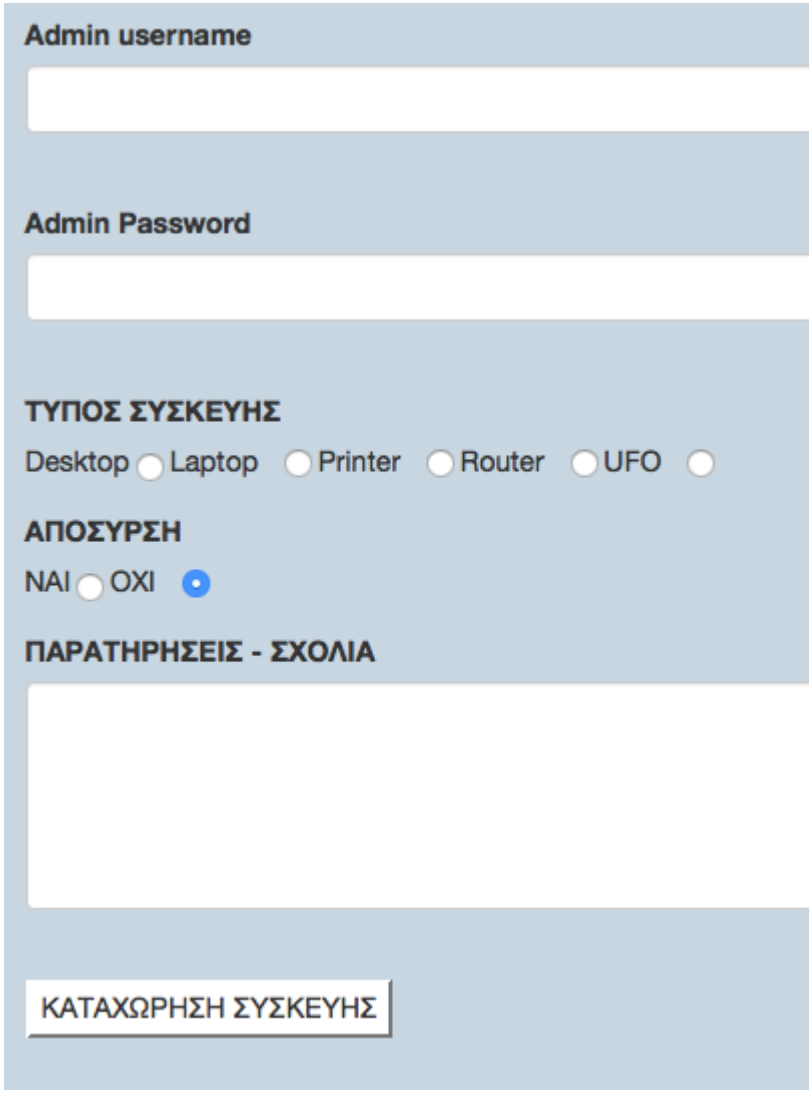
- προσθήκη, επεξεργασία, αναζήτηση συσκευών
- προσθήκη, επεξεργασία, αναζήτηση χρηστών
- προσθήκη, επεξεργασία, αναζήτηση διευθύνσεων
- προσθήκη, επεξεργασία, αναζήτηση διαχειριστών

Συσκευές: κατά την προσθήκη μιας συσκευής στο σύστημα καταγράφουμε τα παρακάτω στοιχεία

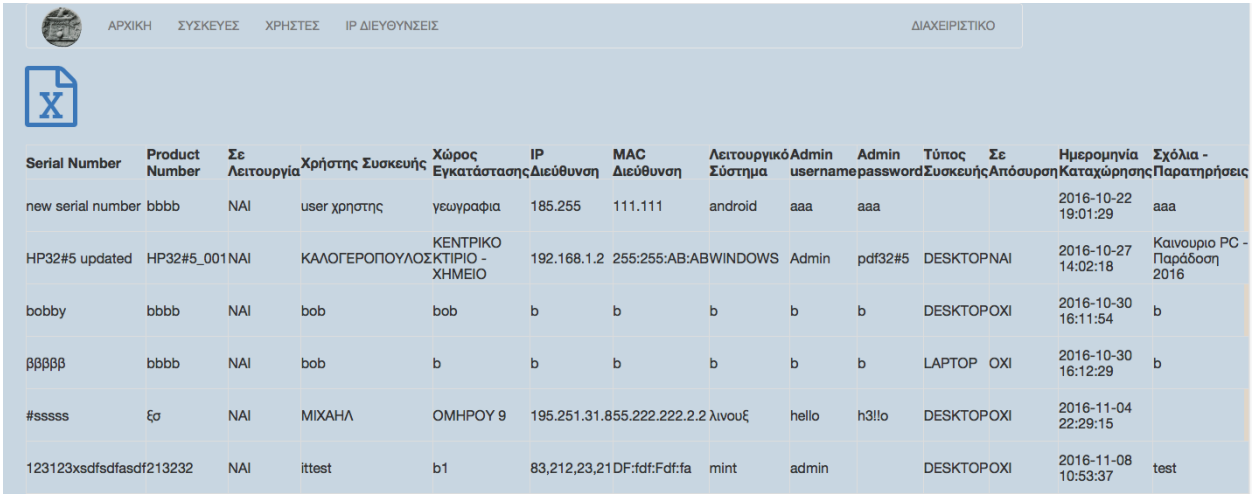
σειριακός αριθμός
αριθμός προϊόντος
αν λειτουργεί η συσκευή
επιλέγουμε τον χρήστη στον οποίο θα την αναθέσουμε
μέρος εντός του πανεπιστημίου που θα εγκατασταθεί
Διεύθυνση IP
Διεύθυνση MAC
Λειτουργικό σύστημα
username διαχειριστή
password διαχειριστή
τύπος συσκευής, (laptop, desktop, router….)
αν η συσκευή θα βγει σε απόσυρση
παρατηρήσεις - σχόλια


SERIAL NUMBER
PRODUCT NUMBER
ΛΕΙΤΟΥΡΓΕΙ
ΝΑΙ
ΟΧΙ
ΑΝΑΘΕΣΗ ΣΕ ΧΡΗΣΤΗ
ΤΟΠΟΘΕΣΙΑ ΕΓΚΑΤΑΣΤΑΣΗΣ ΣΥΣΚΕΥΗΣ
IP ΔΙΕΥΘΥΝΣΗ
MAC ΔΙΕΥΘΥΝΣΗ
ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ

Εικ.6



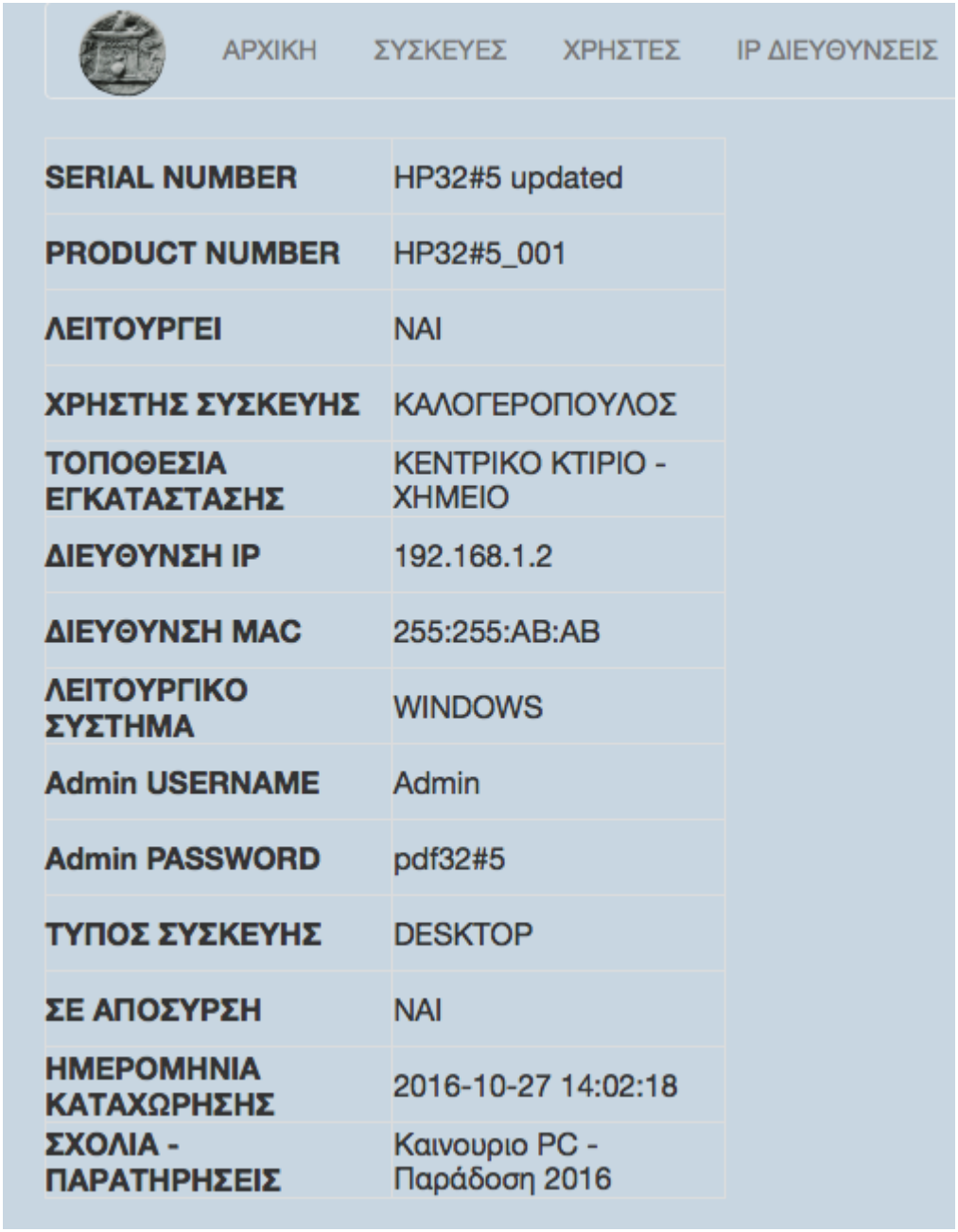
Στη συνέχεια το σύστημα μας δίνει τη δυνατότητα να προβάλουμε όλες τις καταχωρημένες συσκευές, σε μια λίστα καθώς και να εξάγουμε τη λίστα σε excel.



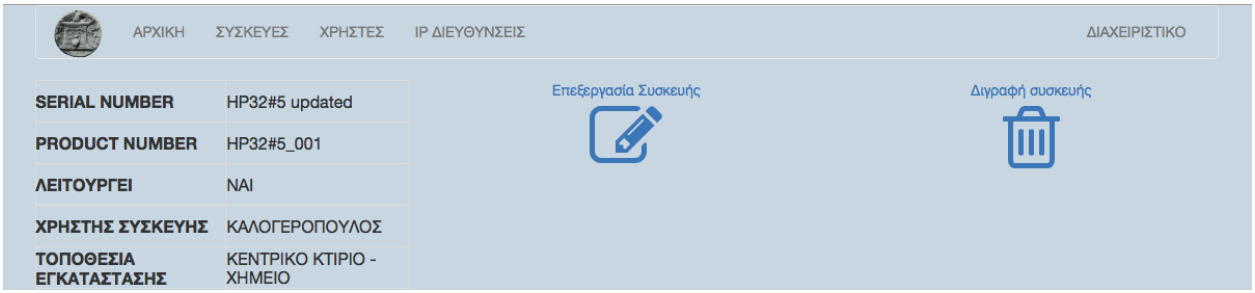
Serial Number	Product Number	Σε Λειτουργία	Χρήστης Συσκευής	Χώρος Εγκατάστασης	IP Διεύθυνση	MAC Διεύθυνση	Λειτουργικό Σύστημα	Admin username	Admin password	Τύπος Συσκευής	Σε Απόσυρση	Ημερομηνία Καταχώρησης	Σχόλια - Παρατηρήσεις
new serial number	bbbb	ΝΑΙ	user χρηστης	γεωγραφια	185.255	111.111	android	aaa	aaa			2016-10-22 19:01:29	aaa
HP32#5 updated	HP32#5_001	ΝΑΙ	ΚΑΛΟΓΕΡΟΠΟΥΛΟΣ	ΚΕΝΤΡΙΚΟ ΚΤΙΡΙΟ - ΧΗΜΕΙΟ	192.168.1.2	255:255:AB:AB	WINDOWS	Admin	pdf32#5	DESKTOP	ΝΑΙ	2016-10-27 14:02:18	Καινουριο PC - Παράδοση 2016
bobby	bbbb	ΝΑΙ	bob	bob	b	b	b	b	b	DESKTOP	ΟΧΙ	2016-10-30 16:11:54	b
βββββ	bbbb	ΝΑΙ	bob	b	b	b	b	b	b	LAPTOP	ΟΧΙ	2016-10-30 16:12:29	b
#sssss	ξσ	ΝΑΙ	ΜΙΧΑΗΛ	ΟΜΗΡΟΥ 9	195.251.31.85	5.222.222.2.2	λινουξ	hello	h3!lo	DESKTOP	ΟΧΙ	2016-11-04 22:29:15	
123123xsdfsdfasdf	213232	ΝΑΙ	ittest	b1	83,212,23,21	DF:fdf:Fdf:fa	mint	admin		DESKTOP	ΟΧΙ	2016-11-08 10:53:37	test

Εικ.7

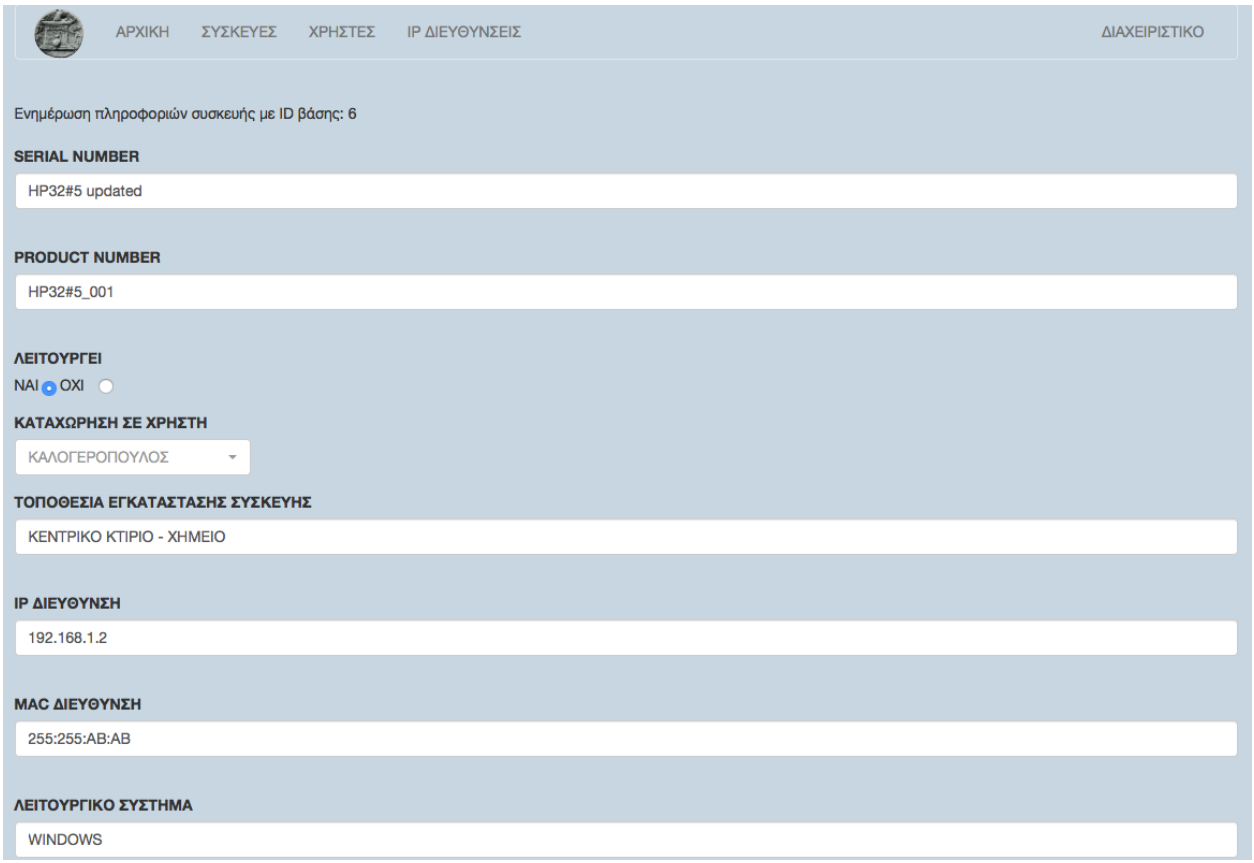
Κατά την προβολή της λίστας μπορούμε να επιλέξουμε μια συγκεκριμένη συσκευή και να προβάλουμε μόνο αυτή



Κατά την προβολή μιας συγκεκριμένης συσκευής έχουμε διαθέσιμες τις επιλογές για επεξεργασία των στοιχείων της ή διαγραφή της εντελώς



Επεξεργασία συσκευής

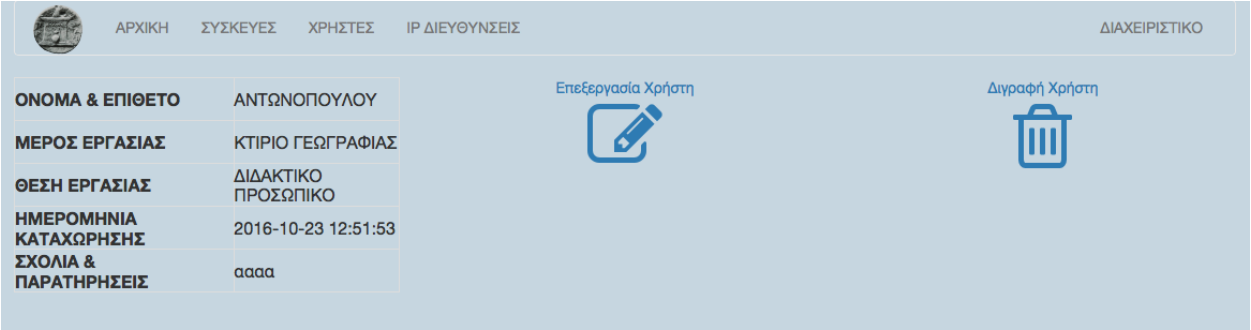


Το σύστημα μας δίνει τη δυνατότητα να αναζητήσουμε συσκευές. Η αναζήτηση θα ψάξει σε όλα τα πεδία του πίνακα συσκευών, εκτός από την ημερομηνία καταχώρησης.

Χρήστες: Κατά την προσθήκη ενός χρήστη στο σύστημα καταγράφουμε τα παρακάτω στοιχεία.

Ονοματεπώνυμο
μέρος εντός του πανεπιστημίου που εργάζεται (αριθμό γραφείου - κτίριο)
Θέση εργασίας (διδακτικό προσωπικό - διδακτορικός φοιτητής - ερευνητής)
παρατηρήσεις - σχόλια

Όπως και στις συσκευές, το σύστημα μας δίνει τη δυνατότητα να προβάλουμε όλους τους καταχωρημένους χρήστες σε λίστα και να την εξάγουμε σε excel. Μπορούμε να επιλέξουμε ένα χρήστη και να προβάλουμε τα στοιχεία του.



Εκεί θα έχουμε διαθέσιμες τις επιλογές για επεξεργασία των στοιχείων του και διαγραφή του.

Το σύστημα μας δίνει τη δυνατότητα να αναζητήσουμε χρήστες. Η αναζήτηση θα ψάξει σε όλα τα πεδία του πίνακα συσκευών, εκτός από την ημερομηνία καταχώρησης.

IP διευθύνσεις: κατά την προσθήκη μιας IP διεύθυνσης καταγράφουμε τα παρακάτω στοιχεία.

Διεύθυνση
αν είναι διαθέσιμη, δηλαδή ελεύθερη η IP
τον χρήστη στον οποίο την αναθέτουμε
παρατηρήσεις

Οι διευθύνσεις διαθέτουν παρόμοιες λειτουργίες με τις προηγούμενες οντότητες. Προβολή όλων των διευθύνσεων σε λίστα, επιλογή μιας από τη λίστα και προβολή των στοιχείων της, με διαθέσιμες τις επιλογές για επεξεργασία και διαγραφή.

Το σύστημα μας δίνει τη δυνατότητα να αναζητήσουμε διευθύνσεις. Η αναζήτηση θα ψάξει σε όλα τα πεδία του πίνακα συσκευών, εκτός από την ημερομηνία καταχώρησης.

5.3 Ανάπτυξη - Προγραμματισμός εφαρμογής

Η εφαρμογή αναπτύχθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού PHP. Για την υποστήριξη των λειτουργιών της εφαρμογής στο backend χρησιμοποιήθηκε το MVC framework Codeigniter. Για την εμφάνιση των δεδομένων χρησιμοποιήθηκε το framework Bootstrap, καθώς και Javascript, jquery και τεχνικές AJAX

5.3.1 PHP - Codeigniter

Η ανάπτυξη λογισμικού χρησιμοποιώντας frameworks είναι πολύ διαδεδομένη σήμερα. Προσφέρει πολλά πλεονεκτήματα με το κυριότερο να είναι η ταχύτητα ανάπτυξης του λογισμικού. Επίσης σημαντικό είναι το γεγονός ότι με frameworks αναπτύσσουμε λογισμικό επεκτάσιμο, πιο εύκολα συτηρήσιμο και με λιγότερα σφάλματα, καθώς ένα framework έχει ήδη χρησιμοποιηθεί αρκετά με αποτέλεσμα να έχουν διορθωθεί τυχόν λάθη που προέκυψαν. Συνοπτικά ένα framework είναι μια έννοια κατά την οποία λογισμικό που παρέχει γενικευμένη λειτουργικότητα, μπορεί να επεκταθεί από τον χρήστη με σκοπό να παρέχει εξειδικευμένη λειτουργικότητα για μια εφαρμογή. Ένα framework παρέχει ένα standard τρόπο κατασκευής εφαρμογών.

Τα frameworks διαφέρουν από τις βιβλιοθήκες σε τρία σημεία.

- Αντιστροφή ελέγχου: Αντίθετα με τις βιβλιοθήκες, στα frameworks ο έλεγχος του προγράμματος δεν ορίζεται από τον χρήστη, αλλά από το ίδιο το framework.
- Επεκτασιμότητα: Ο χρήστης μπορεί να επεκτείνει το framework ή να προσθέσει δικό του κώδικα για να παρέχει συγκεκριμένες λειτουργίες.
- Μη επεξργάσιμος κώδικας. Τα frameworks δεν σχεδιάστηκαν για να μεταβάλει ο χρήστης τον κώδικά τους. Δηλαδή, μπορεί να τα επεκτείνει αλλά όχι να τα μεταβάλει.

Το Codeigniter είναι ένα framework σχεδιασμένο για δημιουργία εφαρμογών σε PHP. Στόχος του είναι να παρέχει τη δυνατότητα ανάπτυξης web εφαρμογών ταχύτατα, προσφέροντας κατανοητή δομή, πολλές επιλογές βιβλιοθηκών για συνηθισμένα tasks και εύκολο interface. Παρέχεται δωρεάν με την άδεια του MIT και η χρήση του είναι εντελώς ελεύθερη.Τα πλεονεκτήματα του codeingiter σε σχέση με άλλα frameworks είναι ότι είναι εύκολο στην εξοικείωση για τον προγραμματιστή-χρήστη του, είναι πολύ ελαφρύ στη χρήση πόρων συστήματος και εξαιρετικά γρήγορο.

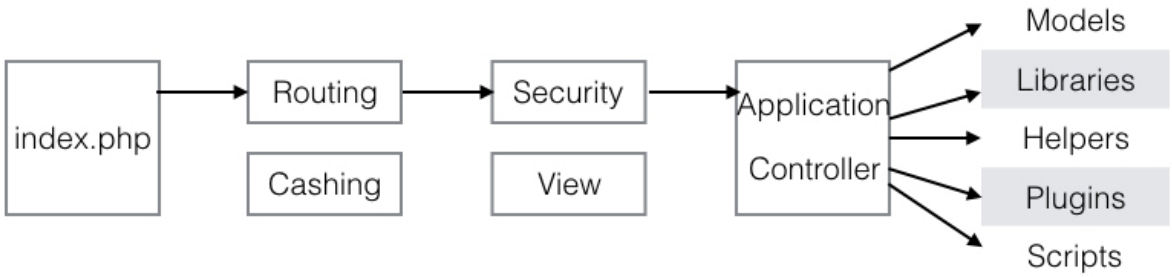
MVC Αρχιτεκτονική

Το Codeigniter χρησιμοποιεί την MVC αρχιτεκτονική. Το MVC design pattern είναι μια προσέγγιση λογισμικού στην οποία διαχωρίζεται το logic της εφαρμογής, από την προβολή των δεδομένων. Έτσι οι σελίδες που είναι υπεύθυνες για την προβολή, περιέχουν τον ελάχιστο δυνατό κώδικα php. Το MVC προκύπτει από το Model - View - Controller.

- Model: Το μοντέλο είναι υπεύθυνο για τα δεδομένα. Οι κλάσεις του μοντέλου θα περιέχουν τις μεθόδους εκείνες που αποκτούν, ενημερώνουν και εισάγουν δεδομένα στη βάση δεδομένων.
- View: Το view είναι η πληροφορία που εμφανίζεται στον χρήστη. Μπορεί να είναι μια ολόκληρη σελίδα ή τμήμα της όπως header, footer κ.α.
- Controller: Ο controller είναι ο διαμεσολαβητής ανάμεσα στο model και στο view.

Ροή δεδομένων

Στο codeigniter τα δεδομένα ακολουθούν την εξής πορεία:



Ο router εξετάζει το http αίτημα και αποφασίζει τι πρέπει να κάνει. Αν βρει cashed file, προωθεί αύτο το αρχείο στον browser παρακάμπτοντας τη συνηθισμένη διαδρομή. Πριν φορτωθεί ο controller, το http αίτημα μαζί με τα δεδομένα που τυχόν εστάλησαν από το χρήστη, φιλτράρονται για ασφάλεια. Στη συνέχεια ο controller φορτώνει το μοντέλο, τις βιβλιοθήκες και ότι άλλους πόρους χρειάζεται για να επεξεργαστεί το αίτημα. Το τελικό view γίνεται rendered και στέλνεται στον browser για να προβληθεί στον χρήστη. Επίσης αποθηκεύεται στην cash μνήμη για να παρέχεται σε μελλοντικά αιτήματα.

Απαιτήσεις συστήματος

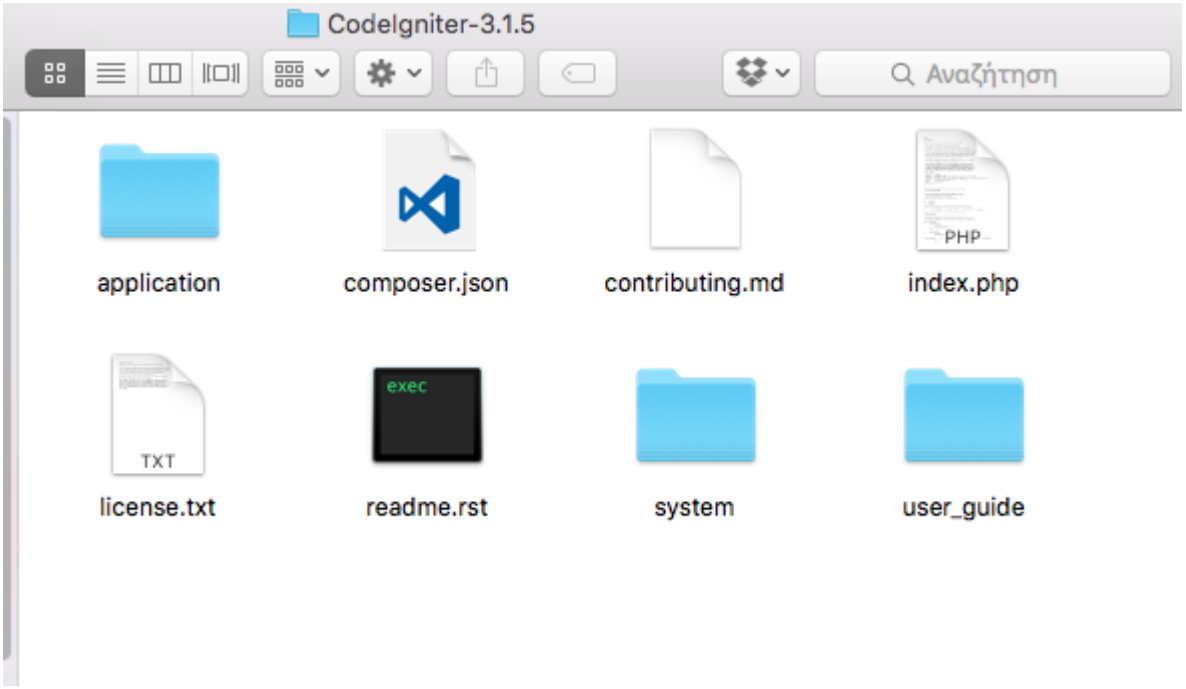
Το codeigniter όπως αναφέρθηκε είναι αρκετά αποδοτικό και ελαφρύ στη χρήση πόρων συστήματος. Οι απαιτήσεις της τελευταίας έκδοσης (3.1.5) διαμορφώνονται ως εξής.

- PHP 5.6 ή νεότερη έκδοση για πλήρη υποστήριξη. Λειτουργεί εξίσου καλά και σε 5.3.7 αλλά δε συνίσταται.
- MySQL 5.1 ή νεότερη (αν απαιτείται βάση δεδομένων)

Εγκατάσταση

Για να εγκαταστήσουμε το codeigniter ακολουθούμε την παρακάτω διαδικασία. Αρχικά το κατεβάζουμε από τη διεύθυνση: https://www.codeigniter.com/download

Με τη λήψη λαμβάνουμε ένα zip αρχείο το οποίο περιέχει μια δομή φακέλων και αρχείων όπως φαίνεται στην παρακάτω εικόνα.



Μεταφέρουμε τους φακέλους application & system καθώς και το αρχείο index.php στον web server μας στην τοποθεσία που επιθυμούμε. Στον φάκελο application/config βρίσκουμε το αρχείο config.php. Σε αυτό δηλώνουμε το base_URL το οποίο θα χρησιμοποιείται στα urls της εφαρμογής μας. Πρόκειται για τη διεύθυνση του server + το μονοπάτι της τοποθεσίας στην οποία μεταφέραμε τα αρχεία του codeigniter.

πχ έστω ο server με την IP 83.212.101.55 στον αρχικό κατάλογο διαθέτει τους φακέλους html, css, hua. Έστω ότι μεταφέραμε τα αρχεία στον φάκελο hua. Τότε το base_URL θα είναι "http://83.212.101.55/hua/".

Αν χρησιμοποιήσουμε βάση δεδομένων, τότε αρκεί να βρούμε το αρχείο databse.php στον φάκελο application/config και να ορίσουμε τα στοιχεία της βάσης μας όπως username, password, database.

```php
$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => 'root',
    'database' => 'dbhua',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
```

Δομή εφαρμογής

Όπως αναφέρθηκε προηγουμένως, στο codeingiter έχουμε τη δομή model - view - controller. Τα views παρουσιάζουν τα δεδομένα, το models έχουν τις μεθόδους που σχετίζονται με τη βάση δεδομένων και οι controllers λειτουργούν ως η σύνδεση μεταξύ view και model. Παρακάτω περιγράφεται η δομή της εφαρμογής καθώς και η λογική του διαχωρισμού των αρχείων.

Controllers:

Αρχικά υπάρχει ξεχωριστός controller για τις συσκευές, τους χρήστες, τις διευθύνσεις και τους διαχειριστές (Device.php, User.php, Ip.php, Admin.php). Δηλαδή ένας για κάθε οντότητα στην εφαρμογή. Αυτοί περιέχουν τις μεθόδους οι οποίες είναι υπεύθυνες για την ανάκτηση δεδομένων από τη βάση και την προβολή τους. Καλούν τις κατάλληλες μεθόδους των model και φορτώνουν τα κατάλληλα views για την κάθε προβολή των συσκευών, χρηστών, διευθύνσεων και διαχειριστών. Στη συνέχεια έχουμε ξεχωριστούς controllers για εισαγωγή, ενημέρωση και διαγραφή (Register.php, Update.php, Delete.php). Έτσι έχουμε διαχωρίσει το κυρίως μέρος της εφαρμογής σχετικά με τους controllers, με τέτοιο τρόπο ώστε να είναι εύκολα κατανοητή η σχεδίαση του κώδικα και η λογική της εφαρμογής. Επιπλέον είναι εύκολο

να επεκτείνουμε την εφαρμογή προσθέτοντας είτε νέα οντότητα, είτε να επεξεργαστούμε τον κώδικα για να προσθέσουμε λειτουργίες, καθώς παρέχεται ξεκάθαρη δομή. Στη συνέχεια υπάρχει ο controller για την εξαγωγή των δεδομένων σε excel (Excel_export.php) ο οποίος έχει τις μεθόδους για εξαγωγή και των 4 οντοτήτων σε excel. Ο controller που είναι υπεύθυνος για τη σύνδεση του χρήστη στην εφαρμογή, διαχειρίζεται το validation μέσω του model, το session και την αποσύνδεση. Τέλος ο controller Home.php καθορίζει την αρχική οθόνη της εφαρμογής από την οποία έχουμε όλες τις επιλογές διαθέσιμες, ενώ ο controller Content.php είναι υπεύθυνος για την προβολή των στατικών σελίδων της εφαρμογής.

Models:

Ο διαχωρισμός των μοντέλων είναι ακόμα πιο απλός από των controllers. Εδώ έχουμε ξεχωριστά τα μοντέλα για εισαγωγή, ενημέρωση και διαγραφή (Register_model.php, Udpate_model.php, Delete_model.php), καθένα από τα οποία διαθέτει μεθόδους για εισαγωγή, ενημέρωση και διαγραφή κάθε οντότητας. Για παράδειγμα το μοντέλο Register_model.php διαθέτει μεθόδους για εισαγωγή συσκευής, χρήστη, διεύθυνσης και διαχειριστή. Το μοντέλο που είναι υπεύθυνο για όλες τις λειτουργίες ανάκτησης δεδομένων από τη βάση, εκτός από αναζήτηση, είναι το Select_model.php και είναι αυτό που καλείται από τους controller Device.php, User.php, Ip.php, Admin.php. Οι λειτουργίες της αναζήτησης παρά το ότι πρόκειται για ανάκτηση δεδομένων, διαχωρίστηκαν σε ξεχωριστό μοντέλο καθώς αντιμετωπίστηκε ως ένα ξεχωριστό τμήμα της εφαρμογής. Το μοντέλο Login_model.php είναι αυτό που καλείται από τον controller Login.php για τη σύνδεση και αποσύνδεση ενός χρήστη στην εφαρμογή. Τέλος το μοντέλο Stats_model.php είναι υπεύθυνο για την προβολή των στατιστικών για κάθε οντότητα. Ο διαχωρισμός και εδώ προσφέρει ευκολότερη επεκτασιμότητα του κώδικα στο μέλλον για προσθήκη περισσότερων λειτουργιών.

Views:

Τα views δηλαδή οι οθόνες της εφαρμογής έχουν χωριστεί ως εξής. Ο φάκελος contentpages περιέχει όλα τα views των στατικών σελίδων της εφαρμογής. Ο φάκελος forms περιέχει όλα τα views που περιέχουν φόρμες εισαγωγής. Μέσα σε αυτό το φάκελο υπάρχει επιπλέον διαχωρισμός, σε φόρμες σύνδεσης, εισαγωγής, αναζήτησης και ενημέρωσης. Στον φάκελο selectviews περιέχονται τα views τα οποία προβάλουν δεδομένα μετά από ανάκτηση από τη βάση. Προβάλουν δηλαδή τα select των συσκευών, χρηστών, διευθύνσεων και διαχειριστών. Τα views για προβολή των αποτελεσμάτων της αναζήτησης για κάθε οντότητα βρίσκονται στο φάκελο searchselectviews. Επιπλέον έχουμε τα views για την αρχική οθόνη κάθε οντότητας (devicehome.php, userhome.php, iphome.php, adminhome.php) και τέλος έχουμε το

homeview.php το οποίο προβάλει την αρχική οθόνη της εφαμοργής.

Η δομή που περιγράφτηκε είναι σχεδιασμένη για την όσο το δυνατόν ευκολότερη κατανόηση του κώδικα με σκοπό την επεξεργασία και επέκταση στο μέλλον.

Κεφάλαιο 6. Συμπεράσματα

Στο κεφάλαιο αυτό παρουσιάζονται τα βασικότερα συμπεράσματα τα οποία προέκυψαν από την παρούσα πτυχιακή, ενώ παράλληλα προτείνονται μελλοντικές επεκτάσεις της εφαρμογής.

6.1 Συμπεράσματα

Σκοπός της πτυχιακής είναι η ανάπτυξη μιας web-based εφαρμογής η οποία θα χρησιμοποιείται για την καταγραφή και διαχείριση του εξοπλισμού (ηλεκτρονικού) του Πανεπιστημίου.

Οι web τεχνολογίες σήμερα, μας δίνουν τη δυνατότητα ανάπτυξης web εφαρμογών πλούσιων σε λειτουργίες. Παρέχονται στο χρήστη μέσω web browser, γεγονός που τις καθιστά ασυναγώνιστες στην προσβασιμότητα. Ανεξάρτητα από τη συσκευή ή την τοποθεσία του, ο χρήστης είναι ικανός να εκτελέσει λειτουργίες ανά πάσα στιγμή.

Απώτερος σκοπός της πτυχιακής είναι η διευκόλυνση του έργου της ομάδας τεχνικής υποστήριξης του Πανεπιστημίου μας, με της παροχή οργανωμένης πληροφορίας μέσω μιας εύχρηστης web εφαρμογής.

Το πανεπιστήμιο δε διαθέτει πληροφοριακό σύστημα στο οποίο να έχει πρόσβαση η ομάδα της τεχνικής υποστήριξης και να παρέχει πληροφορίες για τον εξοπλισμό του Πανεπιστημίου. Πολύ γρήγορα γίνεται αντιληπτό πόσο χρήσιμο είναι ένα τέτοιο σύστημα για τις εργασίες της ομάδας τεχνικής υποστήριξης.

- Η ομάδα λοιπόν χρησιμοποιώντας την εφαρμογή είναι σε θέση να αναζητεί άμεσα και στοχευμένα πληροφορίες οι οποίες είναι αναγκαίες για την επίλυση ενός ζητήματος. Επιπλέον τα μέλη της ομάδας, μπορούν να έχουν πρόσβαση στην εφαρμογή από τον υπολογιστή στον οποίο επιχειρούν επίλυση ζητήματος ή ακόμα και από το κινητό τους. Έτσι έχουν τις πληροφορίες που χρειάζονται διαθέσιμες πάντα και παντού.
- Η ανταλλαγή πληροφοριών ανάμεσα στην ομάδα και τον διαχειριστή συστήματος έγινε εύκολη υπόθεση πλέον, αφού και αυτός έχει πρόσβαση στην εφαρμογή, και κατ' επέκταση σε όλες τις πληροφορίες. Επιπλέον μειώθηκε ο χρόνος για την επικοινωνία ομάδας - διαχειριστή καθώς έχουν πρόσβαση στο ίδιο σύστημα και οποιεσδήποτε αλλαγές πραγματοποιεί ένας χρήστης της εφαρμογής, είναι άμεσα διαθέσιμες σε

όλους.

6.2 Μελλοντική εργασία

Η κυριότερη μελλοντική επέκταση της εφαρμογής αυτής, θα ήταν να μπορεί το προσωπικό του Πανεπιστημίου να συνδεθεί και να δηλώσει τυχόν πρόβλημα στη συσκευή του, για το οποίο θα ειδοποιούνται οι διαχειριστές.

Τέλος αξίζει να αναφέρουμε ως μελλοντική επέκταση, τη σύνδεση της εφαρμογής με ένα λογιστικό σύστημα ή σύστημα αγορών εξοπλισμού, για απευθείας καταχώρηση νέων συσκευών.

Βιβλιογραφικές Αναφορές

[1] https://mitpress.mit.edu/sites/default/files/titles/content/9780262015387_sch_0001.pdf

[2] http://www.zetta.net/about/blog/history-data-storage-technology

[3]https://eclass.hua.gr/modules/document/file.php/DIT136/%CE%94%CE%B9%CE%B1%CE%BB%CE%AD%CE%BE%CE%B5%CE%B9%CF%82/12_IO.pdf

[4] https://en.wikipedia.org/wiki/Hard_disk_drive

[5] http://www.slideshare.net/inam12/gr-1-34253962

[6]
https://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjl1MqS2obOAhUDShQKHTF3DDUQFggcMAA&url=http%3A%2F%2Fcourses.dbnet.ntua.gr%2Ffsr%2F3228%2F1_Eisagwgi.pdf&usg=AFQjCNHE-J3Ad3RVAQL9VoKnUcM9qANwcg&sig2=rNLDxZqJaJZ1daYA4lLuCg&cad=rja

[7] Σύγχρονα Πληροφοριακά Συστήματα Επιχειρήσεων
https://repository.kallipos.gr/pdfviewer/web/viewer.html?file=/bitstream/11419/2256/5/%CE%A3%CF%8D%CE%B3%CF%87%CF%81%CE%BF%CE%BD%CE%B1_%CE%A0%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CE%B1%CE%BA%CE%AC_%CE%A3%CF%85%CF%83%CF%84%CE%AE%CE%BC%CE%B1%CF%84%CE%B1_%CE%95%CF%80%CE%B9%CF%87%CE%B5%CE%B9%CF%81%CE%AE%CF%83%CE%B5%CF%89%CE%BD.pdf

[8] Ηλεκτρονική Συνταγογράφηση
https://el.wikipedia.org/wiki/%CE%97%CE%BB%CE%B5%CE%BA%CF%84%CF%81%CE%BF%CE%

BD%CE%B9%CE%BA%CE%AE_%CF%83%CF%85%CE%BD%CF%84%CE%B1%CE%B3%CE%BF%CE%B3%CF%81%CE%AC%CF%86%CE%B7%CF%83%CE%B7

[9] ΗΔΙΚΑ (Ηλεκτρονική Διακυβέρνηση Κοινωνικής Ασφάλισης)
http://www.idika.gr/files/odigies_xristi_iatrou_v3a.pdf

[10] Πληροφοριακά Συστήματα στον τομέα της υγείας
http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/6338/Kirimis.pdf?sequence=2

[11] Πληροφοριακά Συστήματα Υγείας
http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/6033/Milona.pdf?sequence=2&isAllowed=y

[12] Εφαρμογή ηλεκτρονικών φορολογικών υπηρεσιών στην Ελλάδα
http://okeanis.lib.teipir.gr/xmlui/bitstream/handle/123456789/1353/log_00004.pdf?sequence=1

[13] Web Application

https://en.wikipedia.org/wiki/Web_application

[14] World Wide Web

https://en.wikipedia.org/wiki/World_Wide_Web

[15] Ajax Programming

https://en.wikipedia.org/wiki/Ajax_(programming)

[16] The XMLHttpRequest Object

https://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/

[17] …….

……

[18] Operating Systems: Internals and Design principles

William Stallings, Chapter 16: Client/Server computing

[19] 3 types of web application architecture, (Article on February 18, 2016)

https://mobidev.biz/blog/3_types_of_web_application_architecture

[20] Δικτυακός προγραμματισμός: Καθηγήτρια ΕΜΠ θεοδόρα Βαρβαρίγου

[21] https://en.wikipedia.org/wiki/HTML

[22] About W3C

https://www.w3.org/Consortium

[23] A re-introduction to JavaScript - Mozilla Developer Network

https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

[24] Douglas Crockford: The JSON Saga

https://www.youtube.com/watch?v=-C-JoyNuQJs

[25] Douglas Crockford: JSON in Javascript

https://web.archive.org/web/20160710230817/http://www.json.org/js.html

[26] ECMA International: ECMA 404 - The JSON data interchange format

http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[27] ECMAScript 2016 Language Specification

http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

[28] Jesse James Garret (adaptive path): AJAX - A new approach to web applications (February 2005)

http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

[29] MySQL Documentation: MySQL PHP API

https://dev.mysql.com/doc/apis-php/en/

[30] MySQL Documentation: Connectors and APIs - Chapter 8 MySQL and PHP

https://dev.mysql.com/doc/connectors/en/apis-php.html

[31] How to install Apache, MySQL, PHP on ubuntu 14.04 (DigitalOcean)

https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04

[32] MySQL Workvench

https://dev.mysql.com/doc/workbench/en/

[33] How to install phpMyAdmin on Ubuntu | DigitalOcean

https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-12-04

[34] phpMyadmin Documentation

https://docs.phpmyadmin.net/en/latest/

[35] About phpMyAdmin

https://www.phpmyadmin.net

[36] MySQL reference manual: Using foreign key constraints

https://dev.mysql.com/doc/refman/5.6/en/create-table-foreign-keys.html