ΥΠΟΥΡΓΕΙΟ ΠΟΛΙΤΙΣΜΟΥ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Αποστολάκης Γ., Αραμπατζής Γ., Κατσαντώνης Μ.,
Κοτίνη Ι., Σταυρίδης Κ., Τζελέπη Σ.

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ

Γ' Τάξη ΕΠΑ.Λ.

ΣΗΜΕΙΩΣΕΙΣ **ΜΑΘΗΤΗ**

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΕΚΔΟΣΕΩΝ
«ΔΙΟΦΑΝΤΟΣ»

ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ
Πρόεδρος: **Γκλαβάς Σωτήριος**

ΓΡΑΦΕΙΟ ΕΡΕΥΝΑΣ, ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΕΦΑΡΜΟΓΩΝ Β΄
Προϊστάμενος: **Μάραντος Παύλος**
Επιστημονικά Υπεύθυνος: **Δρ. Τσαπέλας Θεοδόσιος**

ΣΥΓΓΡΑΦΙΚΗ ΟΜΑΔΑ:
Αποστολάκης Γιάννης, Εκπαιδευτικός Πληροφορικής
Αραμπατζής Γιώργος, Εκπαιδευτικός Πληροφορικής
Κατσαντώνης Μενέλαος, Εκπαιδευτικός Πληροφορικής
Κοτίνη Ισαβέλλα, Σχολική Σύμβουλος Πληροφορικής
Σταυρίδης Κωνσταντίνος, Εκπαιδευτικός Πληροφορικής
Τζελέπη Σοφία, Σχολική Σύμβουλος Πληροφορικής

ΕΠΙΜΕΛΕΙΑ ΣΥΝΤΟΝΙΣΜΟΣ ΟΜΑΔΑΣ:
Κοτίνη Ισαβέλλα, Σχολική Σύμβουλος Πληροφορικής
Τζελέπη Σοφία, Σχολική Σύμβουλος Πληροφορικής

ΕΠΙΤΡΟΠΗ ΚΡΙΣΗΣ:
Βογιατζής Ιωάννης, Επίκουρος Καθηγητής Τ.Ε.Ι. Αθηνών
Μελετίου Γεώργιος, Μηχανικός Πληροφορικής MSc
Παραδείση Άρτεμις, Εκπαιδευτικός Πληροφορικής

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή στην Ανάπτυξη Λογισμικού

1.1. Το Λογισμικό, Κατηγορίες Λογισμικού 6

1.2. Το πεδίο της τεχνολογίας Λογισμικού 7

1.3. Λογισμικό και Πληροφοριακά Συστήματα 10

1.4. Διοίκηση - Διαχείριση έργου ανάπτυξης λογισμικού 12

1.5. Διαδίκτυο και Επικοινωνία 15

1.6. Τεχνολογίες Υπολογιστικού Σύννεφου 32

1.7. Αρχιτεκτονική Εφαρμογών 36

ΚΕΦΑΛΑΙΟ 2: Κύκλος ζωής ανάπτυξης συστήματος

2.1 Ανάπτυξη Συστήματος 48

2.2 Κύκλος Ζωής Ανάπτυξης Συστήματος 50

2.3 Κλασσικές προβλέψιμες προσεγγίσεις του Κύκλου ζωής Ανάπτυξης Συστήματος 53

2.4 Ευέλικτες – Προσαρμοστικές προσεγγίσεις του Κύκλου ζωής Ανάπτυξης Συστήματος 55

2.5 Στάδια του Κύκλου ζωής Ανάπτυξης Συστήματος 58

2.6 Μεθοδολογίες, μοντέλα, εργαλεία και τεχνικές 60

2.7 Δομημένη Ανάπτυξη Συστήματος (SADT) 63

2.8 Αντικειμενοστραφής Ανάπτυξη Συστήματος 64

2.9 Σύγχρονες Τάσεις στην Ανάπτυξη Συστήματος 65

ΚΕΦΑΛΑΙΟ 3: Ανάλυση απαιτήσεων και καθορισμός προδιαγραφών

3.1 Κατηγορίες απαιτήσεων 73

3.2 Διαδικασία προσδιορισμού απαιτήσεων 75

3.3 Μοντέλα ανάλυσης απαιτήσεων 78

3.4 Καθορισμός Προδιαγραφών 92

ΚΕΦΑΛΑΙΟ 4: Σχεδιασμός Αρχιτεκτονικής Συστήματος και Μονάδων Λογισμικού

4.1 Εισαγωγή στις Αρχιτεκτονικές Σχεδίασης 97

4.2 Σχεδιασμός Προσανατολισμένος στις διαδικασίες 99

4.3 Σχεδιασμός Προσανατολισμένος στα Αντικείμενα 106

4.4 Σχεδιασμός Διεπαφής Χρήστη 109

ΚΕΦΑΛΑΙΟ 5: Σχεδιασμός και υλοποίηση διαδικτυακών εφαρμογών

5.1 Εισαγωγή 124

5.2 Σχεδιασμός και Ενσωμάτωση Δικτύου 125

5.3 Το περιβάλλον ανάπτυξης και η αρχιτεκτονική της εφαρμογής 129

5.4 Προγραμματισμός εξυπηρετητή σε PHP 153

ΠΑΡΑΡΤΗΜΑ 177

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στην ανάπτυξη λογισμικού

Περιεχόμενα

1.1. Το Λογισμικό, Κατηγορίες Λογισμικού
1.2. Το πεδίο της τεχνολογίας Λογισμικού
1.3. Λογισμικό και Πληροφοριακά Συστήματα
1.4. Διοίκηση-Διαχείριση έργου ανάπτυξης λογισμικού
1.5. Διαδίκτυο και Επικοινωνία
1.6. Τεχνολογίες Υπολογιστικού Σύννεφου
1.7. Αρχιτεκτονική Εφαρμογών

Διδακτικοί Στόχοι

Στόχοι του κεφαλαίου αυτού είναι:

- Να περιγράφουν τις κατηγορίες λογισμικού.
- Να δίνουν τα χαρακτηριστικά των εφαρμογών λογισμικού στο Διαδίκτυο .
- Να αντιλαμβάνονται και να αποτυπώνουν τη δομή ενός Πληροφοριακού Συστήματος.
- Να οργανώνουν και να συμμετέχουν σε ομάδες ανάπτυξης λογισμικού εφαρμογών.
- Να διακρίνουν τα συστατικά στοιχεία των διαφόρων Διαδικτυακών δομών καθώς και να καταθέτουν προτάσεις για την βέλτιστη αξιοποίησή τους.

1.1. Το Λογισμικό, Κατηγορίες Λογισμικού

Θα μπορούσαμε να ορίσουμε το **λογισμικό** ως ένα σύνολο προγραμμάτων -δομές δεδομένων και εντολές- καθώς και υλικού τεκμηρίωσης. Οι δομές δεδομένων επιτρέπουν στο πρόγραμμα να διαχειρίζεται πληροφορίες, ενώ οι εντολές, όταν εκτελούνται, παρέχουν στους χρήστες τις επιθυμητές λειτουργίες. Η τεκμηρίωση περιγράφει τον τρόπο λειτουργίας και χρήσης των προγραμμάτων.

Στην ορολογία της επιστήμης των υπολογιστών μπορούμε να διακρίνουμε τις παρακάτω **κατηγορίες λογισμικού** (με βάση τον τρόπο παρέμβασης) χωρίς να είναι πλήρως διακριτές μεταξύ τους:

- **Λογισμικό Συστήματος** (system software): λειτουργικά συστήματα, λειτουργικά περιβάλλοντα, οδηγοί συσκευών, λογισμικά δικτύου μεταγλωττιστές, λογισμικά για αντιμετώπιση ιών, λογισμικά περιήγησης στο Διαδίκτυο κ.α.

- **Λογισμικό Εφαρμογών** (applications software): κάθε λογισμικό που σκοπό έχει να αντιμετωπίσει συγκεκριμένες ανάγκες του χρήστη ή/και συγκεκριμένου φορέα ή επιχείρησης (π.χ. Διαχείριση Φαρμακείου, Διαχείριση Βιοϊατρικού εξοπλισμού κ.α.). Το λογισμικό αυτό εντάσσεται στα λεγόμενα εσωτερικά Πληροφοριακά Συστήματα οργανισμών ή φορέων. Το Λογισμικό Διαδικτυακών Εφαρμογών (web applications) αποτελεί μια ειδική κατηγορία εφαρμογών λογισμικού. Είναι το λογισμικό που αξιοποιεί τις τεχνολογίες του Διαδικτύου για την παροχή ηλεκτρονικών υπηρεσιών σε εστιασμένους χρήστες ή σε ένα ευρύτερο κοινό ενός φορέα ή μιας επιχείρησης (π.χ. Υποβολή φορολογικών δηλώσεων στο Υπουργείο Οικονομικών, Αγορά αγαθών π.χ. βιβλίων από ηλεκτρονικά καταστήματα).

- **Λογισμικό για την αύξηση της παραγωγικότητας** (productivity software): λογισμικό που χρησιμοποιείται από ευρύ φάσμα χρηστών για την εκτέλεση κάποιων λειτουργιών ευρείας χρήσης (π.χ. επεξεργαστές κειμένου, επεξεργαστές πινάκων, Εργαλεία παρουσιάσεων, Εργαλεία CASE κ.λπ.).

- **Επιστημονικό Λογισμικό** (scientific software): λογισμικό που χρησιμοποιείται από επιστημονικούς κλάδους (Ιατρική, αστρονομία, βιολογία αρχιτεκτονική κ.λπ.) για εστιασμένους ερευνητικούς και διαχειριστικούς σκοπούς. π.χ. τα λογισμικά για γενετιστές-βιολόγους, που αφορούν στην περιγραφή του γενετικού αποτυπώματος.

- **Λογισμικό Τεχνητής Νοημοσύνης** (artificial intelligence software): λογισμικό για ρομπότ, νευρωνικά δίκτυα, λογισμικό εμπείρων συστημάτων και συστημάτων λήψης απόφασης.

- **Ενσωματωμένο Λογισμικό** (embedded software): λογισμικό που έχει ενσωματωθεί σε υπολογιστικές μηχανές ειδικού σκοπού (π.χ. holter καταγραφής της αρτηριακής πίεσης και καρδιακής συχνότητας, ηλεκτρονικά πιεσόμετρα κ.λπ.).

Θα μπορούσαμε επίσης να αναφέρουμε και να κατηγοριοποιήσουμε το λογισμικό σε -εμπορικό ή μη εμπορικό- χαρακτηρίζοντας ως **εμπορικό** το λογισμικό που χρειάζεται να το αγοράσουμε για να το χρησιμοποιήσουμε. Επίσης θα μπορούσαμε να μιλήσουμε για **ανοικτό** υπό την έννοια της δυνατότητας αγοράς του πηγαίου κώδικα (με δυνατότητα επαναχρησιμοποίησης του και συμπλήρωσης του) και κλειστό λογισμικό όταν υπάρχει η δυνατότητα αγοράς μόνο του εκτελέσιμου κώδικα. Υπάρχει και το λεγόμενο **ελεύθερο** λογισμικό υπό την έννοια ότι ο «ιδιοκτήτης-κατασκευαστής» του το διαθέτει χωρίς κάποιο κόστος (www.opensoft.gr). Μια ιδιαίτερη κατηγορία είναι το **ΕΛ/ΛΑΚ** (Ελεύθερο Λογισμικό / Λογισμικό Ανοικτού Κώδικα) το οποίο συνδυάζει και τα δύο παραπάνω χαρακτηριστικά (https://ellak.gr).

Το **Πανελλήνιο Σχολικό Δίκτυο** (www.sch.gr) διαθέτει Διαδικτυακή πύλη για λογισμικό στην εκπαίδευση (εκπαιδευτικό λογισμικό) το οποίο είναι ΕΛ/ΛΑΚ (opensoft.sch.gr).

1.2. Το πεδίο της τεχνολογίας λογισμικού

Η **τεχνολογία λογισμικού** μπορεί να οριστεί ως ο κλάδος της πληροφορικής που ασχολείται με την εύρεση και θεμελίωση μεθόδων και με το συστηματικό σχεδιασμό και την ανάπτυξη προϊόντων λογισμικού. Συγκεκριμένα μελετά την εφαρμογή προσεγγίσεων για την ανάπτυξη, λειτουργία και συντήρηση του λογισμικού www.ieee.org.

Κατ' αναλογία με την παραγωγή βιομηχανικών προϊόντων η διαδικασία της ανάπτυξης του λογισμικού μπορεί να παρίσταται με τα παρακάτω τρία βήματα:

Σχήμα 1.1: *Τα βήματα ανάπτυξης λογισμικού*

Στην **Ανάλυση** καθορίζονται σε συνεργασία με τους τελικούς χρήστες (end-users) τι ακριβώς θα κάνει το λογισμικό - ποιες ακριβώς λειτουργίες- και ποιοι ενδεχόμενοι περιορισμοί τίθενται στη λειτουργία του. Η δραστηριότητα που εμπεριέχει αυτό το βήμα λέγεται **προσδιορισμός απαιτήσεων.** Σε αυτή προδιαγράφουμε το τι θα κάνει το σύστημα. Η ομάδα εργασίας-ανάπτυξης (του αναδόχου) συνεργάζεται με τον

πελάτη (οργανισμό ή εταιρεία ή ιδιώτη) και τους τελικούς χρήστες με σκοπό να συμφωνήσουν από κοινού στη λειτουργικότητα του λογισμικού και την παραγωγή του *εγγράφου προδιαγραφών απαιτήσεων λογισμικού.*

Κατά την **Κατασκευή** αναπτύσσεται προϊόν που ικανοποιεί τις απαιτήσεις του βήματος της ανάλυσης. Η **Σχεδίαση** απαντά στο ερώτημα πώς θα κατασκευαστεί το λογισμικό έτσι ώστε να κάνει αυτά που περιγράφουν οι απαιτήσεις. Συνήθως περιλαμβάνει δύο επίπεδα. Το πρώτο επίπεδο είναι η αρχιτεκτονική σχεδίαση (architectural design) που αφορά τον προσδιορισμό του σκελετού του λογισμικού (κατ' αναλογία με την αρχιτεκτονική των κτηρίων) και το δεύτερο επίπεδο είναι η λεπτομερής σχεδίαση (detailed design) που αφορά στην οργάνωση και επικοινωνία των επιμέρους μονάδων του λογισμικού. Η **Κωδικοποίηση** είναι η δραστηριότητα που περιλαμβάνει κυρίως τον προγραμματισμό και παράγει το τελικό προϊόν του λογισμικού.

Στον **Έλεγχο** τέλος βλέπουμε αν το λογισμικό που αναπτύχθηκε ικανοποιεί τις αρχικές απαιτήσεις που τέθηκαν. Ο έλεγχος απαιτείται για να διαπιστώσουμε την «ορθότητα» του λογισμικού. Επεκτείνεται επίσης και στον έλεγχο κάποιων ποιοτικών χαρακτηριστικών του λογισμικού όπως η αποδοτικότητα, η μεταφερσιμότητα, η αξιοπιστία κ.τ.λ.

Η **συντήρηση** του λογισμικού μπορεί να εκληφθεί ως επανάληψη των δραστηριοτήτων του προσδιορισμού απαιτήσεων, της σχεδίασης, του προγραμματισμού και του ελέγχου σε ένα προϊόν λογισμικού που ήδη έχει ολοκληρωθεί και παραδοθεί. Η συντήρηση πολύ συχνά πραγματοποιείται από μία διαφορετική ομάδα από αυτή που το είχε αναπτύξει. Αποτελεί αντικείμενο μελέτης της τεχνολογίας λογισμικού λόγω της μεγάλης σημασίας της στην βιωσιμότητα του τελικού προϊόντος.

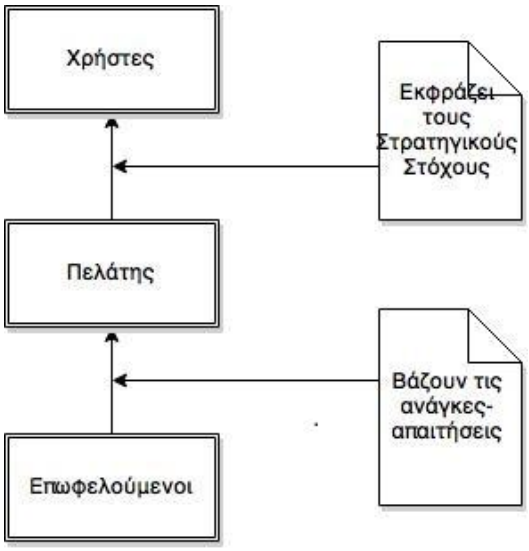
Συνήθως οι **αποδέκτες ενός έργου ανάπτυξης λογισμικού** ανήκουν σε μία από τις παρακάτω κατηγορίες:

- **οι χρήστες,** το πρόσωπο ή ομάδα ατόμων, στελέχη του οργανισμού ή της εταιρίας που θα αξιοποιήσουν το λογισμικό που θα αναπτυχθεί, θα εισάγουν τα δεδομένα, θα αναζητήσουν πληροφορία, θα πάρουν αποτελέσματα και θα δώσουν εκτυπώσεις κ.λπ.

- **ο πελάτης,** είναι η εταιρεία, ο οργανισμός ή ο ιδιώτης που ξεκινά την διαδικασία ανάπτυξης υπό την έννοια ότι χρηματοδοτεί το έργο και δίνει αυτός ή τα στελέχη του (και οι χρήστες) τις απαιτήσεις στην ομάδα ανάπτυξης.

- **οι επωφελούμενοι από τη λειτουργία του λογισμικού** αποδέκτες των υπηρεσιών του λογισμικού. Εάν για παράδειγμα πρόκειται για το λογισμικό διαχείρισης ΑΜΚΑ-Αριθμός Μητρώου Κοινωνικής Ασφάλισης του

Υπουργείου Απασχόλησης, οι επωφελούμενοι είναι όλοι οι Έλληνες πολίτες καθώς και όλοι οι δημόσιοι ή ιδιωτικοί φορείς.



Σχήμα 1.2: *Οι τελικοί αποδέκτες του λογισμικού*

Υπεύθυνη για την ανάπτυξη, θα πρέπει να είναι κατάλληλα στελεχωμένη ομάδα π.χ. της αναδόχου εταιρίας, η οποία μπορεί να ονομαστεί ***Ομάδα Εργασίας Αναδόχου (ΟΕΑ).*** Η ομάδα θα διοικείται από τον ***Υπεύθυνο - Συντονιστή του συνολικού έργου.*** Η ΟΕΑ θα αποτελέσει τον κύριο κορμό ανάπτυξης του έργου, και θα επιβλέπει, θα οργανώνει, θα προδιαγράφει, θα εκτελεί το έργο από τη σύλληψη του μέχρι την υλοποίηση, την εγκατάσταση και την πλήρη αποδοχή του από τους τελικούς χρήστες.

Ο οργανισμός (ή εταιρεία, ή ιδιώτης) θα πρέπει αντίστοιχα, να δημιουργήσει ***Ομάδα Εργασίας Οργανισμού (ΟΕΟ),*** αποτελούμενη από εντεταλμένους χρήστες των εμπλεκόμενων τμημάτων, με επικεφαλής στέλεχος του Οργανισμού που θα αποκαλείται ***Υπεύθυνος Έργου του Οργανισμού*** και ο οποίος θα συντονίζει το έργο από την μεριά του Οργανισμού. Επίσης, ο Οργανισμός θα πρέπει να ορίσει κατάλληλο στέλεχος του, που θα είναι υπεύθυνο για την ενσωμάτωση όλων των ποιοτικών χαρακτηριστικών που σταδιακά θα προδιαγράφει. Η άμεση και καθημερινή συμμετοχή τόσο των στελεχών όσο και των χρηστών του Οργανισμού στη διαδικασία ανάπτυξης, θεωρείται απαραίτητη για την εξασφάλιση της πληρότητας και ποιότητας των λειτουργικών απαιτήσεων αλλά και τη μεταφορά της τεχνογνωσίας που ο οργανισμός προσδοκά.

*Η **Ομάδα Εργασίας Οργανισμού (ΟΕΟ)** μπορεί να έχει τις παρακάτω αρμοδιότητες:*

- Καθορισμός των προδιαγραφών του έργου.
- Διαμόρφωση, σε συνεργασία με την ΟΕΑ, της τελικής μορφής των εγχειριδίων τεκμηρίωσης (Προσδιορισμός Απαιτήσεων, Σχεδιασμού, Εγχειριδίων χρήσης κ.λπ.) καθώς και της μορφής της άμεσης βοήθειας (on line Help) που θα παρέχεται στο λογισμικό.

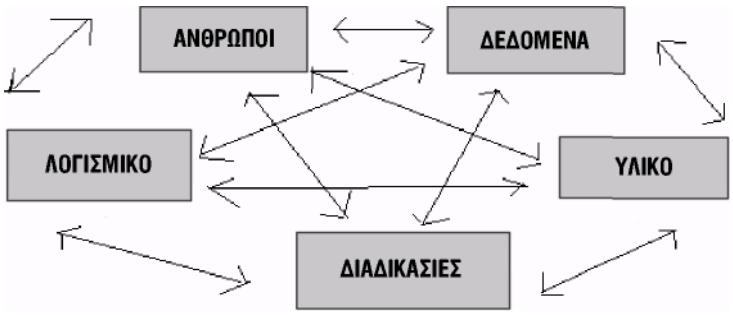
- Υποστήριξη και υποβοήθηση του αναδόχου στην υλοποίηση του έργου.
- Διευκόλυνση του αναδόχου στην επικοινωνία και τις επαφές με τα στελέχη και τους χρήστες του Οργανισμού.
- Παρακολούθηση της προόδου των εργασιών. Έλεγχος των παραδοτέων της ΟΕΑ για την πλήρη αντιστοίχιση με τις προδιαγραφές του έργου.

1.3. Λογισμικό και Πληροφοριακά Συστήματα

Πληροφοριακό Σύστημα μιας επιχείρησης/οργανισμού είναι ένα σύστημα που αποτελείται από ανθρώπους, διαδικασίες και εξοπλισμό (Υλικό, Λογισμικό, Δεδομένα) μέσω των οποίων παράγονται, φυλάσσονται, διακινούνται και μετασχηματίζονται οι πληροφορίες που είναι χρήσιμες για την επίτευξη των σκοπών της επιχείρησης/οργανισμού.

Οι συνιστώσες του είναι:

- άνθρωποι
- διαδικασίες
- λογισμικό
- δεδομένα
- υλικό



Σχήμα 1.3: *Οι συνιστώσες ενός Πληροφοριακού Συστήματος.*

Το Πληροφοριακό Σύστημα εξυπηρετεί όλες τις οργανωτικές μονάδες του οργανισμού/επιχείρησης με στόχο την επίτευξη του κοινού σκοπού. Θέλοντας να γίνει κατά το δυνατόν συγκεκριμένο το τι είναι Πληροφοριακό Σύστημα θα δώσουμε μια συνοπτική περιγραφή των συνιστωσών του:

α) Άνθρωποι

Οι άνθρωποι ενός Πληροφοριακού Συστήματος θα μπορούσαν να ταξινομηθούν σε δυο κατηγορίες:

- στους χρήστες (users) και
- στους χειριστές (operators) του συστήματος.

Στην πρώτη κατηγορία ανήκουν οι τελικοί χρήστες (end-users), οι προϊστάμενοι (user-managers) και ο ιδιοκτήτης του συστήματος, δηλαδή αυτός που έδωσε την εντολή για την ανάπτυξη (και χρηματοδοτώντας την ταυτόχρονα) και τη λειτουργία του συστήματος ενώ στην δεύτερη κατηγορία ανήκουν οι χειριστές των Η/Υ, δηλαδή όσοι εισάγουν στοιχεία (data entry), όσοι συντηρούν το υλικό και/ή το λογισμικό κ.λπ.

β) Διαδικασίες

Διαδικασία είναι μια σειρά από οδηγίες, οι οποίες καθορίζουν τον τρόπο με τον οποίο θα ενεργήσουν οι άνθρωποι σε συγκεκριμένες περιστάσεις και απευθύνονται στους ανθρώπους που συμμετέχουν στο σύστημα. Επιγραμματικά μπορούμε να πούμε ότι μια διαδικασία:

- υποστηρίζει ανθρώπινες δραστηριότητες,
- εξασφαλίζει τι πληροφορία θα έχει ο συγκεκριμένος άνθρωπος την συγκεκριμένη χρονική στιγμή,
- δίνει τον τρόπο μετασχηματισμού της πληροφορίας.

Στα Πληροφοριακά Συστήματα έχουμε διαδικασίες που αφορούν τους χρήστες και διαδικασίες που αφορούν τους χειριστές. Για τους χρήστες υπάρχουν οδηγίες για το πώς θα αξιοποιηθεί το υλικό (Hardware), το λογισμικό (Software) και τα δεδομένα (data) ώστε τελικά να παράγουμε το επιθυμητό αποτέλεσμα ενώ για τους χειριστές είναι οδηγίες το πως ξεκινά και πως κλείνει το σύστημα του Η/Υ, πως εξασφαλίζονται τα αντίγραφα ασφαλείας (backup), πως επαναφέρονται τα δεδομένα στο σύστημα (restore), ποια είναι τα βασικά σημεία συντήρησης του συστήματος, πως γίνεται η εποικοδομητική "τακτοποίηση" του συστήματος, πως επιτυγχάνονται οι διασυνδέσεις με άλλα Συστήματα Η/Υ ή Διεθνή Δίκτυα, πως εξασφαλίζεται η "γενικότερη" ασφάλεια του συστήματος, πως γίνεται ο ορισμός και η εξουσιοδότηση νέων χρηστών κ.λπ.

γ) Λογισμικό (Software)

Το λογισμικό ενός Πληροφοριακού Συστήματος μπορούμε να το διακρίνουμε στις κατηγορίες όπως αυτές παρουσιάστηκαν στην πρώτη ενότητα του κεφαλαίου.

δ) Δεδομένα (data)

Ο όρος δεδομένα (data) σημαίνει μια παράσταση γεγονότων, εννοιών ή εντολών κατά τέτοιο τρόπο που να είναι σε μορφή κατάλληλη για επικοινωνία, ερμηνεία ή επεξεργασία από άνθρωπο ή από το αυτοματοποιημένο μέσο (π.χ. Η/Υ).

ε) Υλικό (Hardware)

Η συνιστώσα αυτή είναι όλος ο εξοπλισμός (Hardware) του/των Η/Υ που χρησιμοποιούνται στο Πληροφοριακό Σύστημα. Σε αυτόν περιλαμβάνονται και οι περιφερειακές συσκευές (π.χ. εκτυπωτές κ.λπ.) καθώς και ο πιθανός δικτυακός εξοπλισμός (π.χ. καλώδια, κάρτες κ.λπ.).

1.4. Διοίκηση-Διαχείριση έργου ανάπτυξης λογισμικού

Το εγχειρίδιο που εξέδωσε το Project Management Institute (PMI), ορίζει ως **έργο** το *...προσωρινό εγχείρημα που στοχεύει στη δημιουργία ενός μοναδικού προϊόντος ή υπηρεσίας. Προσωρινό σημαίνει ότι κάθε έργο έχει καθορισμένο τέλος. Μοναδικό σημαίνει ότι το προϊόν ή η υπηρεσία διαφέρει κατά διακριτό τρόπο από όλα τα υπόλοιπα παρόμοια προϊόντα ή υπηρεσίες*.

Ο Turner ορίζει ως **έργο** το *...εγχείρημα κατά το οποίο ανθρώπινοι πόροι, μηχανές, οικονομική πόροι και πρώτες ύλες οργανώνονται κατά καινοφανή τρόπο, με στόχο την ανάληψη συγκεκριμένου αντικειμένου εργασιών που έχουν συγκεκριμένες προδιαγραφές και υπόκεινται σε δεδομένους κοστολογικούς και χρονικούς περιορισμούς, ώστε να παραχθεί μία επωφελής μεταβολή, η οποία ορίζεται μέσω ποσοτικών και ποιοτικών στόχων*.

Διοίκηση Έργου (Project Management) είναι η διεργασία συνδυασμού συστημάτων, τεχνικών και γνώσης με σκοπό την ολοκλήρωση ενός έργου μέσα σε καθορισμένα πλαίσια χρόνου, προϋπολογισμού, ποιότητας και σκοπού/αντικειμένου.

Το εγχειρίδιο για τη **διαχείριση έργου** (Project Management Body Of Knowledge, PMBOK) ορίζει ως διαχείριση έργου τη διαδικασία κατά την οποία: *...εφαρμόζουμε γνώσεις, δεξιότητες, εργαλεία και τεχνικές κατά την εκτέλεση των δραστηριοτήτων του έργου, με στόχο να ικανοποιήσουμε τις απαιτήσεις και τις προσδοκίες των συμμετεχόντων.*

Στις μέρες μας η ανάπτυξη της επιστημονικής διαχείρισης έργων συνοδεύεται από πληθώρα **λογισμικών διαχείρισης έργων**. Στον Πίνακα 1.1 παρατίθενται κάποια από αυτά.

Πίνακας 1.1: *Λογισμικά Διαχείρισης Έργων*

ΛΟΓΙΣΜΙΚΟ	ΕΤΑΙΡΕΙΑ
Timeline	*Symantic*
Instaplan	*Instaplan*
Project Scheduler	*Scitor*
Mac Project	*Clavis*
FlowCharting	*Patton & Patton*
Project Management	*Primavera Systems*
MS-Project	*Microsoft*
Gantter	*InQuest Technologies*

Στην ενότητα αυτή θα δώσουμε περισσότερα στοιχεία για το λογισμικό Gantter (http://www.gantter.com) το οποίο είναι ένα δωρεάν εργαλείο λογισμικού στο Υπολογιστικό σύννεφο (cloud-based). Συνεργάζεται με όλους τους γνωστούς

παρόχους Cloud υπηρεσιών (βλ. ενότητα 1.6 για το cloud) όπως το Google Drive, Dropbox και OneDrive της Microsoft και έχει δυνατότητες αλληλεπίδρασης με τις εφαρμογές Google Apps. Επίσης οι χρήστες έχουν την δυνατότητα να εισάγουν, να επεξεργάζονται και να εξάγουν αρχεία από το λογισμικό MS Project.

Από την αρχική σελίδα (http://www.gantter.com) ο χρήστης από το κουμπί "Start Now" (Σχήμα 1.4) επιλέγει τον τρόπο με τον οποίο επιθυμεί να συνδεθεί στην εφαρμογή (εναλλακτικά μπορεί να βρεθεί στο περιβάλλον του Google Drive και να ενσωματώσει το λογισμικό Gantter στις εφαρμογές που χρησιμοποιεί).

Σχήμα 1.4: *Αρχική Οθόνη Gantter*

Στη συνέχεια ο χρήστης, μεταφέρεται στην κύρια οθόνη λειτουργιών από όπου μπορεί να διαχειριστεί τον χρονοπρογραμματισμό ενός έργου σχεδιάζοντας ένα μοντέλο ελέγχου και παρακολούθησης των ενεργειών (tasks) και των πόρων (resources) που απαιτούνται για την επιτυχή ολοκλήρωση του.

Για την καλύτερη κατανόηση των δυνατοτήτων του Gantter θα χρησιμοποιήσουμε ως παράδειγμα διαχείρισης έργου το **έργο της ανάπτυξης ενός Διαδικτυακού Τόπου.**

Αρχικά ο χρήστης ορίζει τις κύριες και τις επιμέρους ενέργειες που απαιτούνται για την δημιουργία ενός Δικτυακού Τόπου και κάνει μια εκτίμηση του χρόνου που θα χρειαστεί ώστε αυτές να ολοκληρωθούν. Επίσης μπορεί να κάνει και έναν προϋπολογισμό στους πόρους που θα απαιτηθούν (άνθρωποι, υλικά) για την υλοποίηση του έργου. Σημαντικό επίσης είναι ο χρήστης να γνωρίζει πότε οι

ενέργειες (tasks) ξεκινούν και ποιες από αυτές προαπαιτούν την ολοκλήρωση κάποιας άλλης ενέργειας προκειμένου να υλοποιηθούν.

Μέσω της κύριας οθόνης λειτουργίας ο χρήστης μπορεί να καταχωρήσει όλες τις απαιτούμενες ενέργειες (tasks), να ορίσει ποιες είναι κύριες και ποιες δευτερεύουσες, να θέσει τις ημερομηνίες έναρξης και λήξης των εργασιών, να δηλώσει τους πόρους (resources) (εργαζόμενοι, υλικά) που χρειάζονται και να ορίσει τις σχέσεις μεταξύ των εργασιών. Όλα τα ανωτέρω αποτυπώνονται πλήρως στην οθόνη του Gantter (Σχήμα 1.5).

Ενδεικτικά τα κύρια και επιμέρους (δευτερεύοντα) στάδια για την **δημιουργία ενός Διαδικτυακού τόπου** που θα αποτυπωθούν στο Gantter είναι τα εξής:

1. **Ανάλυση Απαιτήσεων** – *Έναρξη έργου*

   - Αποτύπωση σκοπού, ανάλυση λειτουργικών αναγκών
   - Ορισμός λειτουργικών απαιτήσεων σε υλικό και λογισμικό
   - Καθορισμός κόστους ανάπτυξης και συντήρησης
   - Καθορισμός χρονοδιαγράμματος υλοποίησης

2. **Σχεδίαση Δικτυακού Τόπου**

   - Σχεδιασμός αρχιτεκτονικής (Hardware/Software)
   - Σχεδιασμός εικαστικού θέματος και δομής

3. **Υλοποίηση, Ανάπτυξη Δικτυακού Τόπου**

   - Παραμετροποίηση Web Server (Hosting)
   - Εγκατάσταση λογισμικού στον Web Server
   - Ανάπτυξη Δικτυακού Τόπου (δομή, σελίδες, μενού)
   - Καταχώρηση περιεχομένου (κείμενα, γραφικά, εικόνες, κ.α.)

4. **Πιλοτική Λειτουργία και Ανάδραση Αποτελεσμάτων**

   - Κλειστή Δημοσίευση και δοκιμές λειτουργίας
   - Διορθώσεις επί των αποτελεσμάτων της πιλοτικής λειτουργίας

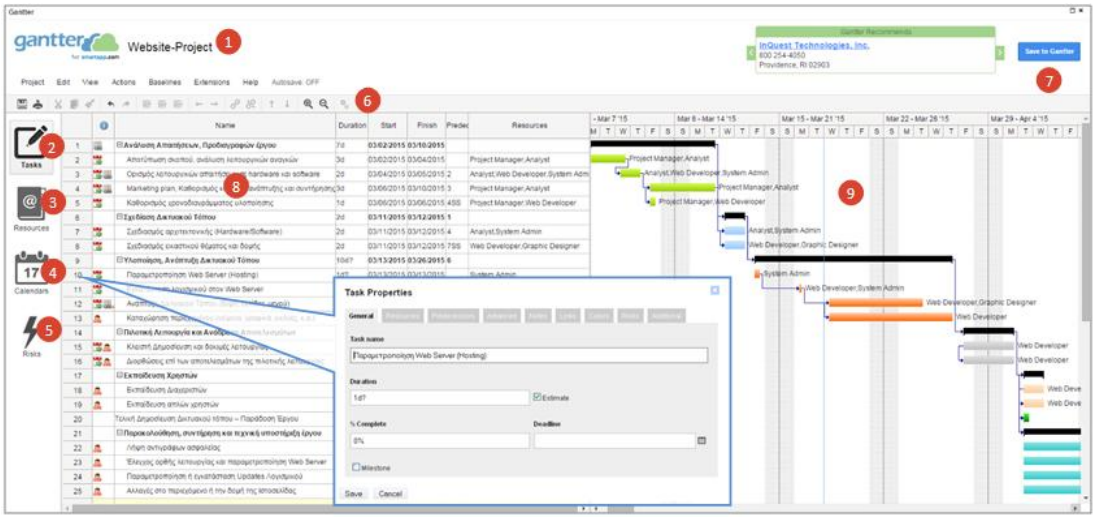
5. **Εκπαίδευση Χρηστών**

   - Εκπαίδευση Διαχειριστών
   - Εκπαίδευση απλών χρηστών

6. **Δημοσίευση Δικτυακού τόπου** – *Παράδοση Έργου*

7. **Παρακολούθηση, συντήρηση και τεχνική υποστήριξη έργου** - *Υποστήριξη έργου*

- Λήψη αντιγράφων ασφαλείας
- Έλεγχος ορθής λειτουργίας και παραμετροποίηση Web Server
- Παραμετροποίηση και εγκατάσταση επικαιροποιήσεων Λογισμικού
- Αλλαγές στο περιεχόμενο ή την δομή της Ιστοσελίδας



Σχήμα 1.5: *Κύρια Οθόνη λειτουργιών του Gantter*

Πίνακας 1.2: *Βασικές λειτουργίες του Gantter*

1	**Τίτλος έργου**	Ορισμός ονόματος Project.
2	**Tasks**	Οθόνη διαχείρισης των ενεργειών και του διαγράμματος χρονοπρογραμματισμού Gantt.
3	**Resources**	Οθόνη ορισμού των πόρων (εργαζόμενοι και υλικά) καθώς και εκτίμηση του κόστους.
4	**Calendars**	Καθορισμός του ημερολογίου χρήσης. Δυνατότητα επιλογής 3 ημερολογίων (Standard, 24-Hours, Night Shift), τα οποία μπορούν να τροποποιηθούν ανάλογα με τις ανάγκες.
5	**Risks**	Διαχείριση κινδύνου σε επίπεδο έργου ή ενεργειών. Δυνατότητα ορισμού πιθανότητας, επιπέδου
6	**Γραμμή Εργαλείων**	Συντόμευση σε βασικά εργαλεία του κυρίως μενού.
7	**Αποθήκευση**	Αποθήκευση του έργου στο cloud (google drive, dropbox, onedrive).
8	**Ενέργειες**	Περιοχή καταχώρισης των ενεργειών του έργου και των ιδιοτήτων τους (π.χ. διάρκεια).
9	**Διάγραμμα**	Περιοχή απεικόνισης του διαγράμματος Gantt.

1.5. Διαδίκτυο και επικοινωνία

Το Διαδίκτυο έχει καθιερωθεί ως το βασικό κανάλι επικοινωνίας σχεδόν σε όλο τον κόσμο. Το ηλεκτρονικό ταχυδρομείο, τα άμεσα μηνύματα (IM = instant messaging),

η τηλεφωνία μέσω Διαδικτύου (VoIP), τα φόρουμ, οι επιγραμμικές συζητήσεις (online chat) και η κοινωνική δικτύωση αποτελούν μερικά από τα σημαντικά διαδικτυακά εργαλεία επικοινωνίας με φίλους και συνεργάτες. Η χρήση των σύγχρονων εργαλείων και μεθόδων επικοινωνίας μέσω του Internet, οδηγεί τόσο τον κόσμο της επιχειρήσεων όσο και την καθημερινότητα των ανθρώπων στη μοντέρνα εποχή των ψηφιακών επικοινωνιών.

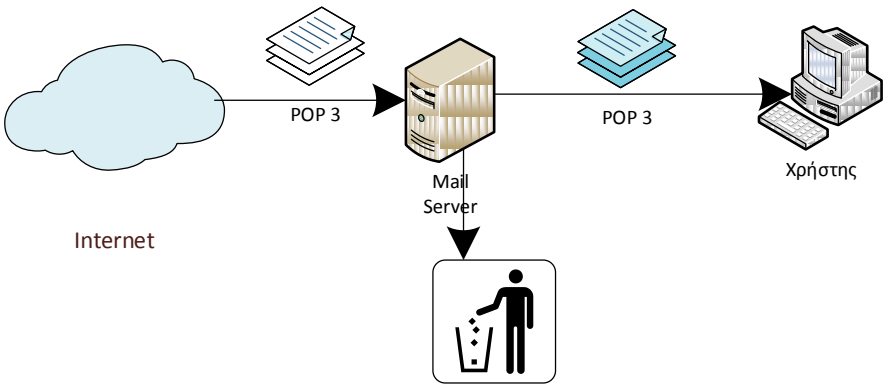
1.5.1. Ηλεκτρονικό Ταχυδρομείο

Η αρχή του Ηλεκτρονικού Ταχυδρομείου, μας οδηγεί πολύ πίσω όταν το 1962 το δίκτυο AUTODIN παρείχε δυνατότητες μηνυμάτων μεταξύ 1.350 τερματικών και διαχειρίζονταν 30 εκατομμύρια μηνύματα κάθε μήνα.

Έκτοτε το e-mail έχει μπει στην καθημερινότητα πολλών εκατομμυρίων χρηστών του διαδικτύου. Η αποστολή και λήψη του ταχυδρομείου γίνεται βάσει συγκεκριμένων πρωτοκόλλων.

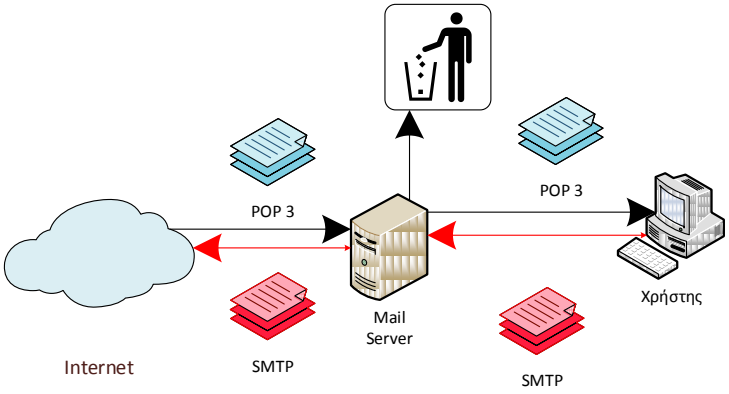
Το POP3 (Post Office Protocol), είναι ένα πρωτόκολλο το οποίο αποτελεί εξέλιξη των προηγούμενων μορφών του πρωτοκόλλου, τα οποία ονομαζόταν POP1 και POP2 (https://el.wikipedia.org). Σύμφωνα με αυτό, επιτρέπει στους χρήστες του διαδικτύου που έχουν προσωρινές συνδέσεις (πχ dial-up) να παραλαμβάνουν την ηλεκτρονική τους αλληλογραφία, να την αποθηκεύουν στον τοπικό σκληρό δίσκο και στην συνέχεια να την διαβάζουν χωρίς να χρειάζεται να παραμένουν συνδεδεμένοι στο Διαδίκτυο. Παρόλο που υπάρχει η δυνατότητα να παραμείνουν στον εξυπηρετητή του ηλεκτρονικού ταχυδρομείου, οι περισσότερες εφαρμογές που στηρίζονται στο POP3, λαμβάνουν όλα τα μηνύματα και τα σβήνουν από τον mailserver (Σχήμα 1.6).

Το POP3 χρησιμοποιεί την πόρτα 110 για να εγκαθιδρύσει μία σύνδεση TCP με τον mail server. Πολλά προγράμματα ηλεκτρονικού ταχυδρομείου χρησιμοποιούν κρυπτογράφηση ούτως ώστε τα δεδομένα που διακινούνται στην σύνδεση αυτή να μην είναι αναγνώσιμα από άλλους. Για να αποδεχθεί ο mail server την σύνδεση, θα πρέπει ο χρήστης να δώσει το όνομα χρήστη και τον κωδικό πρόσβασής του. Η αρχική έκδοση του POP3 μετέδιδε τα ευαίσθητα αυτά δεδομένα σε μορφή απλού κειμένου, οπότε οποιοσδήποτε μπορούσε να τα διαβάσει. Στην συνέχεια όμως το πρωτόκολλο βελτιώθηκε και πλέον παρέχει την δυνατότητα κρυπτογραφημένης μετάδοσης του ονόματος χρήστη και του κωδικού. Παρόλα αυτά όμως, πολλοί χρήστες δεν γνωρίζουν αυτήν την δυνατότητα και συνεπώς δεν την χρησιμοποιούν.



Σχήμα 1.6: *Λειτουργία του πρωτόκολλου POP3*

SMTP (Simple Mail Potocol). Ενώ το POP3 είναι ένα πρωτόκολλο λήψης της αλληλογραφίας, το SMTP είναι πρωτόκολλο αποστολής της αλληλογραφίας. Άρα όταν θέλουμε να στείλουμε ένα e-mail με πιθανά επισυναπτόμενα αρχεία, έχει την διεύθυνση στην οποία θέλουμε να σταλεί και το e-mail φεύγει από τον υπολογιστή μας και μεταφέρεται στον mail server του παρόχου μας για να προωθήσει το mail μας. Στο Σχήμα 1.7 βλέπουμε ο χρήστης να στέλνει ένα e-mail.



Σχήμα 1.7**:** *Λειτουργία του πρωτόκολλου SMTP*

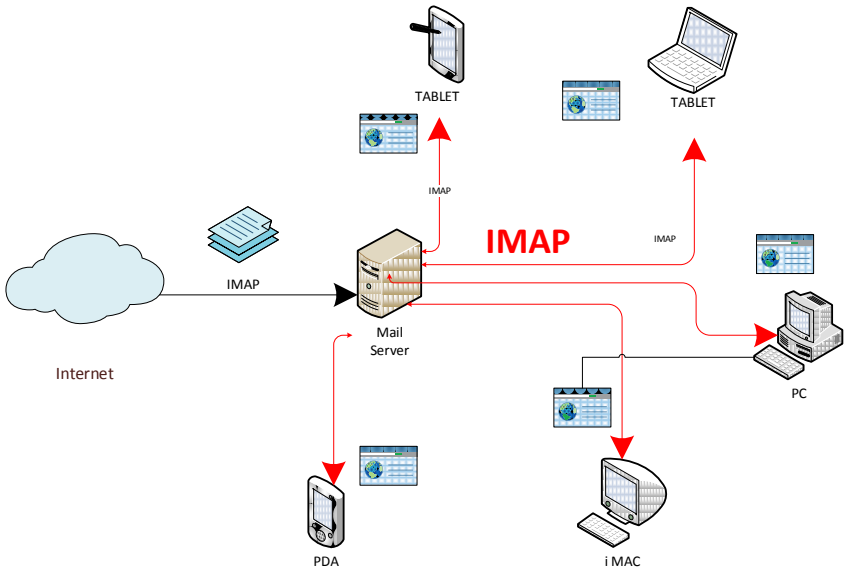
IMAP (Internet Message Access Protocol). Λόγω της διάδοσης έξυπνων συσκευών όπως κινητά, tablets, φορητοί υπολογιστές διαφόρων μεγεθών και δυνατοτήτων, καθώς και κατασκευής έξυπνων γυαλιών και ρολογιών, υπάρχει η ανάγκη για κάποιον χρήστη να μπορεί να διαβάζει τα e-mail του από διάφορες συσκευές, χωρίς να τα "κατεβάζει" σε μια συσκευή. Αν δηλαδή διαβάζει ένα e-mail με POP3 από το κινητό του, τότε το e-mail αυτό θα αποθηκευθεί στη μνήμη του κινητού και αν θελήσει να διαβάσει το ίδιο e-mail από τον υπολογιστή του σπιτιού του θα είναι αδύνατο γιατί το e-mail αυτό έχει μεταφερθεί από τον mail server στο κινητό και δεν υπάρχει παρά μόνον εκεί.

Με το πρωτόκολλο αυτό λοιπόν, αν ο χρήστης το ενεργοποιήσει στην εφαρμογή του ταχυδρομείου του και σε όσες συσκευές θέλει, μπορεί να συνδέεται με τον

εξυπηρετητή ταχυδρομείου του παρόχου του και όταν ζητήσει το ταχυδρομείο του, ο server του στέλνει μόνο την κεφαλίδα (Header) του mail, δηλαδή ποιος είναι ο αποστολέας, ποιο είναι το θέμα, και λίγες γραμμές από το σώμα του γράμματος. Το κυρίως e-mail παραμένει στον server και θα μπορεί να το ξαναδιαβάσει από άλλη συσκευή. Όταν κάποτε αποφασίσει μπορεί να "κατεβάσει" το e-mail σε κάποια συσκευή και να διαγραφεί πλέον από τον εξυπηρετητή.

Στο Σχήμα 1.8 βλέπουμε το ίδιο e-mail να μπορεί να κοινοποιηθεί σε πολλές συσκευές ταυτόχρονα και να παραμένει στον εξυπηρετητή.

Η έκδοση IMAP που χρησιμοποιείται σήμερα είναι η έκδοση 4, αναθεώρηση 1 (IMAP 4 rev 1) η οποία ορίζεται από το RFC 3501. Ένας διακομιστής IMAP στην πράξη δέχεται επικοινωνία από την port 143. Το IMAP όταν χρησιμοποιείται με Secure Sockets Layer, είναι γνωστό ως IMAPS και δέχεται επικοινωνία από την port 993 (https://el.wikipedia.org).



Σχήμα 1.8: *Λειτουργία του πρωτόκολλου IMAP*

Υπάρχουν και άλλα πρωτόκολλα ταχυδρομείου, τα οποία όμως χρησιμοποιούνται από κλειστά περιβάλλοντα και από συγκεκριμένα λογισμικά, όπως το MAPI (Messaging Application Programming Interface), της Microsoft. Για παράδειγμα το συγκεκριμένο μπορεί να χρησιμοποιηθεί από το Microsoft Outlook για την επικοινωνία του με το Microsoft Exchange.

1.5.2. Τηλεφωνία μέσω Διαδικτύου.

Η πραγματοποίηση κλήσεων ομιλίας μέσω Διαδικτύου (VoIP), η σωστότερα "φωνή επί διαδικτυακού πρωτοκόλλου", είναι πλέον καθημερινότητα. Ο συγκεκριμένος τρόπος επικοινωνίας χαρακτηρίζει μια ομάδα πρωτοκόλλων – τεχνολογιών (H. 323, SIP), η οποία προσφέρει επιγραμμική ομιλία (online) με αρκετά καλή ποιότητα και με πολύ μικρό κόστος. Μέχρι πριν λίγα χρόνια, οι συνομιλίες αυτές για να γίνουν έπρεπε

ο χρήστης να έχει Η/Υ, με μικρόφωνο, ακουστικά και κατάλληλο λογισμικό για να προωθεί την φωνή στο Διαδίκτυο. Η κλήση αυτή κατέληγε σε ένα άλλο ανάλογα εξοπλισμένο PC, χωρίς να υπάρχει κάποια χρέωση εκτός από την χρέωση για τον πάροχο του Internet μιας και δεν μεσολαβούσε εταιρία τηλεπικοινωνιών (π.χ. ΟΤΕ) (Σχήμα 1.9).

Τελευταία έχουν δημιουργηθεί εταιρίες οι οποίες με τις υπηρεσίες τους γεφυρώνουν χρήστες που κάνουν VoIP κλήσεις και απευθύνονται σε τηλέφωνα σταθερής τηλεφωνίας.

Στο σχήμα 1.9 μπορούμε να δούμε πως η επικοινωνία γίνεται χωρίς την μεσολάβηση υποδομής σταθερής τηλεφωνίας αλλά μόνο Διαδικτυακή.

Επειδή όλο και περισσότεροι χρήστες έχουν ήδη τον αναγκαίο εξοπλισμό στα σπίτια τους ή στις εταιρίες τους (DSL ή VDSL Routers), η υιοθέτηση της τηλεφωνίας αυτής είναι εύκολη.

Βέβαια να αναφέρουμε και τα **προβλήματα** από τον συγκεκριμένο τρόπο τηλεφωνίας. Κάποια από αυτά είναι :

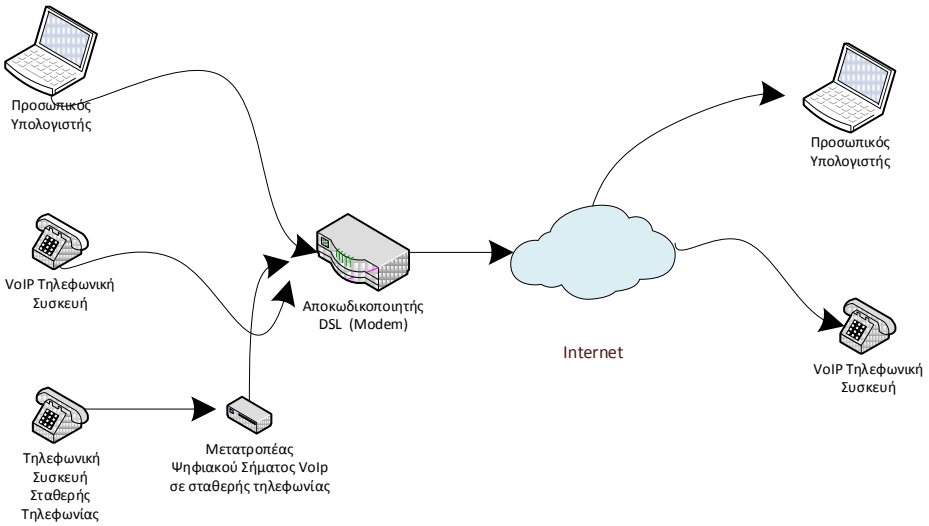
- Αν ο υπολογιστής δεν λειτουργεί ή το Διαδίκτυο διακοπεί, η τηλεφωνία διακόπτεται.

- Δεν είναι εύκολος ο εντοπισμός του καλούντος, γιατί η ταυτότητά του (IP) περνά μέσα το Internet. Αν χρειαστεί αναγνώριση του καλούντος από την Πυροσβεστική ή άλλη υπηρεσία άμεσης επέμβασης, δεν είναι εύκολο.

- Συσκευές όπως τηλεομοιοτυπικές συσκευές (Fax), καλωδιακή τηλεόραση κλπ, δεν είναι σίγουρη η λειτουργία τους.

- Η παρακολούθηση συσκευών VoIP για λόγους ασφαλείας δεν είναι εφικτή.

- Τέλος και πιο σημαντικό, η ποιότητα του ήχου είναι κατώτερη. Και αυτό συμβαίνει γιατί η μετάδοση της φωνής γίνεται μέσα από μεταγωγή πακέτων τα οποία ταξιδεύουν ασύγχρονα σε όλο το Διαδίκτυο. Όταν φτάσουν λοιπόν στον προορισμό τους και πρέπει να γίνει η επανένωση τους, μπορεί να έχουμε φαινόμενα απώλειας κάποιων πακέτων ή καθυστέρηση αφίξεων κάποιων από αυτά.



Σχήμα 1.9: *Λειτουργία συνδεσμολογίας VoIP*

1.5.3. Βιοντεοκλήσεις μέσω Διαδικτύου

Η Επικοινωνία αυτή μπορεί να γίνει με αποστολή εικόνας μεταξύ δύο ή περισσότερων χρηστών. Με την εξάπλωση του φθηνού γρήγορου internet μετά το 1990, την κατασκευή γρήγορων επεξεργαστών και αποτελεσματικών τεχνικών συμπίεσης εικόνας, η επικοινωνία με βίντεο έχει εξαπλωθεί στις επιχειρήσεις, στην εκπαίδευση, στην ιατρική και στα μέσα μαζικής επικοινωνίας.

Η NASA στη δεκαετία του 60 επιχείρησε επικοινωνία με εικόνα με τις επανδρωμένες αποστολές της μέσω VHF και UHF αναλογικών σημάτων, όπως επίσης και αργότερα η επικοινωνία δημοσιογράφων με τους τηλεοπτικούς σταθμούς γίνονταν μέσω δορυφορικού σήματος. Όλες αυτές οι επικοινωνίες όμως ήταν πολύ ακριβές για πιο κοινούς σκοπούς όπως στην εκπαίδευση, στις επιχειρήσεις στην ιατρική κ.λπ.

Από το 1980 όμως με την ψηφιακή τηλεφωνία, και τα δίκτυα τύπου ISDN, εξασφαλίστηκε μια μίνιμουμ ταχύτητα επικοινωνίας (συνήθως 128 kilobits/s) η οποία μπορεί να μεταφέρει συμπιεσμένο βίντεο και ήχο. Τελικά, στη δεκαετία του 1990, η βίντεο-επικοινωνία βασισμένη στο Internet Protocol (IP), έγινε δυνατή και αναπτύχθηκε επιτρέποντας προσωπικούς υπολογιστές να μεταγάγουν πακέτα πληροφοριών στο Διαδίκτυο με συμπιεσμένη εικόνα και ήχο (https://en.wikipedia.org/wiki/Videoconferencing).

Υπάρχουν δύο είδη συστημάτων επικοινωνίας με βίντεο :

- **Εξειδικευμένα Συστήματα** ή Ειδικά Συστήματα (dedicated systems) τα οποία είναι συσκευές οι οποίες έχουν όλα τα απαραίτητα εξαρτήματα και μηχανισμούς σε μια κατασκευή. Αυτή είναι συνήθως μια κονσόλα με μια τηλεχειριζόμενη κάμερα υψηλής ανάλυσης. Η κάμερα είναι PTZ, δηλαδή με

δυνατότητα ελεύθερης οριζόντιας κίνησης (Pan), ελεύθερης κάθετης κίνησης (Tilt) και με δυνατότητα δυναμικής εστίασης (Zoom).

- **Επιτραπέζια Συστήματα (Desktop).** Στην κατηγορία αυτή το σύστημα αποτελείται από ένα απλό PC, με επιπρόσθετες κάρτες επέκτασης που του παρέχουν τους απαραίτητους κωδικοποιητές και διεπαφές. Τα συστήματα αυτά συνήθως χρησιμοποιούν το H. 323 πρότυπο κωδικοποίησης σήματος.

1.5.4. Επικοινωνία με Βίντεο στο Υπολογιστικό Σύννεφο

Η επικοινωνία με βίντεο στο σύννεφο δεν απαιτεί τον εξοπλισμό των παραπάνω συστημάτων. Προορίζεται για μικρομεσαίες επιχειρήσεις ή για μεγάλες πολυεθνικές εταιρίες κοινωνικής δικτύωσης. Τέτοια συστήματα (Cloud Based) μπορούν να υποστηρίξουν 2D ή 3D εκπομπή βίντεο. Επίσης υποστηρίζουν επικοινωνία βίντεο με έξυπνες συσκευές VoIP παρέχοντας και δυνατότητες εγγραφής για αρχειοθέτηση των προβαλλόμενων βίντεο.

1.5.5. Μεταφορά Αρχείων και Περιεχομένου

Το Πρωτόκολλο Μεταφοράς Αρχείων (**File Transfer Protocol (FTP))** είναι ένα ευρέως χρησιμοποιούμενο πρωτόκολλο. Η απλή μεταφορά αρχείων από υπολογιστή σε υπολογιστή μέσω Διαδικτύου, αποτέλεσε μια από τις βασικές υπηρεσίες του από το ξεκίνημά του. Στηρίζεται στο πρωτόκολλο επικοινωνίας TCP / IP. Ο υπολογιστής που τρέχει εφαρμογή FTP client μόλις συνδεθεί με τον εξυπηρετητή (server) μπορεί να εκτελέσει ένα πλήθος διεργασιών όπως ανέβασμα αρχείων στον server, κατέβασμα αρχείων από τον server, μετονομασία ή διαγραφή αρχείων από τον server κ.ο.κ.

Ο τρόπος λειτουργίας του είναι σχετικά απλός. Αρχικά ο FTP server ανοίγει την θύρα (port) 21 περιμένοντας έναν FTP client να συνδεθεί. Στη συνέχεια ο client ξεκινά μια νέα σύνδεση από μια τυχαία θύρα προς την θύρα 21 του server. Μόλις γίνει η σύνδεση παραμένει ανοιχτή για όλη τη διάρκεια της συνόδου FTP. Η συγκεκριμένη σύνδεση ονομάζεται σύνδεση ελέγχου (control connection).

Σχετικά με τη δημιουργία της σύνδεσης δεδομένων (οι τρόποι για να δημιουργηθεί, με χρήση της ενεργητικής λειτουργίας (active mode) ή με χρήση της παθητικής λειτουργίας (passive mode).

Ενεργητική Λειτουργία (Active Mode)

Στην ενεργητική λειτουργία (active mode) ο FTP client διαλέγει μια τυχαία θύρα στην οποία δέχεται τα δεδομένα της σύνδεσης. Ο client στέλνει τον αριθμό της θύρας παραλαβής, στην οποία επιθυμεί να "ακούει" (listen) για εισερχόμενες συνδέσεις. Ο FTP server δημιουργεί μια σύνδεση από την θύρα 20 στην ανοιχτή θύρα του client για τη μεταφορά των δεδομένων.

Οποιαδήποτε πληροφορία ζητήσει ο client, ανταλλάσσεται με βάση αυτή τη σύνδεση, που βασίζεται στο TCP. Όταν η μεταφορά ολοκληρωθεί ο server κλείνει τη σύνδεση αποστέλλοντας ένα πακέτο FIN, όπως σε κάθε σύνδεση βασισμένη στο TCP. Κάθε φορά που ο client ζητάει δεδομένα, δημιουργείται κατά παρόμοιο τρόπο μια σύνδεση δεδομένων και η διαδικασία επαναλαμβάνεται.

Παθητική Λειτουργία (ActiveMode)

Στην παθητική λειτουργία (passive mode) ο client ζητά από τον server να διαλέξει μια τυχαία θύρα του, στην οποία θα "ακούει" (listen) για την σύνδεση δεδομένων (data connection). Ο server ενημερώνει τον client για την θύρα την οποία έχει διαλέξει και ο client συνδέεται σε αυτή για τη μεταφορά των δεδομένων. Η μεταφορά ολοκληρώνεται όπως και στην ενεργητική λειτουργία (active mode), αφού η σύνδεση δεδομένων βασίζεται στο TCP.

Ανώνυμη Σύνδεση

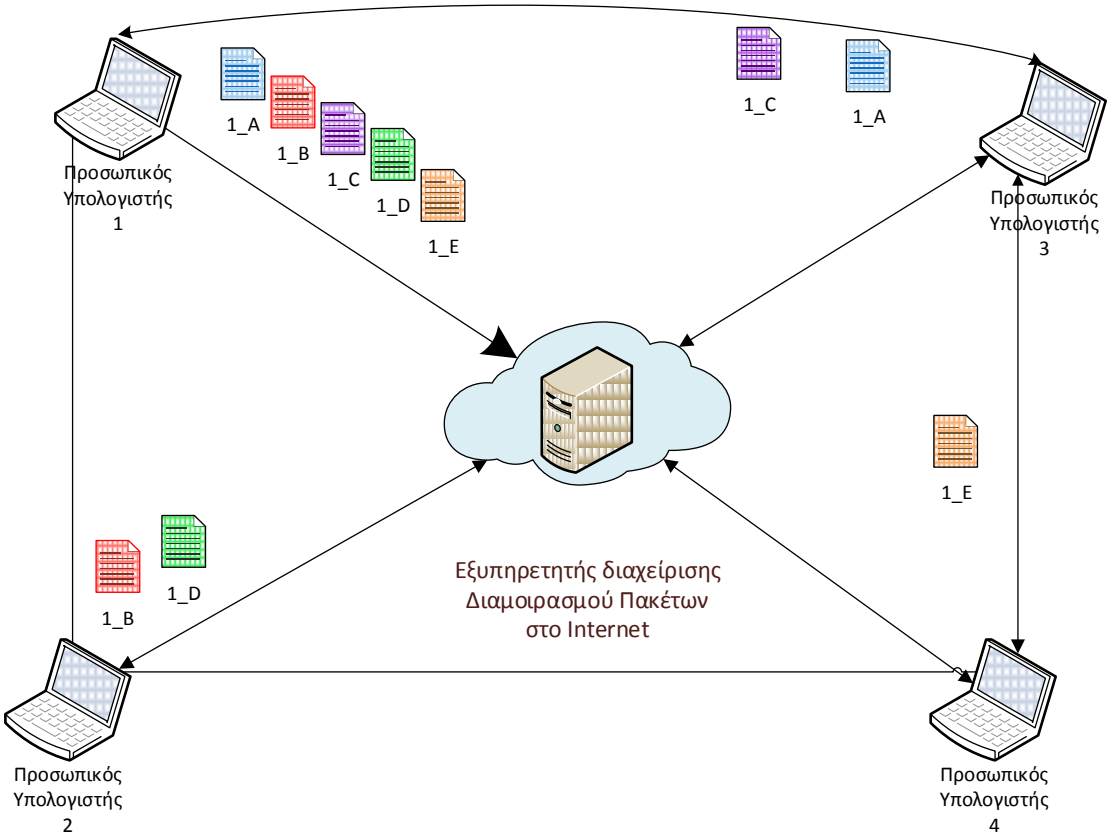
Στο πρωτόκολλο επικοινωνίας FTP, έχει επικρατήσει και η δυνατότητα σύνδεσης στον FTP server, χωρίς καταχωρημένο όνομα χρήστη και κωδικό για τον χρήστη. Υπάρχει δηλαδή λογαριασμός στον οποίο μπορεί να συνδεθεί κάποιος χρήστης με όνομα **Anonymous**, χωρίς κωδικό και να γίνει δεκτός στο σύστημα. Η "ελεύθερη" αυτή σύνδεση χρησιμοποιείται για αρχεία που είναι ανοιχτά στο κοινό, σαν αποθήκη πληροφοριών.

Βέβαια ο βαθμός πρόσβασης των χρηστών αυτών είναι περιορισμένος. Δεν μπορούν δηλαδή να έχουν πρόσβαση σε όλους τους φακέλους του server, ούτε έχουν δικαιώματα διαγραφής ή μεταβολής των ιδιοτήτων και πληροφοριών των αρχείων.

Με την εξέλιξη των εφαρμογών του Διαδικτύου, η μεταφορά αρχείων και περιεχομένου μπορεί να γίνει ως έναν βαθμό και από περιβάλλοντα κοινωνικής δικτύωσης, ανταλλαγής μηνυμάτων, ηλεκτρονικού ταχυδρομείου κ.λπ.

1.5.6. Επικοινωνία σε Ομότιμα Δίκτυα (Peer to Peer)

Η επικοινωνία μεταξύ ομότιμων χρηστών στο Διαδίκτυο εστιάζει κυρίως στην ανταλλαγή και διαμοιρασμό αρχείων μεταξύ τους. Ο τρόπος αυτός μεταφοράς αρχείων έχει εξελιχθεί μέσα από διάφορα στάδια, ξεκινώντας από δίκτυα όπως το Napster, για μοίρασμα αρχείων ήχου και τραγουδιών, κάνοντας την τεχνολογία αυτή γνωστή στο ευρύ Διαδικτυακό κοινό. Η εξέλιξη έφερε το πρωτόκολλο Bit Torrent, το οποίο αρχικά χρησιμοποιήθηκε για τον διαμοιρασμό του λειτουργικού συστήματος Linux, αλλά και στη συνέχεια για διανομή ταινιών, μουσικής και κάθε είδους αρχεία. Σήμερα αποτελεί από τους βασικούς τρόπους διανομής δεδομένων στο Διαδίκτυο (Σχήμα 1.10).



Σχήμα 1.10: *Επικοινωνία Ομότιμων Δικτύων*

Στο Σχήμα 1.10 ο χρήστης 1 ανεβάζει ένα αρχείο στο Διαδίκτυο και τμήματα του αρχείου αυτού μεταφέρονται στον χρήστη 2, στον χρήστη 3 και στον χρήστη 4. Ο χρήστης 2 δέχεται τμήματα του αρχείου που διαθέτει ο 1 αλλά παράλληλα διαθέτει τμήματα του ίδιου αρχείου στον χρήστη 3 και στον χρήστη 4. Ο χρήστης 3 δέχεται τμήματα του αρχείου και από τον χρήστη 1 αλλά και από τον 2 και 4. Τελικά εκτός από τον υπολογιστή 1 που είναι η πηγή του αρχικού αρχείου, όλοι οι υπόλοιποι ενδιαφερόμενοι μοιράζονται πακέτα μεταξύ τους μέχρι όλοι να έχουν όλα τα κομμάτια. Μπορούμε να φανταστούμε όταν οι υπολογιστές που βοηθούν στο διαμοιρασμό είναι χιλιάδες (https://en.wikipedia.org/wiki/Peer-to-peer_file_sharing).

Αν ο χρήστης 1 ήταν η μοναδική πηγή για την μεταφορά του αρχείου και προς τους υπόλοιπους , ίσως να μην μπορούσε να εξυπηρετήσει ταυτόχρονα και τους τρείς. Τώρα όμως όλοι αναλαμβάνουν να τροφοδοτούν όλους, το βάρος που έχει ο χρήστης – υπολογιστής 1 μειώνεται δραστικά.

Εδώ πρέπει να σημειώσουμε ότι όλη την διαχείριση του ποιός στέλνει σε ποιόν και τι, το κάνει ένας εξυπηρετητής ή πολλοί εξυπηρετητές σε ένα σημείο ή σε διάφορα γεωγραφικά σημεία του πλανήτη. Αυτή είναι η κεντρική ιδέα του πρωτοκόλλου αυτού.

1.5.7. Έλεγχος από απόσταση (Remote Desktop Software)

Στην περίπτωση αυτή μιλάμε για ένα χαρακτηριστικό του λειτουργικού συστήματος το οποίο επιτρέπει έναν προσωπικό συνήθως υπολογιστή να παραχωρήσει το περιβάλλον εργασίας του σε έναν άλλο υπολογιστή από μικρή ή μεγάλη απόσταση. Η παραχώρηση αυτή μπορεί να είναι μόνο για απλή εμφάνιση, αλλά μπορεί να συνοδευτεί και από εκχώρηση του ελέγχου του υπολογιστή. Έτσι ο μακρινός υπολογιστής μπορεί να έχει τον πλήρη έλεγχο του υπολογιστή που του επιτρέπει την είσοδο.

Συνήθως ο υπολογιστής που "δίνει" το περιβάλλον λειτουργίας του γίνεται τερματικό (Dummy Terminal) αυτού που παίρνει τον έλεγχο και δεν έχει καμία δυνατότητα λειτουργίας. Ο έλεγχος του κεντρικού συστήματος μπορεί να μεταφερθεί ακόμη και σε έξυπνες συσκευές. Δημοφιλή λογισμικά για την λειτουργία αυτή είναι το Team Viewer, το Remote Desktop Manager και πολλά άλλα.

1.5.8. Επικοινωνία μέσα από 3D Εικονικούς Κόσμους (3D Virtual Worlds)

Η επικοινωνία αυτή είναι ιδιαίτερα αγαπητή στους νέους, πολλά εκατομμύρια των οποίων συνδέονται καθημερινά στους κόσμους αυτούς κυρίως γα να παίξουν. Παράλληλα όμως με την διασκέδαση οι νέοι γνωρίζουν και τα μυστικά του τρόπου αυτού επικοινωνίας.

Ο χρήστης αποκτά πρόσβαση σε εικονικούς κόσμους οι οποίοι είναι εικονικές προσομοιώσεις από υπολογιστή ή κατανεμημένο δίκτυο υπολογιστών στο Διαδίκτυο. Ο χρήστης συνήθως αποκτά κάποιο 3D μοντέλο (avatar), το οποίο αντιπροσωπεύει την παρουσία του στον κόσμο αυτό. Οι κόσμοι αυτοί παρέχουν ρεαλιστικά χαρακτηριστικά και φυσικούς νόμους όπως βαρύτητα, κίνηση, αλληλεπίδραση μεταξύ των φυσικών σωμάτων, επιγραμμική επικοινωνία (online) με ήχο και φωνή (real time voice communication) και γενικά αίσθηση πραγματικότητας.

Σχήμα 1.11.: *Σχολική τάξη στον 3D εικονικό κυβερνοχώρο9 του Second Life (λήψη από το περιβάλλον του second life)*

Οι εικονικοί αυτοί κυβερνοχώροι εμπεριέχουν και έννοιες όπως είναι η εικονική οικονομία, μιας και στους κόσμους αυτούς υπάρχουν συναλλαγές με κάποιο είδος νομίσματος, η γεωγραφία η οποία είναι βασικό στοιχείο της κατασκευής των κόσμων αυτών καθώς και η κοινωνική δικτύωση μιας και οι χρήστες συμμετέχουν σε κοινωνικές δραστηριότητες και συζητήσεις.

Στο Σχήμα 1.11 βλέπουμε μια εικονική σχολική τάξη, στην οποία μπορούν να συμμετέχουν online χρήστες του περιβάλλοντος οι οποίοι μπορούν να παρακολουθήσουν το μάθημα, δίνοντας έτσι την ευκαιρία, σε μαθητές από δυσπρόσιτες περιοχές να έχουν εκπαίδευση με σύγχρονη επικοινωνία (real time), να εκφράσουν τις απορίες τους, να τους απαντήσουν οι καθηγητές που είναι συνδεδεμένοι, αλλά και να συνεργαστούν με άλλους μαθητές για κάποια κοινή δραστηριότητα.

1.5.9. Πρωτόκολλα Επικοινωνίας στο Διαδίκτυο

Πρωτόκολλο επικοινωνίας μεταξύ δυο υπολογιστών είναι ένα σύνολο από κανόνες βάσει των οποίων ανταλλάσουν πληροφορίες. Είναι με άλλα λόγια, ένα πακέτο από κανόνες στους οποίους συνήθως στηρίζεται η επικοινωνία τους. Αυτοί καθορίζουν την μορφή των δεδομένων, τον χρόνο μετάδοσης των δεδομένων και την σειρά με την οποία στέλνονται. Τα χαρακτηριστικά των πρωτοκόλλων ορίζονται επακριβώς από διεθνείς οργανισμούς και γίνονται αποδεκτά από όλους.

Τα περισσότερο διαδεδομένα πρωτόκολλα για την επικοινωνία υπολογιστών στο διαδίκτυο είναι το TCP/IP, το NETBEUI και το IPX/SPX, με το TCP/IP να έχει επικρατήσει ως το πιο διαδεδομένο από όλα.

1.5.10. Δομή Πρωτόκολλου Ελέγχου Μεταφοράς (TCP/IP)

Κάθε κατανεμημένη εφαρμογή, όπως είναι η μεταφορά δεδομένων (FTP), το ηλεκτρονικό ταχυδρομείο (E-mail) και άλλες, απαιτούν ένα σύνολο πολύπλοκων λειτουργιών για να εκτελεστούν. Στηρίζονται σε μια τμηματική αρχιτεκτονική στην οποία τα διάφορα τμήματα έχουν συγκεκριμένη αποστολή.

Το Πρωτόκολλου Ελέγχου Μεταφοράς δεδομένων (ΠΕΜ), στηρίζεται σε 3 βασικούς πυλώνες. Την διαδικασία (process), τους υπολογιστές που επικοινωνούν (hosts) και τα δίκτυα (networks) στα οποία συνδέονται οι υπολογιστές.

Οι διαδικασίες είναι βασικές οντότητες επεξεργασίας οι οποίες εκτελούνται στους υπολογιστές και επικοινωνούν μεταξύ τους. Για παράδειγμα, η μεταφορά αρχείων από έναν υπολογιστή σε έναν άλλο είναι μια τέτοια διαδικασία. Η μια διαδικασία στον υπολογιστή Α στέλνει τα δεδομένα και η άλλη διαδικασία στον υπολογιστή Β τα δέχεται. Ένα άλλο παράδειγμα αποτελεί η σύνδεση σε ένα ιστότοπο κοινωνικής δικτύωσης (log in). Στον υπολογιστή μας πληκτρολογούμε τον κωδικό και το όνομα χρήστη. Η διαδικασία στον υπολογιστή μας τα στέλνει στον απομακρυσμένο

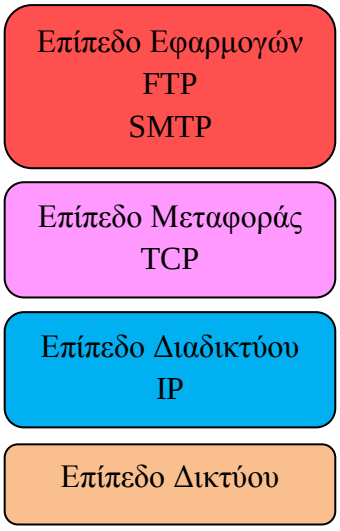
υπολογιστή ενεργοποιώντας εκείνη την διαδικασία η οποία ελέγχει τα στοιχεία και μας παρέχει πρόσβαση στο προφίλ μας.

Οι διαδικασίες εκτελούνται σε υπολογιστές, οι οποίοι μπορεί να εκτελούν ταυτόχρονα πολλές τέτοιες διαδικασίες, όπως είναι για παράδειγμα η χρησιμοποίηση του ηλεκτρονικού ταχυδρομείο, η μεταφορά αρχείων και η επικοινωνία με κάποια σελίδα κοινωνικής δικτύωσης. Όλες αυτές οι διαδικασίες εκτελούνται ταυτόχρονα, χωρίς να επηρεάζει η μια την άλλη.

Τέλος, τα δίκτυα είναι η υποδομή στην οποία συνδέονται όλοι οι υπολογιστές παρέχοντας τα μονοπάτια (paths) επικοινωνίας. Συνεπώς, το δίκτυο επικοινωνίας έχει σαν αποστολή να διαβιβάζει τα δεδομένα από ένα Η/Υ σε έναν άλλο, ενώ οι Η/Υ έχουν σαν αποστολή να κατευθύνουν δεδομένα από την διαδικασία που βρίσκεται στον αποστολέα υπολογιστή στην αντίστοιχη διαδικασία που βρίσκεται στον παραλήπτη υπολογιστή.

Μετά από αυτά μπορούμε να οργανώσουμε την επικοινωνία δύο υπολογιστών σε τέσσερα σχετικά ανεξάρτητα επίπεδα (Σχήμα 1.12):

1. **Επίπεδο Εφαρμογών** (Application Layer)
2. **Επίπεδο Μεταφοράς** (Transport Layer)
3. **Επίπεδο Διαδικτύου** (Internet Layer)
4. **Επίπεδο Δικτύου** (Network or Link Layer)



Σχήμα 1.12.: *Επίπεδα του Πρωτόκολλου Ελέγχου Μεταφοράς*

1.5.11. Λειτουργία

Το σχήμα 1.13 μας δείχνει τον τρόπο με τον οποίο τα πρωτόκολλα και τα επίπεδα λειτουργούν. Η πύλη είναι συσκευή ή υπολογιστής η οποία λειτουργεί στο Επίπεδο

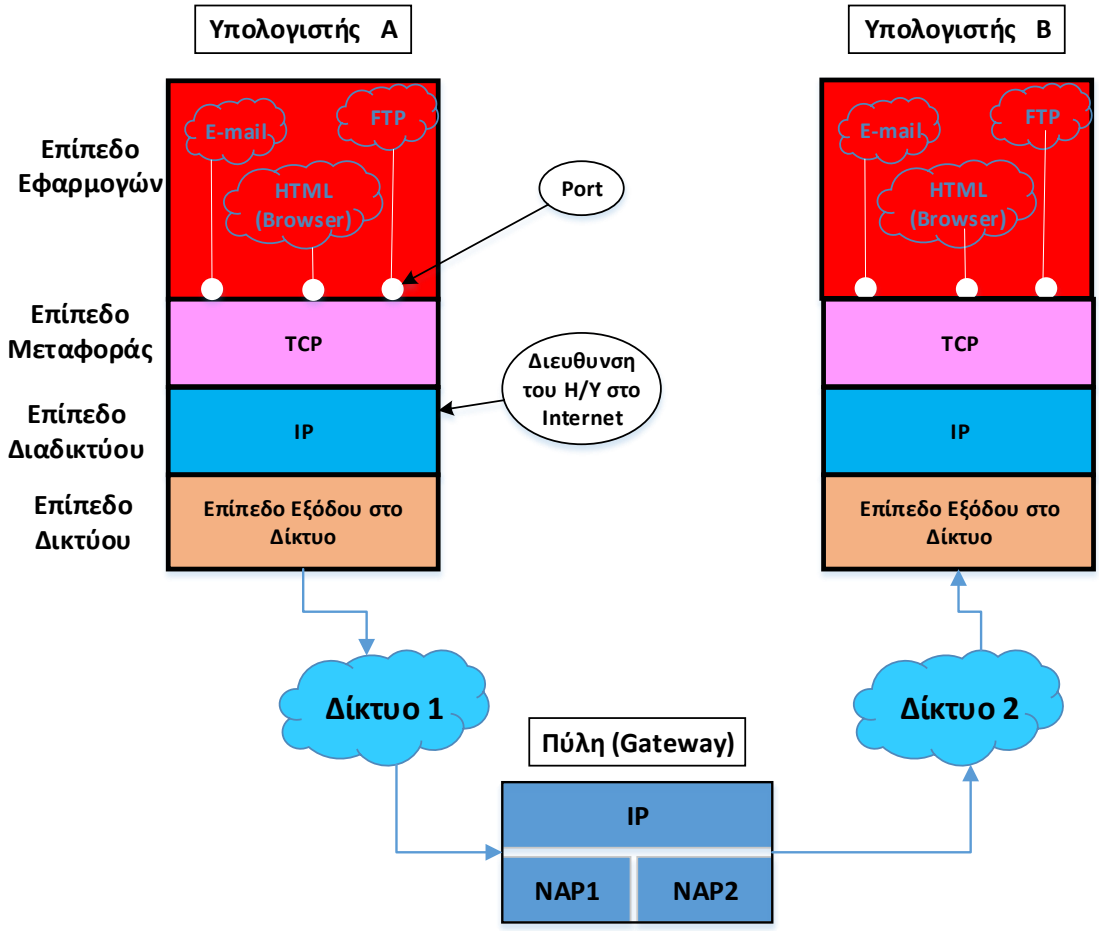
Δικτύου και "γεφυρώνει την επικοινωνία" μεταξύ των δύο δικτύων στα οποία βρίσκονται οι υπολογιστές που θέλουν να επικοινωνήσουν. Επίσης χρησιμοποιεί και το Επίπεδο του Διαδικτύου το οποίο παρέχει την διεύθυνση του παραλήπτη για να μπορέσει η πύλη να κατευθύνει τα κομμάτια των δεδομένων προς τον προορισμό τους.

Στην αρχιτεκτονική αυτή υπάρχουν δύο επίπεδα διευθύνσεων. Το ένα είναι η διεύθυνση IP η οποία είναι μοναδική για κάθε υπολογιστή που συνδέεται στο Internet και αποτελεί την μοναδική του ταυτότητα. Το δεύτερο αφορά τον υπολογιστή όπου υπάρχουν διαδικασίες που επιζητούν επικοινωνία με άλλες διαδικασίες υπολογιστών που βρίσκονται μακριά. Πως λοιπόν ο υπολογιστής θα ξέρει τα δεδομένα που έρχονται από έξω σε ποια διαδικασία πρέπει να οδηγηθούν; Το πρόβλημα αυτό λύνεται με έναν μοναδικό (σε κάθε υπολογιστή) αριθμό ο οποίος ονομάζεται θύρα ή πόρτα. Όταν λοιπόν αρχίζει μια διαδικασία, ο Η/Υ της αποδίδει τον αριθμό μιας θύρας και έτσι την ξεχωρίζει από τις άλλες. Το SMTP πρωτόκολλο για την αποστολή αλληλογραφίας "ακούει" στο port 25 για τις επικοινωνίες του, το POP για την λήψη στο 110 κλπ.

Ας δούμε λοιπόν ένα παράδειγμα επικοινωνίας. Έστω ότι μια διαδικασία βρίσκεται στον Υπολογιστή Α του σχήματος 1.13 και έχει την θύρα 1 και θέλει να στείλει ένα μήνυμα σε μια διαδικασία στον Υπολογιστή Β στη θύρα 2. Το Επίπεδο Εφαρμογών δίνει το μήνυμα στο Επίπεδο Μεταφοράς με οδηγίες να το μεταφέρει στον Υπολογιστή Β στη θύρα 2. Στη συνέχεια το Επίπεδο Μεταφοράς δίνει το μήνυμα στο Επίπεδο Διαδικτύου με οδηγίες να το στείλει στον Υπολογιστή Β. Εδώ το επίπεδο αυτό δεν χρειάζεται να γνωρίζει τον αριθμό της θύρας. Το μόνο που χρειάζεται είναι ότι προορίζεται για τον Υπολογιστή Β. Τέλος το Επίπεδο Διαδικτύου δίνει το μήνυμα στο Επίπεδο Δικτύου με οδηγίες να το στείλει στην Πύλη (Gateway).

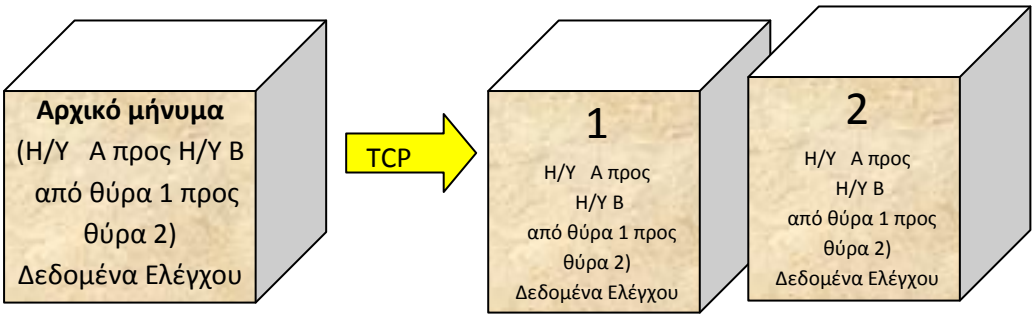
Για να ελέγχεται η επικοινωνία, μαζί με τα δεδομένα πρέπει να στέλνονται και δεδομένα ελέγχου. Το μήνυμα που δημιουργεί η διαδικασία είναι συνήθως ένα μεγάλο κομμάτι δεδομένων. Το πρωτόκολλο TCP τεμαχίζει το κομμάτι αυτό και δημιουργεί μικρότερα. Στο κάθε ένα από αυτά βάζει μεταξύ άλλων μια "ετικέτα" με τον αύξοντα αριθμό του κομματιού, τους αριθμούς των θυρών που εμπλέκονται και τον κώδικα ελέγχου (Checksum), ο οποίος ελέγχει αν το κομμάτι παραλήφθηκε επιτυχώς (βλέπε Σχήμα 1.14).

Στη συνέχεια τα τμήματα (segments) πηγαίνουν στο επίπεδο Διαδικτύου, το οποίο προσθέτει το δεδομενόγραμμα (IP Datagram) σε κάθε τμήμα της αποστολής. Τέλος το Επίπεδο Διαδικτύου (Internet) μεταβιβάζει τα τμήματα με το IP Datagram στο Επίπεδο Δικτύου το οποίο σε κάθε τμήμα προσθέτει την διεύθυνση του υποδικτύου (Subnetwork address) του παραλήπτη για να είναι γνωστό σε ποιά συσκευή του προοριζόμενου δικτύου θα παραδοθεί το τμήμα των δεδομένων.

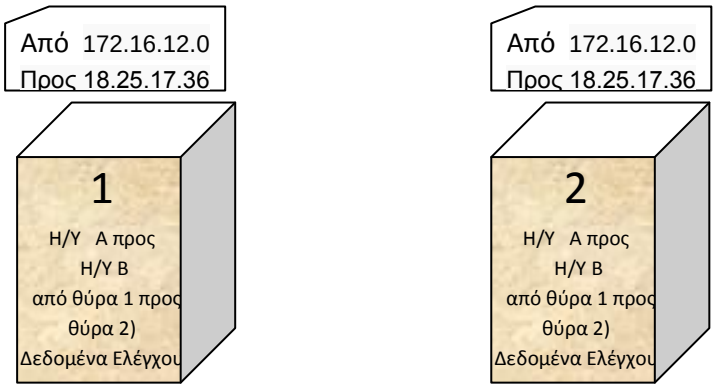


Σχήμα 1.13.: *Επικοινωνία με το TCP/IP.*

Επίσης προστίθενται πληροφορίες για χρήση συγκεκριμένων χαρακτηριστικών που αφορούν την διαδρομή του τμήματος όπως η προτεραιότητα (Priority) κλπ.

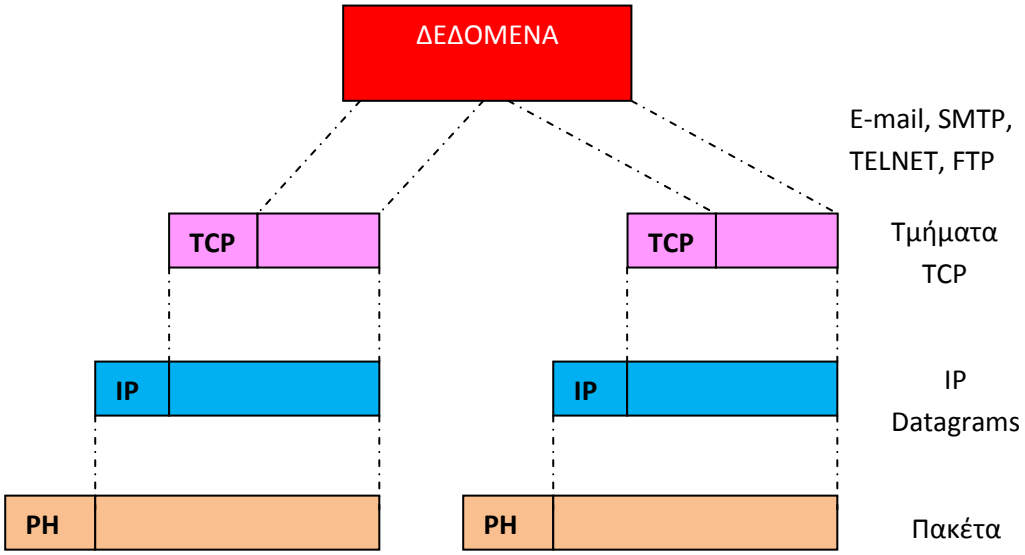


Σχήμα 1.14.: *Επεξεργασία του αρχικού μηνύματος από το πρωτόκολλο TCP/IP.*



Σχήμα 1.15.: *Το Επίπεδο Διαδικτύου μεταξύ άλλων προθέτει την IP διεύθυνση του αποστολέα και του παραλήπτη.*

Η "συσκευασία" η οποία περιέχει το τελικό σύνολο των πληροφοριών ονομάζεται πακέτο (Packet) και αποτελείται από πληροφορίες που είναι απαραίτητες για τα δεδομένα της αρχικής διαδικασίας ώστε να ταξιδέψουν μέχρι την άλλη άκρη της γης μέχρι να φτάσουν στο προορισμό τους και να αποσταλεί πίσω η "Απόδειξη Παραλαβής" από τον παραλήπτη. Και όλα αυτά σε ελάχιστο χρόνο. Στο σχήμα 1.16 βλέπουμε μια σχηματική απεικόνιση των παραπάνω, πως δηλαδή κάθε επίπεδο "στοιβάζει" το δικό του τμήμα πάνω στα προηγούμενα.



Σχήμα 1.16. *Δεδομένα χρήστη και πληροφορίες ελέγχου πρωτοκόλλων*

1.5.12. Μηχανισμοί Πρωτοκόλλων

Το κύριο χαρακτηριστικό όλων των αρχιτεκτονικών επικοινωνίας είναι ότι ένα ή περισσότερα πρωτόκολλα λειτουργούν σε κάθε επίπεδο της αρχιτεκτονικής και ότι δύο αντίστοιχα πρωτόκολλα στο ίδιο επίπεδο αλλά σε διαφορετικά συστήματα,

συνεργάζονται για να επιτύχουν την λειτουργία της επικοινωνίας. Οι μηχανισμοί οι οποίοι είναι κοινοί σε κάθε πρωτόκολλο είναι οι εξής :

- **Κατάτμηση και αναδόμηση.** Ο τεμαχισμός των δεδομένων σε μικρότερα κομμάτια έχει πολλά πλεονεκτήματα. Τα δίκτυα δέχονται να διακινήσουν κομμάτια πληροφορίας μικρού μεγέθους και όχι πάνω από 64K (65535 bytes). Ο έλεγχος λαθών είναι ευκολότερος, δεν δεσμεύονται πόροι για μεγάλα χρονικά διαστήματα και η μεταφορά είναι γρηγορότερη λόγω της δυναμικής διασποράς των πακέτων από διαφορετικούς εναλλακτικούς δρόμους.

- **Ενθυλάκωση.** Κάθε πακέτο ή PDU (Packet Data Unit) αποτελείται από στρώσεις ενθυλακωμένων δεδομένων σε ομόκεντρους κύκλους με τα δεδομένα στον πυρήνα του. Κάθε στρώση έχει πληροφορίες ελέγχου και εξασφάλισης της αξιοπιστίας της μεταφοράς του πακέτου.

- **Έλεγχος Σύνδεσης.** Η τεχνολογία της επικοινωνίας μπορεί να είναι χωρίς σύνδεση (connectionless) ή με σύνδεση (connection oriented). Κατά την πρώτη τα πακέτα στέλνονται χωρίς κάποιο προγραμματισμό για να βρουν τον δρόμο τους προς τον προορισμό τους, βασιζόμενα στις πύλες και στους δρομολογητές να φροντίζουν για την πορεία τους. Στη δεύτερη περίπτωση, ο αποστολέας επικοινωνεί πρώτα με τον παραλήπτη, συμφωνούν να γίνει η επικοινωνία και αρχίζει η μεταφορά των πληροφοριών με συντεταγμένο και καθορισμένο τρόπο. Η σύνδεση αυτή χρησιμοποιείται συνήθως για μεταφορές μεγάλων ποσοτήτων δεδομένων όπως το FTP, το SMTP κλπ. Τα πρωτόκολλα που χρησιμοποιούν μόνο το TCP, όπως τα παραπάνω, εφαρμόζουν αποκλειστικά επικοινωνία με σύνδεση, ενώ τα πρωτόκολλα που χρησιμοποιούν το IP εφαρμόζουν ασύνδετη επικοινωνία, όπως το Internet Control Message Protocol (ICMP) και το Exterior Gateway Protocol (EGP).

- **Διευθυνσιοδότηση.** Εδώ υπάρχει η έννοια του υποδικτύου (Subnetwork), το οποίο παρέχει την διεύθυνση του υπολογιστή όπως αυτή χρησιμοποιείται στο τοπικό του δίκτυο. Υπάρχει όμως και η διεύθυνση IP σε επίπεδο Internet, η οποία αντιπροσωπεύει τον αποστολέα στο παγκόσμιο Διαδίκτυο. Το Επίπεδο Δικτύου είναι επιφορτισμένο με την αντιστοίχιση της IP διεύθυνσης του υπολογιστή στο υποδίκτυο, με την IP διεύθυνση που χρησιμοποιεί για την επικοινωνία με τον έξω κόσμο. Τέλος, όταν τα δεδομένα φτάσουν στον παραλήπτη, περνάνε στο Επίπεδο Μεταφοράς (TCP) και πρέπει να "δοθούν" στη διαδικασία που τα περιμένει. Εδώ βασικό ρόλο παίζει η θύρα στην οποία περιμένει η διαδικασία να "ακούσει" τα δεδομένα. Συνεπώς ο συνδυασμός της θύρας (η οποία είναι μοναδική σε κάθε υπολογιστή) και της IP με την οποία η διαδικασία επικοινωνεί στο παγκόσμιο διαδίκτυο ορίζουν μοναδικά μια διαδικασία ή εφαρμογή. Αυτή λοιπόν η τριάδα διευθύνσεων, δηλαδή η θύρα, η διεύθυνση του υπολογιστή (Subnetwork attachment point address) και η διεύθυνση στο Internet,

αποτελούν το λεγόμενο Socket. Υπάρχει περίπτωση το socket να αποτελείται μόνο από την θύρα και την IP διεύθυνση που "βγαίνει" στο παγκόσμιο Internet.

Υπάρχουν διάφορα είδη από sockets όπως:

- **Datagram Sockets**, γνωστά και ως μη συνδεδεμένα, τα οποία χρησιμοποιούν το User Datagram Protocol (UDP)
- **Stream sockets**, γνωστά και ως συνδεδεμένα, τα οποία χρησιμοποιούν πρωτόκολλο TCP ή Stream Control Transmission Protocol (SCTP)
- **Raw sockets** όπου το επίπεδο Μεταφοράς παρακάμπτεται και οι κεφαλίδες των πακέτων οδηγούνται κατευθείαν στην διαδικασία ή στην εφαρμογή.

1.5.13. Αρχιτεκτονικές TCP/ IP και OSI

Η αρχιτεκτονική που περιγράψαμε αφορά το πρωτόκολλο μεταφοράς TCP/IP. Υπάρχει όμως και μια ισοδύναμη αρχιτεκτονική, η οποία δημιουργήθηκε για να υιοθετήσει τα παγκόσμια πρωτόκολλα των επικοινωνιακών δικτύων. Αυτή είναι η αρχιτεκτονική OSI (International Standards Protocol). Στο παρακάτω σχήμα βλέπουμε την αντιστοιχία των επιπέδων των δύο αρχιτεκτονικών.

Σχήμα 1.15 *Σύγκριση των Αρχιτεκτονικών TCP/IP και OSI*

Για περισσότερη μελέτη των δύο αρχιτεκτονικών και των διαφορών τους προτείνεται αναφορά στο σχετικό κεφάλαιο του βιβλίου «Δίκτυα Υπολογιστών» της Γ ' τάξης των ΕΠΑΛ.

1.6. Τεχνολογίες Υπολογιστικού Σύννεφου

Το Cloud Computing ή στα ελληνικά "Υπολογιστικό Νέφος" αποτελεί μια νέα τεχνολογία η οποία μας παρέχει μέσα από το Διαδίκτυο πόρους (όπως δίκτυο, εξυπηρετητές, εφαρμογές και υπηρεσίες) με υψηλή ευελιξία, ελάχιστη προσπάθεια από τον χρήστη και υψηλή αυτοματοποίηση.

Ένα χαρακτηριστικό παράδειγμα είναι το EC2 (Amazon Elastic Compute Cloud). Το EC2 επιτρέπει σε χρήστες να νοικιάζουν εικονικούς υπολογιστές, στους οποίους έχουν τα δεδομένα τους, τις εφαρμογές τους, την επεξεργασία των δεδομένων που χρειάζονται κλπ. Επίσης το σύννεφο έχει σχεδιαστεί για να παρέχει web scale προγραμματισμό, δηλαδή να υπάρχει ένας τεράστιος αριθμός υπολογιστών οι οποίοι μπορούν να εξυπηρετούν απαιτητικές υπολογιστικές ανάγκες. Είναι ένα παράδειγμα αναδυόμενου μοντέλου που διαδίδεται σταθερά στον κόσμο του ηλεκτρονικού εμπορίου και όχι μόνο.

Υπάρχει ακόμη και η τάση ενσωμάτωσης σχεδόν όλων των ψηφιακών υπηρεσιών στο σύννεφο, σε συνδυασμό με την έννοια του "The Internet of Things", δηλαδή της σύνδεσης των πάντων στο Διαδίκτυο όπως για παράδειγμα είναι οι λειτουργίες και οι υπηρεσίες μιας πόλης. Όλα φαίνονται τόσο εύκολα όταν οι ψηφιακές υπηρεσίες γίνονται ευκολότερες με μια απλή σύνδεση στο Διαδίκτυο τόσο για τον απλό χρήστη όσο και για τις επιχειρήσεις.

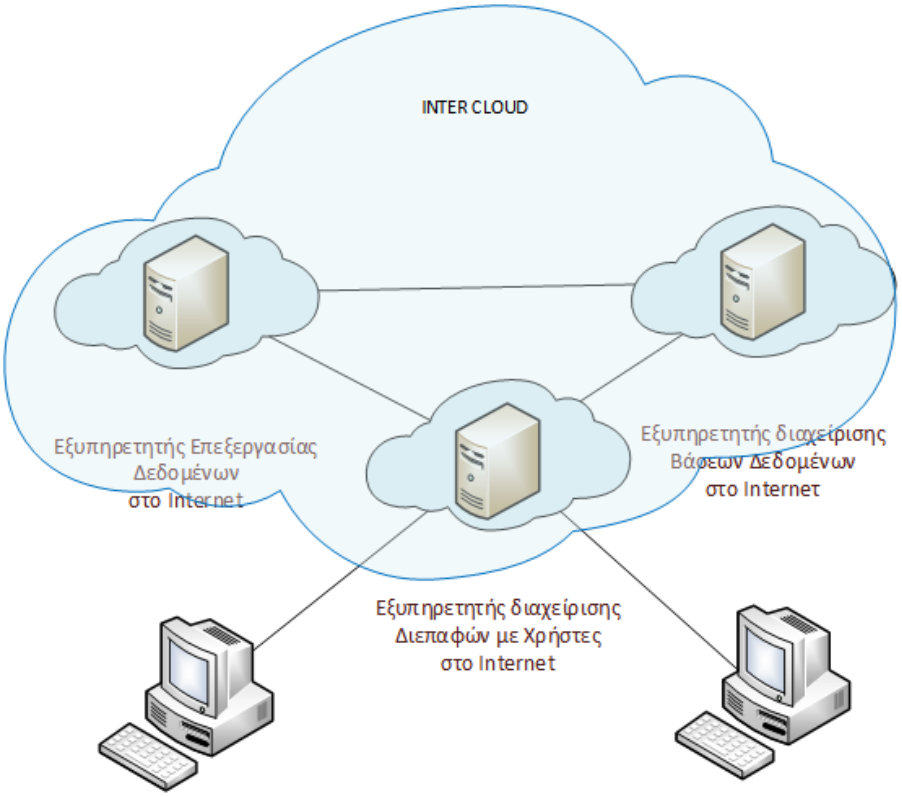
Στο υπολογιστικό νέφος η αποθήκευση, η επεξεργασία και η χρήση δεδομένων, λογισμικού και υπηρεσιών γίνεται διαδικτυακά, μέσω απομακρυσμένων υπολογιστών σε κεντρικά υπολογιστικά κέντρα. Υπηρεσίες όπως η παροχή εικονικών μηχανών, το διαδικτυακό ηλεκτρονικό ταχυδρομείο, η επικοινωνία με εικόνα ή ήχο καθώς και τα περιβάλλοντα κοινωνικής δικτύωσης συχνά βασίζονται στην τεχνολογία του υπολογιστικού νέφους.

Οι χρήστες εξοικονομούν χρήματα και προσωπικό από την αγορά, την συντήρηση και συγγραφή λογισμικού, την αγορά και εγκατάσταση ακριβών εξυπηρετητών καθώς και εγκαταστάσεων αποθήκευσης δεδομένων.

Μια τελευταία μελέτη από το Boston Consulting Group, κατ' εντολή της Microsoft, έδειξε την σταθερά ανοδική πορεία των τεχνολογιών αυτών. Η άνοδος αυτή (περίπου 20% ετησίως) φαίνεται και από έρευνα του Ινστιτούτου Οικονομικών Επιστημών Xefri. Τα οφέλη του περιβάλλοντος αυτού άμεσης πρόσβασης σε υπηρεσίες, βάσεις δεδομένων και υπολογιστικής επεξεργασίας είναι σημαντικά τόσο για τις μικρές επιχειρήσεις όσο και για τις μεγαλύτερες πολυεθνικές .

Παρόλα αυτά όμως το 2011, υπήρξε προσωρινή δυσλειτουργία του Dropbox, αφήνοντας εκατομμύρια χρήστες χωρίς τις υπηρεσίες του, όπως επίσης και τον Αύγουστο του 2013 έγινε γενικευμένη βλάβη στη πλατφόρμα της εταιρίας Amazon, η οποία ήταν βασισμένη στο υπολογιστικό σύννεφο. Αλλά ούτε και τα προσωπικά

δεδομένα είναι πλέον ασφαλή όταν διέρρευσαν προσωπικές φωτογραφίες χρηστών από μέσα αποθήκευσης προσωπικών δεδομένων. Το υπολογιστικό σύννεφο είναι μια εξαιρετικά ευέλικτη και χρήσιμη τεχνολογία. Πρέπει όμως να ισχυροποιηθεί το νομικό και διαχειριστικό πλαίσιο της λειτουργίας του έτσι ώστε να διασφαλίζει το απόρρητο του χρήστη. Η δημιουργία τόσο των κατάλληλων τεχνικών υποδομών όσο και του νομικού πλαισίου είναι απαραίτητα στοιχεία για την απρόσκοπτη και ασφαλής λειτουργία του υπολογιστικού σύννεφου.



Σχήμα 1.12: *Δομή του υπολογιστικού σύννεφου*

Το υπολογιστικό σύννεφο αποτελεί για μερικούς μια νέα τεχνολογία. Όμως υπάρχει και η άποψη, ότι το νέφος δεν είναι μια νέα τεχνολογία όσο ένα **νέο οικονομικό μοντέλο** για παροχή υπολογιστικών υπηρεσιών.

Τα βασικά του χαρακτηριστικά είναι :

- **Αυτοεξυπηρέτηση κατόπιν απαίτησης** (On-demandSelf-Service). Ο χρήστης είναι ελεύθερος να κάνει αβίαστη χρήση των υπηρεσιών όταν και όποτε θέλει χωρίς κανένα περιορισμό.

- **Ευρεία δικτυακή πρόσβαση** (Ubiquitous network access). Είναι μια έννοια όπου η δυνατότητα υπολογισμών είναι διαθέσιμη παντού και πάντα. Σε αντιπαραβολή με την δυνατότητα των επιτραπέζιων υπολογιστών (Desktop Computing), η ευρεία δικτυακή πρόσβαση μπορεί να επιτευχθεί σε κάθε συσκευή, σε κάθε τοποθεσία και σε κάθε μορφή. Φορητοί υπολογιστές,

tablets, κινητά τηλέφωνα μέχρι και έξυπνα γυαλιά μπορούν να έχουν πρόσβαση στο υπολογιστικό σύστημα.

- **Δυναμική εκχώρηση πόρων** (Dynamic Resource Allocation). Είναι η δυναμική παροχή πόρων υλικού και λογισμικού στις ξαφνικά μεταβαλλόμενες ανάγκες της ζήτησης με πιο θεμελιώδεις και αποτελεσματικούς τρόπους παρά ποτέ.

- **Ταχεία ελαστικότητα** (Rapidelasticity). Αφορά την δυνατότητα διοχέτευσης υπολογιστικής ισχύος καθώς και χώρου αποθήκευσης σε πολλαπλά εφεδρικά ή εξωτερικά συστήματα στο Διαδίκτυο , όταν η ζήτηση έχει ακραίες αυξομειώσεις.

- **Μετρήσιμη υπηρεσία** (Measured Service). Τα συστήματα μπορούν αυτόματα να ελέγχουν και να παρέχουν βέλτιστη χρήση των πόρων τους (Υπολογιστική ισχύ, χώρος αποθήκευσης, χρόνοι εξυπηρέτησης) ανάλογα με το είδος της υπηρεσίας. Οι πόροι χρησιμοποιούνται μετά από συστηματικό και συνεχή έλεγχο παρέχοντας διαφάνεια στον πάροχο και τον χρήστη της παρεχόμενης υπηρεσίας.

Υπάρχουν τρία βασικά είδη υπολογιστικού σύννεφου :

- **Το Ιδιωτικό ή Private.** Ιδιωτικό υπολογιστικό σύννεφο το οποίο κατασκευάστηκε για να εξυπηρετεί έναν και μόνο πελάτη. Βέβαια και σε αυτά ακόμη υπάρχουν διάφορα είδη ανάλογα με τις λειτουργίες που θα νοικιάζει, τον χώρο αποθήκευσης, την ροή των δεδομένων και γενικά την σύμβαση που θα κάνει. Το βασικό όμως είναι ότι δεν θα έχει πρόσβαση σε όλα αυτά παρά μόνο αυτός. Παρά το ιδιοκτησιακό καθεστώς που θα έχει ο πελάτης, το σύνολο των υποδομών - σε όποιο σημείο του πλανήτη και αν βρίσκονται - ανήκουν στην εταιρία που τα εκμεταλλεύεται.

- **Το Δημόσιο Υπολογιστικό Σύννεφο.** Εδώ ο πελάτης δεν έχει κανένα δικαίωμα αποκλειστικής χρήσης ούτε σε υποδομή ούτε σε υπηρεσίες. Μια τέτοια υπηρεσία υποστηρίζει διάφορες εφαρμογές που χρησιμοποιούν οι πελάτες πληρώνοντας το σχετικό αντίτιμο.

- **Το Υβριδικό ή Hybrid.** Στο μοντέλο αυτό έχουμε συνδυασμό των παραπάνω μοντέλων σε όποιο βαθμό επιλέξει ο πελάτης.

Υπάρχουν και δευτερεύουσες κατηγοριοποιήσεις όπως:

- **Κοινοτικό Σύννεφο,** όπου γίνεται χρήση οργανισμών (Δήμου, Κοινότητας, Πολιτών) και άλλων πόρων στο Διαδίκτυο που αφορούν μια συγκεκριμένη κοινότητα.

- **Διαμοιρασμένο Σύννεφο,** όπου το υπολογιστικό σύννεφο αποτελείται από μα συλλογή διαφόρων υπολογιστικών συστημάτων σε διαφορετικές γεωγραφικές τοποθεσίες σε ένα ενιαίο δίκτυο. Τα Διαμοιρασμένα Σύννεφα χωρίζονται με την σειρά τους σε δύο τύπους:

    - **Στα Δημόσια Διαμοιρασμένα Σύννεφα,** τα οποία είναι πιο κοντά στην έννοια του Κατανεμημένου Υπολογισμού (Distributed Computing)

    - **Στα Εθελοντικά Σύννεφα,** όπου το σύννεφο αποτελείται από το "κτίσιμο" ανεξάρτητων πόρων που διατίθενται εθελοντικά. Ονομάζονται επίσης **Peer to Peer Clouds ή Ad Hoc Clouds**. Μια τέτοια ενδιαφέρουσα πλατφόρμα είναι η Cloud@Home, η οποία σκοπεύει να δημιουργήσει μια υποδομή υπολογιστικού σύννεφου χρησιμοποιώντας εθελοντικούς πόρους.

- **Διεθνές Σύννεφο** (InterCloud), όπου έχουμε ένα συνδεδεμένο παγκόσμιο υπολογιστικό σύννεφο από άλλα υπολογιστικά σύννεφα, ή ένα δίκτυο δικτύων, αναπτύσσοντας κατά αυτόν τον τρόπο την δυνατότητα της **διαλειτουργικότητας** (Interoperability).

- **Πολυσύννεφο** (MultiCloud**).** Εδώ έχουμε ένα συγκερασμό από ανομοιογενή υπολογιστικά σύννεφα όσον αφορά την αρχιτεκτονική τους, με σκοπό την παροχή υπηρεσιών στους χρήστες ανεξάρτητα από συγκεκριμένους παρόχους. Διαφέρει από το υβριδικό μοντέλο στο ότι εκείνο συνδυάζει είδη (Δημόσιο, ιδιωτικό κλπ), ενώ εδώ συνδυάζονται υπηρεσίες παρόχων (διαφορετικές εταιρίες και συστήματα).

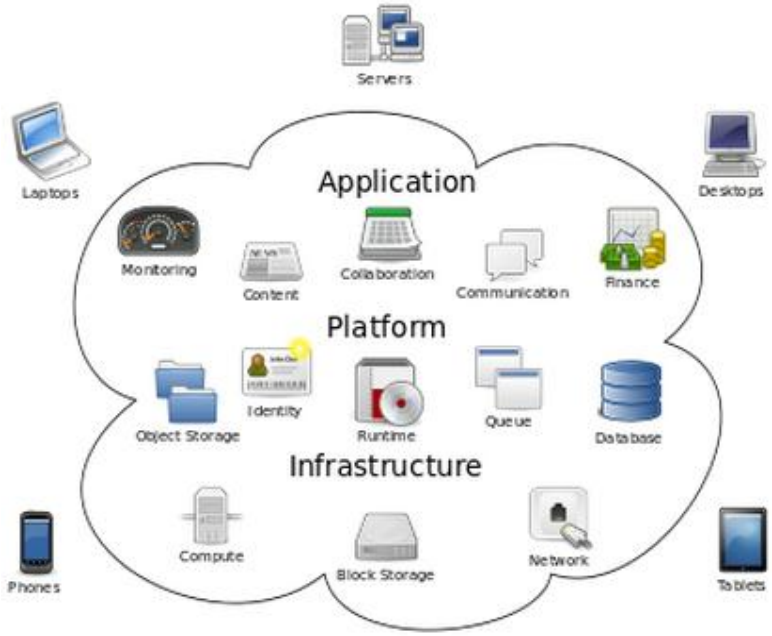
1.6.1. Cloud Computing - Τα μοντέλα υπηρεσιών του Cloud

Οι υπηρεσίες νέφους είναι τριών κατηγοριών ανάλογα με τον τύπο της υπηρεσίας που προσφέρουν (Σχήμα 1.13).

- **Υποδομές ως υπηρεσία** (Infrastructure as a service – IaaS). Είναι η πιο βασική υπηρεσία του σύννεφου, παρέχοντας εικονικούς ή σπανιότερα φυσικούς υπολογιστές για ενοικίαση μέσω του διαδικτύου.

- **Πλατφόρμα ως υπηρεσία** (Platform as a service – PaaS). Η υπηρεσία αυτή είναι πιο εξειδικευμένη, παρέχοντας δυνατότητες περιβάλλοντος επεξεργασίας. Αυτό μπορεί να περιλαμβάνει λειτουργικά συστήματα, βάσεις δεδομένων, γλώσσες προγραμματισμού ή ακόμη και εξυπηρετητές στο Διαδίκτυο (web servers). Τέτοιες είναι η Microsoft Azure, Google App Engine κ.λπ.

- **Λογισμικό ως υπηρεσία** (Software as a service – SaaS). Η υπηρεσία αυτή δίνει την δυνατότητα χρήσης λογισμικού και εγκατεστημένων έτοιμων εφαρμογών από πελάτες.



Σχήμα 1.13: *Υπηρεσίες υπολογιστικού σύννεφου*

1.6.2. Υπολογιστικό Πλέγμα (Grid Computing)

Το πλέγμα είναι μια συλλογή υπολογιστικών πόρων από διάφορα γεωγραφικά σημεία για την επίτευξη ενός σκοπού. Μπορεί να θεωρηθεί ως ένα κατανεμημένο υπολογιστικό σύστημα, χωρίς ιδιαίτερη διάδραση με διεπαφές χρηστών, το οποίο όμως διαχειρίζεται τεράστιο αριθμό αρχείων (https://en.wikipedia.org/wiki/Grid_computing).

Διαφέρει με τα παραδοσιακά κατανεμημένα συστήματα τύπου cluster, στο ότι στο πλέγμα κάθε υπολογιστικό σύστημα επεξεργάζεται το δικό του κομμάτι λογισμικού το οποίο διαφέρει από τους άλλους υπολογιστικούς πόρους. Στα παραδοσιακά συστήματα τύπου cluster, όλα τα υπολογιστικά συστήματα συνεισφέρουν στην ίδια επεξεργασία.

1.7. Αρχιτεκτονική εφαρμογών

Στα Πληροφορικά Συστήματα η αρχιτεκτονική των εφαρμογών είναι ένας από τους πυλώνες του σχεδιασμού της επιχείρησης, όσον αφορά την σχέση της με την πληροφορική.

Η αρχιτεκτονική των εφαρμογών είναι η επιστήμη και η τέχνη του συντονισμού όλων των εφαρμογών οι οποίες χρησιμοποιούνται από μια επιχείρηση, έτσι ώστε να

δημιουργηθεί μια σύνθετη αρχιτεκτονική η οποία να είναι κλιμακωτή, αξιόπιστη, διαθέσιμη πάντα και εύκολη στη διαχείριση της.

Σημαντικό επίσης είναι για έναν αναλυτή, πέρα από την κατανόηση της δυναμικής και της λειτουργικότητας αυτής της σύνθετης αρχιτεκτονικής, να μορφοποιήσει και την στρατηγική υλοποίησης της έχοντας στο νου του τους τεχνολογικούς κινδύνους οι οποίοι θα μπορούσαν να βάλουν σε κίνδυνο την ανάπτυξη των λειτουργιών του οργανισμού ή της επιχείρησης.

Ορισμός: Η Αρχιτεκτονική εφαρμογών περιγράφει την δομή και την οργάνωση των εφαρμογών που χρησιμοποιούνται από μια επιχείρηση ή οργανισμό, εστιάζοντας στο πως αλληλεπιδρούν μεταξύ τους και με τους χρήστες. Επίσης αναλύεται ο τρόπος με τον οποίο παράγονται τα δεδομένα αλλά και χρησιμοποιούνται από τις ίδιες τις εφαρμογές. Το αντικείμενο της αρχιτεκτονικής δεν είναι η εσωτερική δομή των εφαρμογών αλλά η εξωτερική τους συμπεριφορά και η σχέση τους με τα δεδομένα εισόδου / εξόδου.

Η αρχιτεκτονική των εφαρμογών εξειδικεύεται στις απαιτήσεις και ανάγκες της επιχείρησης. Μελετά την επικοινωνία μεταξύ των εφαρμογών του λογισμικού, τις βάσεις δεδομένων και τις εφαρμογές-γέφυρες οι οποίες δρουν συνδέοντας τα διάφορα συστήματα (Διεπαφή χρηστών, Βάσεις Δεδομένων, Λογισμικά Επεξεργασίας Δεδομένων, Διαδίκτυο). Αυτό βοηθά στο να εντοπίσουμε τα δυσλειτουργικά κενά στο σύνολο των εφαρμογών. Στη συνέχεια εντοπίζονται εφαρμογές λογισμικού οι οποίες έφτασαν στο τέλος της ζωής τους (life time cycle) και έχουν τεχνολογικά προβλήματα.

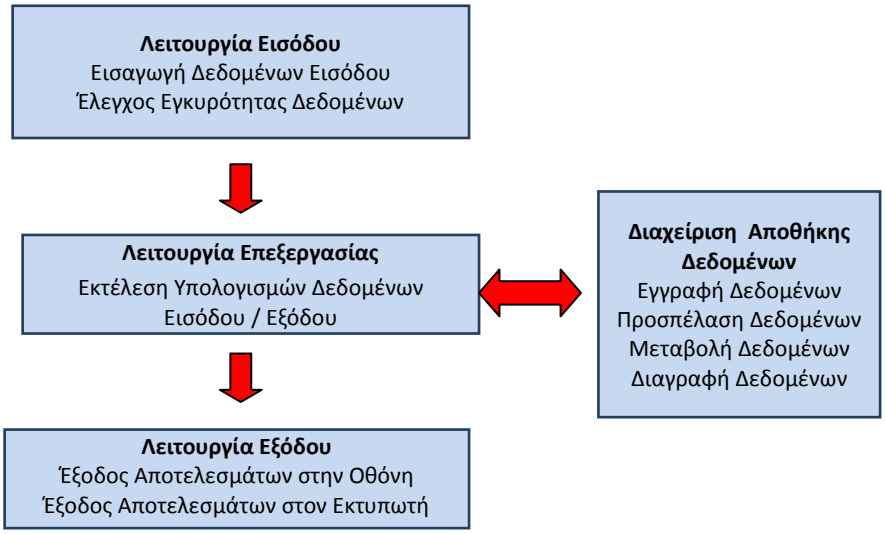
Η αρχιτεκτονική των εφαρμογών παίζει σημαντικό ρόλο, έτσι ώστε όλες οι εφαρμογές να λειτουργούν και να συνεργάζονται αρμονικά και με λειτουργικά οφέλη για τον οργανισμό για τον οποίο σχεδιάστηκαν. Διαφοροποιείται από την αρχιτεκτονική του λογισμικού στο ότι αυτή εστιάζει στη σχεδίαση των εφαρμογών.

Υπάρχουν διάφορα είδη αρχιτεκτονικών εφαρμογών και σε κάποιες περιπτώσεις αν και φαίνονται να έχουν διαφορετικά χαρακτηριστικά, στην ουσία έχουν πολλά κοινά. Ας δούμε λοιπόν δύο από αυτές:

- **Εφαρμογές Επεξεργασίας Δοσοληψιών (Transaction processing applications).** Οι εφαρμογές αυτές είναι επικεντρωμένες στις Βάσεις Δεδομένων (DB Centered applications) και επεξεργάζονται αιτήματα των χρηστών για άντληση πληροφοριών αλλά και ενημέρωση των βάσεων με δεδομένα (διαγραφές, μεταβολές, εισαγωγές δεδομένων). Είναι οργανωμένες με τέτοιο τρόπο ώστε οι κινήσεις των χρηστών δεν συγχέονται μεταξύ τους και η ακεραιότητα της βάσης παραμένει πλήρης.

  Αυτή η κατηγορία εφαρμογών περιλαμβάνει online τραπεζικά συστήματα, e-commerce συστήματα, πληροφοριακά συστήματα και συστήματα κρατήσεων. Για

παράδειγμα αν ο πελάτης κάποιας τράπεζας, πάει σε ένα ATM που βρίσκεται στην Αθήνα για να κάνει μια ανάληψη χρημάτων, ενώ ταυτόχρονα (όσο και να είναι αυτό εφικτό) κάποιος φίλος του προσπαθήσει να μπει στο σύστημα της τράπεζας από μια άλλη πόλη για να κάνει και αυτός μια ανάληψη χρημάτων από τον ίδιο λογαριασμό, η αρχιτεκτονική της τραπεζικής εφαρμογής θα διαφυλάξει την τράπεζα από το να γίνει κάποιο λάθος, με απρόβλεπτα ίσως αποτελέσματα (Σχήμα 1.14.). Η μία δοσοληψία δεν επηρεάζει την άλλη για τον ίδιο λογαριασμό.



Σχήμα 1.14: *Δομή εφαρμογής επεξεργασίας δοσοληψιών*

- **Συστήματα Επεξεργασίας Γλωσσών (Language processing systems).** Στα συστήματα αυτά ο χρήστης του συστήματος παρέχει εντολές σε κάποια τεχνητή γλώσσα προγραμματισμού (π.χ. Python) και το σύστημα αναλαμβάνει να μεταφράσει τις εντολές αυτές σε μια μορφή κατανοητή από το εσωτερικό περιβάλλον του συστήματος. Το πιο γνωστό παράδειγμα ενός τέτοιου συστήματος είναι οι μεταφραστές ή μεταγλωττιστές, οι οποίοι μεταγλωττίζουν τον κώδικα ενός προγράμματος από μια γλώσσα υψηλού επιπέδου σε εντολές που μπορούν να εκτελεστούν από μια βάση δεδομένων ή από ένα πληροφοριακό σύστημα ή από γλώσσες σήμανσης όπως η XML.

Ένας μεγάλος αριθμός εφαρμογών στο Διαδίκτυο εντάσσονται στην κατηγορία των εφαρμογών επεξεργασίας δοσοληψιών (transaction – processing systems) και όλη η ανάπτυξη του λογισμικού στηρίζεται στα συστήματα επεξεργασίας γλωσσών.

1.7.1. Συστήματα Επεξεργασίας Δοσοληψιών

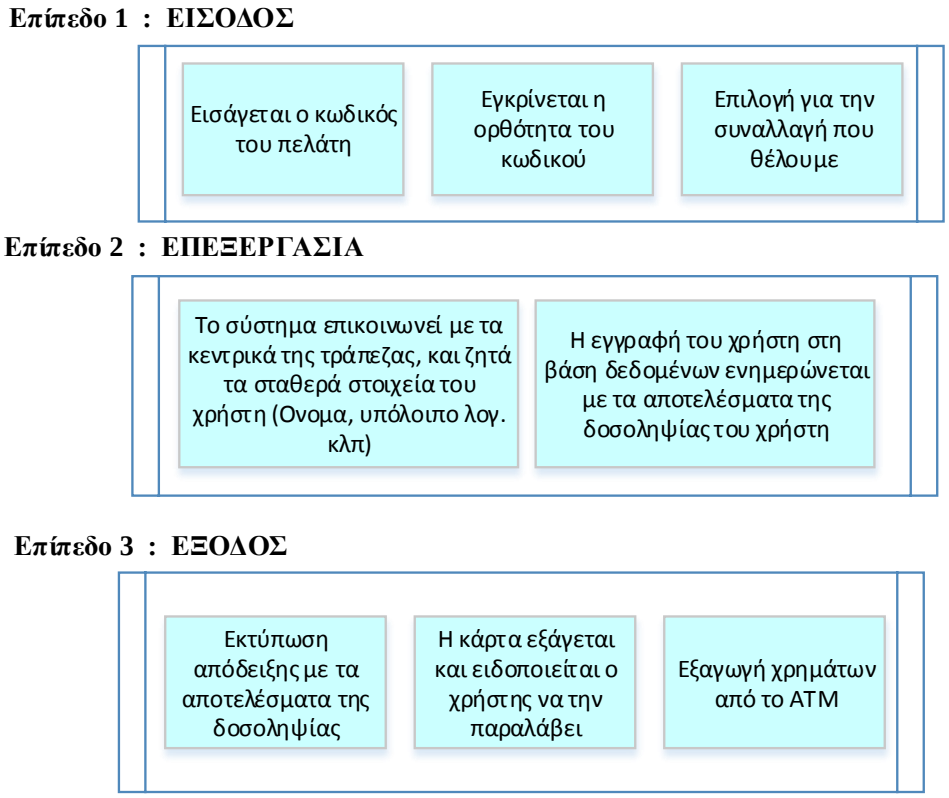
Τα συστήματα αυτά είναι κατά κύριο λόγο σχεδιασμένα να επεξεργάζονται αιτήματα χρηστών, για άντληση δεδομένων από μια βάση δεδομένων, για μεταβολή ή εισαγωγή νέων στοιχείων στη βάση. Από τεχνική άποψη ένα αίτημα προς μια βάση δεδομένων είναι μια ακολουθία ενεργειών οι οποίες λαμβάνονται ως μια. Όλες οι

μεμονωμένες ενέργειες πρέπει να ολοκληρωθούν και μετά να γίνουν οι μόνιμες αλλαγές στα στοιχεία της βάσης. Αυτό προφυλάσσει την ακεραιότητα των δεδομένων της βάσης δεδομένων, από λειτουργίες οι οποίες είναι λάθος ή μπορεί να προκαλέσουν προβλήματα σε αυτή.

Από την πλευρά του χρήστη, ένα αίτημα ή μια δοσοληψία είναι μια ενιαία διαδικασία η οποία επιτυγχάνει ένα σκοπό. Για παράδειγμα "…βρες τις ώρες αναχώρησης των πτήσεων για Παρίσι την ημέρα Τρίτη ….". Αιτήματα σαν και αυτό δεν επηρεάζουν την δομή της βάσης, απλά αντλούν δεδομένα και δεν είναι θεωρούνται δομικές μεταβολές στο περιεχόμενο της βάσης.

Ας κοιτάξουμε με μεγαλύτερη λεπτομέρεια το παραπάνω παράδειγμα με τις τραπεζικές συναλλαγές από ένα τραπεζικό λογαριασμό μέσω ενός ATM. Αυτό απαιτεί να δοθεί ο Κωδικός του Πελάτη, εύρεση του υπολοίπου του λογαριασμού του, ενημέρωση του υπολοίπου του με το ποσό που αφαιρέθηκε και αποστολή από την τράπεζα στη μηχανή του ATM για να "δώσει" τα χρήματα στον πελάτη. Μέχρι να ολοκληρωθούν όλα αυτά τα βήματα ΔΕΝ ενημερώνεται η εγγραφή του πελάτη στη βάση δεδομένων της τράπεζας.

Τα συστήματα δοσοληψιών, είναι συνήθως επιγραμμικά συστήματα (online) στα οποία οι χρήστες κάνουν ασύγχρονες δοσοληψίες. Το Σχήμα 1.15 παρουσιάζει την δομή της αρχιτεκτονικής μιας τέτοιας εφαρμογής.



Σχήμα 1.15: *Αρχιτεκτονική Εφαρμογής Συστήματος ATM*

1. Πρώτα ο χρήστης πρέπει να κάνει ένα αίτημα στο σύστημα μέσα από ένα τμήμα της εφαρμογής υπεύθυνο για την λήψη του αιτήματος.

2. Μετά το αίτημα του χρήστη επεξεργάζεται από κάποια διεργασία του συστήματος.

3. Μια νέα δοσοληψία γεννιέται και μεταφέρεται στον τμήμα της εφαρμογής που είναι υπεύθυνο για την διαχείριση των δοσοληψιών και συνήθως βρίσκεται ενσωματωμένο στο σύστημα διαχείρισης της βάσης (DataBase Management System).

4. Αφού το τμήμα διαχείρισης δοσοληψιών κρίνει ότι το αίτημα έχει ολοκληρωθεί επιτυχώς, σηματοδοτεί στην εφαρμογή ότι η δοσοληψία έχει τελειώσει.

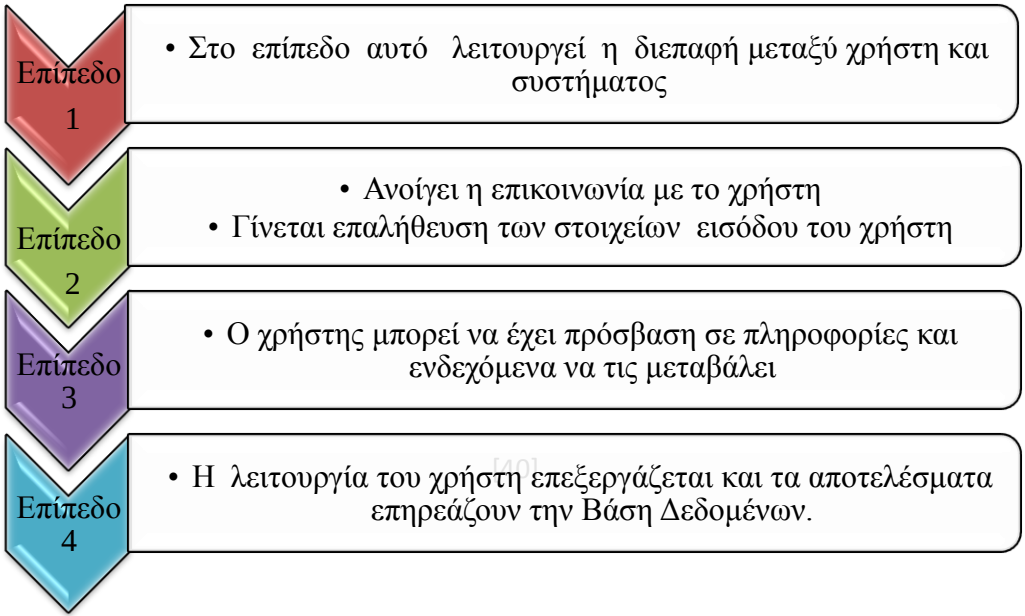
Τα συστήματα επεξεργασίας δοσοληψιών μπορούν να οργανωθούν ως συστήματα αρχιτεκτονικής "διοχέτευσης και φίλτρων" (Pipe and Filter), με τμήματα της εφαρμογής τα οποία εισάγουν – επεξεργάζονται – εξάγουν. Το σύστημα εδώ αποτελείται από δύο τμήματα. Το ένα είναι το λογισμικό του ATM και το λογισμικό επεξεργασίας του τραπεζικού λογαριασμού στο σύστημα διαχείρισης της ΒΔ.

Πληροφοριακά Συστήματα

Όλα τα συστήματα που έχουν αμφίδρομη επικοινωνία με μια διαμοιρασμένη βάση δεδομένων θεωρούνται πληροφοριακά συστήματα βασισμένα στις δοσοληψίες. Ένα τέτοιο πληροφοριακό σύστημα επιτρέπει ελεγχόμενη πρόσβαση σε βάση δεδομένων μεγάλης κλίμακας, όπως κατάλογοι βιβλιοθηκών, πρόγραμμα πτήσεων αεροδρομίων ή αρχείο ασθενών σε ένα νοσοκομείο.

Με αυξανόμενους ρυθμούς αναπτύσσονται και εφαρμογές στο Διαδίκτυο, οι οποίες βασίζονται σε Διαδικτυακές βάσεις δεδομένων (Web Based Databases) και οι χρήστες έχουν πρόσβαση σε αυτές μέσα από τους φυλλομετρητές τους (Browsers).

Το Σχήμα 1.16 αποτυπώνει ένα πολύ γενικό μοντέλο ενός πληροφοριακού συστήματος. Το σύστημα αυτό είναι οργανωμένο σε επίπεδα (Layered) όπου το επίπεδο της κορυφής αποτελεί την Διεπαφή του χρήστη και το κάτω επίπεδο είναι η Βάση των Δεδομένων.



Σχήμα 1.16: *Αρχιτεκτονική Πληροφοριακού Συστήματος με Επίπεδα*

Το επίπεδο επικοινωνίας των χρηστών διοχετεύει όλα τα δεδομένα εισόδου ή εξόδου από το επίπεδο διεπαφής και το επίπεδο άντλησης πληροφοριών περιέχει εξειδικευμένη λογική για πρόσβαση και ενημέρωση της βάσης δεδομένων. Τα επίπεδα αυτά μπορούν να αντιστοιχιστούν σε εξυπηρετητές του διαδικτύου, αν θελήσουμε να εφαρμόσουμε το σύστημα αυτό και στο Web.

Ένα παράδειγμα εφαρμογής του συστήματος των επιπέδων απεικονίζεται στο Σχήμα 1.16, το οποίο δείχνει μια αρχιτεκτονική MHC – PMS (Mental Health Care - Patient Management System). Το σύστημα αυτό παρέχει και διαχειρίζεται πληροφορίες με βάση τις οποίες ασθενείς συμβουλεύονται γιατρούς σχετικά με ένα ιατρικό τους πρόβλημα. Συγκεκριμένα :

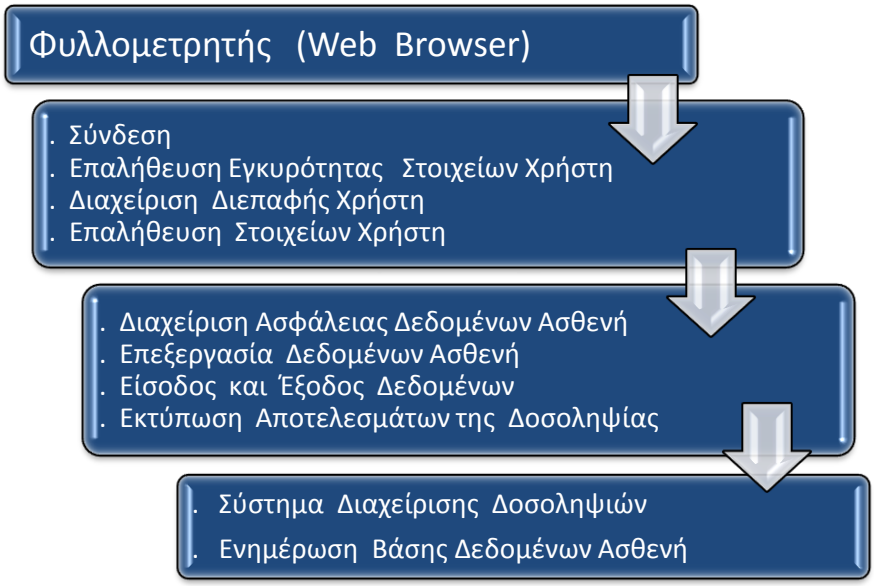
1. Το επάνω επίπεδο έχει ως αποστολή την επικοινωνία του συστήματος με τον χρήστη. Στο συγκεκριμένο παράδειγμα η επικοινωνία γίνεται μέσω του φυλλομετρητή (web browser).

2. Το δεύτερο επίπεδο παρέχει στο επάνω επίπεδο την λειτουργικότητα και τις υπηρεσίες του συστήματος μέσω του φυλλομετρητή. Περιέχει τμήματα λογισμικού που παρέχουν στον χρήστη την δυνατότητα να κάνει σύνδεση με το σύστημα (LOGIN) καθώς και άλλα τμήματα λογισμικού τα οποία ελέγχουν την εγκυρότητα των κωδικών εισόδου του χρήστη αλλά και των επιλογών και δραστηριοτήτων που επιλέγει ο χρήστης. Επίσης το επίπεδο αυτό, παρέχει διαχείριση των οθονών που εμφανίζονται στο χρήστη καθώς και το σύστημα πλοήγησης μέσω των μενού της εφαρμογής.

3. Το τρίτο επίπεδο παρέχει την ασφάλεια του συστήματος, την εισαγωγή και δημιουργία εγγραφών των νέων ασθενών και ενημέρωση των στοιχείων των ήδη υπαρχόντων. Επίσης ασχολείται με την εξαγωγή και είσοδο δεδομένων από άλλες βάσεις δεδομένων, καθώς και με την παροχή εκτυπώσεων και αναφορών.

4. Τέλος το τελευταίο επίπεδο το οποίο συνήθως δημιουργείται από κάποιο Σύστημα Διαχείρισης Βάσεων Δεδομένων, παρέχει διαχείριση των αιτημάτων και δοσοληψιών που δημιουργεί ο χρήστης /ασθενής.

Τα συστήματα πληροφόρησης και διαχείρισης των πόρων του συστήματος γίνονται στην εποχή μας μέσα από το Διαδίκτυο (WEB). Για παράδειγμα τα συστήματα ηλεκτρονικού εμπορίου (e-commerce), γίνονται μέσω διαδικτύου, τα οποία δέχονται ηλεκτρονικές παραγγελίες για προϊόντα και υπηρεσίες και μετά φροντίζουν για την παράδοση των προϊόντων αυτών ή των υπηρεσιών.



Σχήμα 1.17: *Αρχιτεκτονική Πληροφοριακού Συστήματος στο Διαδίκτυο με Επίπεδα*

Σε ένα σύστημα ηλεκτρονικού εμπορίου το επίπεδο που εκτελεί τις βασικές εργασίες του συστήματος, προβλέπει την λειτουργία καλαθιού, όπου ο χρήστης τοποθετεί μέσα τα προϊόντα που επιθυμεί να αγοράσει και να πληρώσει συνολικά με μια συναλλαγή για όλα αυτά.

Η οργάνωση των εξυπηρετητών στα συστήματα αυτά είναι σαν την οργάνωση των 4 επιπέδων του Σχήματος 1.17. Τα συστήματα αυτά υλοποιούνται με αρχιτεκτονικές πολλαπλών διατάξεων (Multi Tier Architectures), όπως πελάτη / εξυπηρετητή.

1. Ο διαδικτυακός εξυπηρετητής (WebServer) είναι υπεύθυνος για τις επικοινωνίες με το χρήστη, μέσω της διεπαφής του φυλλομετρητή.
2. Ο εξυπηρετητής εφαρμογών (ApplicationServer) είναι υπεύθυνος για την εκτέλεση του λογισμικού των εφαρμογών καθώς και για την διαχείριση των αποθηκευτικών χώρων.
3. Τέλος ο εξυπηρετητής των βάσεων δεδομένων (DatabaseServer) ασχολείται με την είσοδο ή έξοδο δεδομένων από τις Βάσεις Δεδομένων και διαχειρίζεται τις δοσοληψίες που κάνει ο χρήστης.

Με την χρήση πολλαπλών εξυπηρετητών είναι δυνατόν για το σύστημα να δέχεται εκατοντάδες δοσοληψίες το λεπτό. Όσο η ζήτηση αυξάνεται, επιπλέον εξυπηρετητές μπορούν να προστεθούν στο σύστημα κάθε ενός από τα παραπάνω επίπεδα για να ανταπεξέλθουν στις παραπάνω απαιτήσεις επεξεργασίας.

Ερωτήσεις – Δραστηριότητες - Θέματα προς συζήτηση

1. Αναζητείστε online λογισμικό π.χ. στο www.onlineocr.net και αναζητείστε κατηγορία (ή κατηγορίες) λογισμικού που θα μπορούσε αυτό να ενταχθεί.

2. Αναζητείστε στο Διαδίκτυο λογισμικό που εντάσσετε σε καθεμία από τις κατηγορίες που δόθηκαν στην πρώτη ενότητα (ενότητα 1.1.) του παρόντος κεφαλαίου.

3. Πολίτες αναζητούν την δυνατότητα της συνδιαμόρφωσης νομοθετικών παρεμβάσεων της κυβέρνησης τους σε όλα τα θέματα. Δείτε αν υπάρχει στηχώρα μας τέτοια παρέμβαση (Διαδικτυακή εφαρμογή). Δώστε σε αυτή την Διαδικτυακή εφαρμογή τους επωφελούμενους, τον πελάτη και τους χρήστες.

4. Ποιες οι διαφορές Πληροφοριακού Συστήματος και λογισμικού;. Δώστε την προσέγγισή σας μέσα από παραδείγματα.

5. Έχοντας τα στοιχεία που δίνονται στην ενότητα 1.4. για την δημιουργία ενός Διαδικτυακού τόπου χρησιμοποιείστε το λογισμικό Gantter (ή όποιο άλλο λογισμικό διαχείρισης έργων είναι διαθέσιμο) για την αποτύπωση του έργου και την δημιουργία ενός διαγράμματος Gantt για το όλο έργο.

6. Ποιές είναι οι μορφές επικοινωνίας στο Διαδίκτυο;

7. Σε τι διαφέρει το πρωτόκολλο POP3 από το IMAP;

8. Εγκαταστήσετε μια εφαρμογή διαμοιρασμού αρχείων ομότιμου δικτύου και διερευνήστε το περιβάλλον του (π.χ. utorrent).

9. Περιγράψτε τα τρία μοντέλα υπηρεσιών του υπολογιστικού σύννεφου.

10. Ποιά είναι τα βασικά είδη υπολογιστικών σύννεφων;

11. Έστω ότι θέλουμε να υλοποιήσουμε ένα σύστημα παροχής ιατρικών υπηρεσιών για μια ομάδα απομακρυσμένων νησιών. Σχεδιάστε τα επίπεδα ενός συστήματος στο Διαδίκτυο, το οποίο να παρέχει σύνδεση, υποβολή ερώτησης, επεξεργασίας της ερώτησης με την χρήση βάσης δεδομένων και αποστολή απάντησης.

12. Στο εργαστήριο πληροφορικής, εγκαταστήσετε σε δύο υπολογιστές μια εφαρμογή για απομακρυσμένο έλεγχο όπως το (Team Viewer). Στη συνέχεια ενεργοποιήστε την εφαρμογή και ελέγξτε τον ένα υπολογιστή από τον άλλο.

Βιβλιογραφία

Στα Ελληνικά

Αποστολάκης Ι. (2007). *Πληροφοριακά Συστήματα Υγείας*. Εκδόσεις Παπαζήση.

Αποστολάκης, Ι. & Τζαναβάρης Δ. (2015). *Εφαρμογές Συνεργατικού Διαδικτύου*. Εκδόσεις Παπαζήση.

Βεργίνης Δ., Κοντούλη Ε., Λάλας Χ., Λαοπόδης Β., Μανουσαρίδης Ζ. & Μπακογιάννης Σ. (2000). *Πληροφοριακά Συστήματα*. Παιδαγωγικό Ινστιτούτο.

Βεσκούκης Β**.** (2000). *Τεχνολογία Λογισμικού Ι*. ΕΑΠ.

Βεσκούκης Β. (2001). *Τεχνολογία Λογισμικού ΙΙ*. ΕΑΠ.

Γιακουμάκης Ε. & Διαμαντίδης Ν. (2009). *Τεχνολογία Λογισμικού*. Εκδόσεις Σταμούλης.

Στα Αγγλικά

PMI (2000). *A Guide to the Project Management Body of Knowledge*. Newtown Square, PA: Project Management Institute.

Qi Zhang Q., Cheng L. & Boutaba R. (2010). *Cloud computing: state-of-the-art and research challenges*. J Internet ServAppl (2010) 1: 7–18.

Marscic I. (2012). *Software Engineering*. Rutgers University, New Brunswick, New Jersey.

Sommerville I. (2011). *Software Engineering*. Addison-Wesley.

Stallings W. (1989). *Handbook of Computer Communications Standards. The TCP/IP Protocol Suite*. 2nd Edition, Howard W.Sams & Company.

Staten J. (2009). *Which Cloud Computing Platform is right for you?*. Forrester Research Inc.

Turner J.R. (1999). *The handbook of project-based management: improving the processes for achieving strategic objectives*. 2nd ed. London : McGraw-Hill.

Turner J.R. & Müller R. (2003). *On the Nature of a Project as a Temporary Organization*. International Journal of Project Management, vol. 21, 1-8.

Διευθύνσεις στο Internet

1. http://www.epset.gr/el/content/ypologistiko-nefos-cloud-computing
2. http://www.inc.com/centurylink/cloud-fueled-growth-for-small-and-medium-businesses.html
3. http://www.bluegreencomputing.com/#!---cloud-computing-ii/c20c8
4. http://ebooks.edu.gr/courses/DSGL-A121/document/544a44c69azz/544a44ca1ki0/544a44f4nenj.pdf
5. http://ebooks.edu.gr/modules/ebook/show.php/DSGL-C115/424/2836,10778/

6. http://www.technologicasol.com/cloud-services.html

7. http://en.wikipedia.org/wiki/Grid_computing

8. http://conta.uom.gr/conta/ekpaideysh/metaptyxiaka/technologies_diktywn/ergasies/2011/Networks%20for%20grid%20and%20cloud%20computing.pdf

9. http://www.cis.aueb.gr/Publications/INFOCOM-2011%20Site.pdf

Κύκλος ζωής ανάπτυξης συστήματος

Περιεχόμενα

2.1 Ανάπτυξη Συστήματος
2.2 Κύκλος Ζωής Ανάπτυξης Συστήματος
2.3 Κλασσικές προβλέψιμες προσεγγίσεις του Κύκλου ζωής Ανάπτυξης Συστήματος
2.4 Ευέλικτες – Προσαρμοστικές προσεγγίσεις του Κύκλου ζωής Ανάπτυξης Συστήματος
2.5 Στάδια του Κύκλου ζωής Ανάπτυξης Συστήματος
2.6 Μεθοδολογίες, μοντέλα, εργαλεία και τεχνικές
2.7 Δομημένη Ανάπτυξη Συστήματος (SADT)
2.8 Αντικειμενοστραφής Ανάπτυξη Συστήματος
2.9 Σύγχρονες Τάσεις στην Ανάπτυξη Συστήματος
2.9.1 Ενοποιημένη διαδικασία (UM)
2.9.2 Ακραίος προγραμματισμός (XM)
2.9.3 Μεθοδολογία Scrum
Ερωτήσεις – Δραστηριότητες – Θέματα προς Συζήτηση
Μελέτη περίπτωσης (Σχολική Βιβλιοθήκη)

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι:

- Να περιγράφουν τα διάφορα στάδια του Κύκλου Ζωής Ανάπτυξης Συστήματος καθώς και να αναγνωρίζουν την συμβολή αυτών στην δημιουργία ενός αποτελεσματικού Πληροφοριακού Συστήματος.
- Να περιγράφουν τις διάφορες προσεγγίσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος.

- Να επιλέγουν και να εφαρμόζουν την κατάλληλη προσέγγιση του Κύκλου Ζωής Ανάπτυξης Συστήματος τόσο σε απλά σενάρια από τον κόσμο των επιχειρήσεων και των οργανισμών όσο και στην καθημερινή τους ζωή.
- Να συγκρίνουν και να αντιπαραβάλλουν διαφορετικές μεθοδολογίες ανάπτυξης συστημάτων και τεχνικών.
- Να εφαρμόζουν αντικειμενοστραφείς και κλασσικές μεθοδολογίες.
- Να περιγράφουν τις σύγχρονες τάσεις στην ανάπτυξη ενός πληροφοριακού συστήματος.

2.1 Ανάπτυξη Συστήματος

Την σημερινή εποχή οι υπολογιστές βρίσκονται παντού. Ζούμε σε ένα ψηφιακό κόσμο, όπου η επικοινωνία και η σύνδεση στο διαδίκτυο αποτελούν μέρος της καθημερινότητάς μας. Οι περισσότερες δραστηριότητες μας τόσο σε προσωπικό επίπεδο όσο και σε επαγγελματικό εξαρτώνται από τα microchips, τις γραμμές δικτύου και τις διάφορες εφαρμογές λογισμικού. Μεγαλώνουμε σε ένα κόσμο υψηλής τεχνολογίας. Χρησιμοποιούμε smartphones, laptops, iPads, notepads και εξοπλισμούς ηλεκτρονικών παιχνιδιών. Μέσω κινητών συσκευών λαμβάνουμε μηνύματα, tweets, παρακολουθούμε βίντεο, παίζουμε παιχνίδια και περιηγούμαστε στο διαδίκτυο. Ποιος είναι όμως ο ρόλος των **συστημάτων ανάλυσης και σχεδιασμού** στην ανάπτυξη υψηλής τεχνολογίας λύσεων και εφαρμογών;

Για να απαντήσουμε στην παραπάνω ερώτηση αυτή ας μελετήσουμε μία παρόμοια κατάσταση, αυτή της κατασκευής κτηρίων. Στο σενάριο αυτό συμμετέχουν ο ιδιοκτήτης/αγοραστής που έχει μία νοερή εικόνα της δημιουργίας του κτηρίου, ο κατασκευαστής που θα κατασκευάσει το κτήριο και ο αρχιτέκτονας που χρησιμεύει ως συνδετικός κρίκος μεταξύ του ιδιοκτήτη και του κατασκευαστή. Ο ρόλος του αρχιτέκτονα είναι να μεταφέρει στον κατασκευαστή εκείνες τις τεχνικές πληροφορίες όπως σχέδια και μακέτες που του είναι απαραίτητες για την κατασκευή του κτηρίου. Οι τεχνικές αυτές πληροφορίες θα προκύψουν μετά από συζήτηση με τον ιδιοκτήτη, κατά την διάρκεια της οποίας ο αρχιτέκτονας θα προσπαθήσει να βοηθήσει τον ιδιοκτήτη να αναπτύξει το όραμα που έχει για το κτήριο. Ο αρχιτέκτονας χρησιμοποιεί διάφορα εργαλεία για να αποτυπώσει τους οραματισμούς του ιδιοκτήτη και στην συνέχεια παραδίδει στον κατασκευαστή σαφείς και ακριβείς οδηγίες για την κατασκευή του κτηρίου σύμφωνα με τις προσδοκίες του ιδιοκτήτη.

Από τα παραπάνω γίνεται σαφές ο σημαντικός ρόλος του αρχιτέκτονα για την επιτυχή κατασκευή του κτιρίου όπως το οραματίστηκε ο ιδιοκτήτης. Ο αρχιτέκτονας έχει να επιτελέσει δύο σημαντικές λειτουργίες. Την ανάλυση και τον σχεδιασμό. Να περιγράψει δηλαδή με λεπτομέρεια το «τι» και το «πώς». «Τι» είναι αυτό το κτήριο που οραματίζεται ο ιδιοκτήτης και «πώς» θα δομηθεί αυτό το σπίτι.

Όπως οι κατασκευαστές όμως δεν αρχίζουν την οικοδόμηση του κτηρίου χωρίς σχέδια, έτσι και οι προγραμματιστές δεν αρχίζουν να προγραμματίζουν, χωρίς να έχουν κάποιον που να έχει αντίστοιχα τον ρόλο του αρχιτέκτονα και ο οποίος θα τους καθοδηγήσει στην δημιουργία του κώδικα μέσω του προσδιορισμού των απαιτήσεων και των περιορισμών του συστήματος. Ο αρχιτέκτονας στο πλαίσιο της ανάπτυξης λογισμικού ονομάζεται **αναλυτής και σχεδιαστής συστημάτων (systems analyst and design)**.

Θα πρέπει να σημειώσουμε ότι η ανάπτυξη ενός πληροφοριακού συστήματος ξεφεύγει από τα στενά όρια του προγραμματισμού και βασίζει την επιτυχία του στην εις βάθος ανάλυση και σχεδιασμό του πληροφοριακού συστήματος. Η δημιουργία ενός συστήματος που βασίζεται σε ελλιπείς ή παρανοημένες απαιτήσεις οδηγεί σε ένα

καθυστερημένο και υπερτιμολογημένο έργο το οποίο δεν επιλύει όλα τα προβλήματα για τα οποία κατασκευάστηκε. Αναπτύσσω ένα πληροφοριακό σύστημα σημαίνει επιλύω προβλήματα με την βοήθεια της τεχνολογίας και όχι μόνο προγραμματίζω. Τι προβλήματα μπορεί να κληθούμε να επιλύσουμε μέσω της ανάπτυξης του κατάλληλου πληροφοριακού συστήματος; Για παράδειγμα, οι πελάτες θέλουν να έχουν την δυνατότητα να παραγγέλνουν προϊόντα όλο το εικοσιτετράωρο. Το πρόβλημα είναι πώς μπορούμε να επεξεργαστούμε αυτές τις παραγγελίες όλο το εικοσιτετράωρο χωρίς να αυξηθεί όμως το κόστος πωλήσεων.

Ο ρόλος λοιπόν του αναλυτή και σχεδιαστή συστημάτων είναι να κατανοήσει τις ανάγκες της επιχείρησης ή του οργανισμού, να συλλάβει το όραμα, να προσδιορίσει την επίλυση του προβλήματος, να επικοινωνήσει το όραμα και την επίλυση, να επιβεβαιώσει ότι η συγκεκριμένη λύση, που υπερκαλύπτει το κόστος ανάπτυξης, ικανοποιεί τις ανάγκες της επιχείρησης ή του οργανισμού και να καθοδηγήσει την αντίστοιχη ομάδα στην υλοποίηση της λύσης. Τα μέλη της ομάδας που συνεργάζονται για την ανάλυση και τον σχεδιασμό του συστήματος πρέπει να κατέχουν τόσο επικοινωνιακές ικανότητες, καθώς θα παίρνουν συνεντεύξεις και θα συνομιλούν με διάφορα μέλη της ομάδας ανάπτυξης όσο και τεχνικές ικανότητες όπως λεπτομερής προσδιορισμός προδιαγραφών και σχεδιασμός λύσεων. Πολλές από τις τεχνικές αυτές ικανότητες σχετίζονται και με την δημιουργία μοντέλων για την αναπαράσταση των προδιαγραφών και της λύσης.

Πιο συγκεκριμένα, η ανάλυση ενός συστήματος περιλαμβάνει εκείνες τις δραστηριότητες που επιτρέπουν στην ομάδα ανάλυσης να κατανοήσει και να προσδιορίσει με ακρίβεια τι ακριβώς πρέπει να πετύχει το νέο σύστημα. Στον παραπάνω ορισμό οι λέξεις κλειδιά είναι «κατανοώ» και «προσδιορίζω». Η ανάλυση ενός συστήματος είναι κάτι πολύ περισσότερο από μια απλή σύντομη δήλωση του προβλήματος. Για παράδειγμα, ένα σύστημα διαχείρισης πελατών θα πρέπει μεταξύ πολλών άλλων περίπλοκων λειτουργιών να παρακολουθεί τους πελάτες, τις εγγραφές των προϊόντων, τις εγγυήσεις και τις διάφορες υπηρεσίες. Η ανάλυση συστήματος περιγράφει λεπτομερώς το "τι" πρέπει να κάνει το προτεινόμενο πληροφοριακό σύστημα για να ικανοποιήσει τις ανάγκες ή να λύσει τα προβλήματα μιας επιχείρησης ή ενός οργανισμού.

Το επόμενο βήμα στην ανάπτυξη ενός συστήματος είναι ο σχεδιασμός. Ο σχεδιασμός ενός συστήματος αποτελείται από εκείνες τις δραστηριότητες που επιτρέπουν στην ομάδα σχεδιασμού να περιγράψει με λεπτομέρεια σε τεχνικό επίπεδο εκείνο το σύστημα που δίνει απάντηση στα υφιστάμενα προβλήματα ή ικανοποιεί τις ανάγκες μιας επιχείρησης ή ενός οργανισμού. Στον παραπάνω ορισμό η λέξη κλειδί είναι «επιλύω». Με άλλα λόγια, ο σχεδιασμός συστημάτων περιγράφει το «πώς» θα λειτουργήσει το σύστημα. Καθορίζει με λεπτομέρεια όλα τα στοιχεία που απαρτίζουν το σύστημα (για παράδειγμα βάσεις δεδομένων, διεπαφές χρηστών, δίκτυα, λογισμικό, λειτουργικές διαδικασίες) και πώς αυτά θα συνεργάζονται μεταξύ τους για την παροχή της επιθυμητής λύσης.

2.2. Κύκλος Ζωής Ανάπτυξης Συστήματος

Η ανάπτυξη ενός συστήματος παρουσιάζει ομοιότητες με αυτές ενός project. Δηλαδή οι δραστηριότητες που απαιτούνται για την ανάπτυξη ενός συστήματος πρέπει να προσδιοριστούν, να χρονο-προγραμματιστούν, να οργανωθούν και να είναι υπό συνεχή εποπτεία. Μπορούμε να θεωρήσουμε την ανάπτυξη ενός συστήματος ως ένα προγραμματισμένο εγχείρημα που έχει μια αρχή και ένα τέλος και παράγει κάποιο προκαθορισμένο αποτέλεσμα. Η διαχείριση της ανάπτυξης ενός συστήματος απαιτεί ένα πλαίσιο διαχείρισης έργου για τον καθορισμό και τον συντονισμό όλων των δραστηριοτήτων της ομάδας έργου. Το πλαίσιο αυτό ονομάζεται **Κύκλος Ζωής Ανάπτυξης Συστήματος (ΚΖΑΣ) (Systems Development Life Cycle (SDLC)).**

Ο Κύκλος Ζωής Ανάπτυξης Συστήματος (ΚΖΑΣ) αποτελεί μία από τις βασικές θεμελιώδεις έννοιες στην ανάπτυξη του πληροφοριακών συστημάτων. Τα πληροφοριακά αυτά συστήματα έχουν την δική τους «ζωή» ή τον δικό τους «κύκλο ζωής» και ανάπτυξης. «Ο κύκλος ζωής» ενός πληροφοριακού συστήματος μπορεί να προσεγγίζεται κάθε φορά διαφορετικά σύμφωνα με τις απαιτήσεις του έργου. Οι διαδικασίες όμως που περιλαμβάνονται σε κάθε στάδιο ανάπτυξης ενός πληροφοριακού συστήματος είναι οι ίδιες. Αξίζει να σημειώσουμε το γεγονός ότι κάθε δραστηριότητα που περιλαμβάνεται σε αυτόν τον κύκλο έχει ως σκοπό να παίρνει αυτό που λέει ο πελάτης και να το μετατρέπει σε κάτι που έχει αξία για τον πελάτη. Όλα γίνονται για τον πελάτη. Ας μελετήσουμε τώρα κάθε ένα από τα στάδια αυτά.

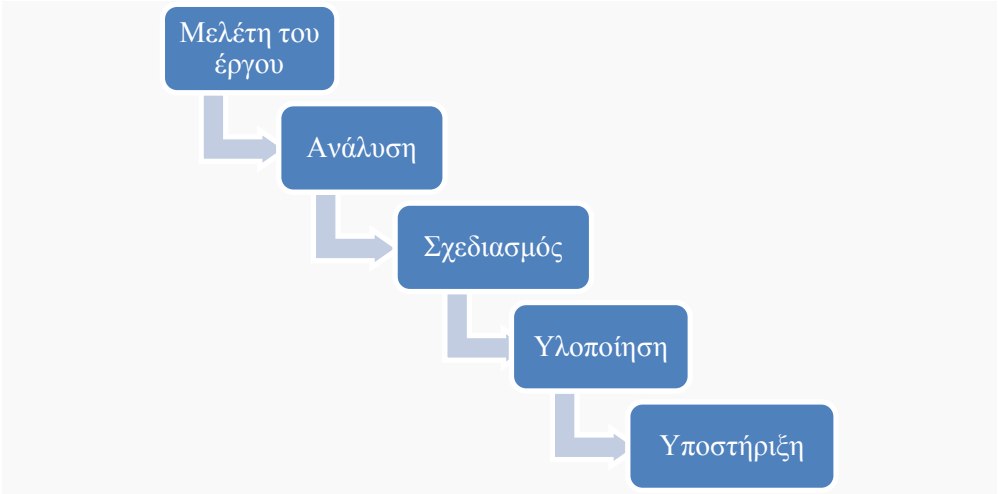
Στις περισσότερες περιπτώσεις υπάρχει ένα σύστημα, ακόμα και αν πρόκειται για χαρτί και μολύβι, που το χρησιμοποιεί ο πελάτης για να επιτελέσει την δουλειά του με τον καλύτερο δυνατό τρόπο. Το σύστημα αυτό ο πελάτης θέλει να το βελτιώσει για να αυξήσει την αποδοτικότητα της επιχείρησης ή του οργανισμού. Ξεκινάμε λοιπόν από το **στάδιο μελέτης του έργου** που έχει σκοπό να προσδιορίσει το πεδίο εφαρμογής του νέου συστήματος, να διασφαλίσει την εφικτότητα του σχεδίου, να αναπτύξει το χρονοδιάγραμμα και να υπολογίσει τον προϋπολογισμό του έργου. Στη συνέχεια στο **στάδιο της ανάλυσης**, συζητάμε με τους πελάτες και προσπαθούμε να βρούμε ποιο είναι το πρόβλημα και τι είναι αυτό που ακριβώς χρειάζονται. Ή «τι δουλεύει και τι δεν δουλεύει» στο παλιό σύστημα. Κατά το **στάδιο του σχεδιασμού,** σχεδιάζεται η απάντηση στην ερώτηση, η λύση στο πρόβλημα με βάση τις πληροφορίες που συγκεντρώθηκαν κατά το στάδιο της ανάλυσης. Στη συνέχεια, **στο στάδιο της υλοποίησης** δημιουργείται το πληροφοριακό σύστημα. Στο **στάδιο της υποστήριξης**, βοηθούμε όσους εμπλέκονται με αυτό να το χρησιμοποιήσουν με τον καλύτερο δυνατό τρόπο. Κάθε στάδιο απαιτεί ένα διαφορετικό τρόπο σκέψης, εργασίας και επικοινωνίας με διαφορετικούς ανθρώπους.

Σε ένα ψηφιακό κόσμο που συνεχώς μεταβάλλεται έχουν αναπτυχθεί και εφαρμοστεί διάφορες προσεγγίσεις ανάπτυξης πληροφοριακών συστημάτων οι οποίες βασίζονται σε διαφορετικούς Κύκλους Ζωής Ανάπτυξης Συστήματος. Οι διαφορετικές αυτές προσεγγίσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος θα μπορούσαν να

ταξινομηθούν με βάση τον βαθμό της προσαρμοστικότητάς τους και της ευελιξίας τους. Στο παρακάτω σχήμα (βλέπε Σχήμα 2.1) στα αριστερά βρίσκονται οι πλήρως προβλέψιμες προσεγγίσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος ενώ στο δεξιό άκρο είναι τοποθετημένες οι πλήρως προσαρμοστικές προσεγγίσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος. Όσο προχωράμε από τα αριστερά προς τα δεξιά τόσο μειώνονται τα στοιχεία της προβλέψιμης συμπεριφοράς και αυξάνονται αυτά της προσαρμοστικής. Στην περίπτωση αυτή αναφερόμαστε σε υβριδικές προσεγγίσεις.

***Σχήμα 2.1**: Η επιλογή του Κύκλου Ζωής Ανάπτυξης Συστήματος εξαρτάται από το έργο*

Στην προβλέψιμη προσέγγιση του Κύκλου Ζωής Ανάπτυξης Συστήματος το έργο σχεδιάζεται και οργανώνεται εκ των προτέρων και το νέο πληροφοριακό σύστημα αναπτύσσεται σύμφωνα με το προβλεπόμενο σχέδιο. Για παράδειγμα, μια εταιρεία μπορεί να θέλει να μετατρέψει το παλιό mainframe σύστημα απογραφής της σε ένα νεότερο διαδικτυακό σύστημα τύπου πελάτη/εξυπηρετητή. Σε αυτήν την περίπτωση οι απαιτήσεις είναι πλήρως καθορισμένες και δεν χρειάζεται να προστεθούν νέες διαδικασίες. Χαρακτηριστικό παράδειγμα της πιο προβλέψιμης προσέγγισης αποτελεί το **μοντέλο καταρράκτη**, όπως φαίνεται στο παρακάτω σχήμα (βλέπε Σχήμα 2.2), όπου τα στάδια ανάπτυξης ενός πληροφοριακού συστήματος εξελίσσονται το ένα μετά από το άλλο με την σειρά. Πριν περάσουμε για παράδειγμα στο στάδιο της σχεδίασης πρέπει να έχει ολοκληρωθεί πλήρως το στάδιο της ανάλυσης. Το στάδιο της υλοποίησης δεν μπορεί να ξεκινήσει πριν την αποπεράτωση του σταδίου της σχεδίασης.



***Σχήμα 2.2**: Μοντέλο Καταρράκτη – Κάθε project έχει μία αρχή και ένα τέλος*

Στην προσαρμοστική προσέγγιση του Κύκλου Ζωής Ανάπτυξης Συστήματος δεν είναι σαφώς καθορισμένες οι απαιτήσεις ή οι ανάγκες των χρηστών. Αυτό έχει ως αποτέλεσμα να μην μπορεί το έργο να προγραμματιστεί πλήρως εκ των προτέρων. Προσδιορίζεται με σαφήνεια ένα μέρος των απαιτήσεων και στη συνέχεια ακολουθούν κάποιες προκαταρκτικές ενέργειες ανάπτυξης. Η λύση πρέπει να είναι ευέλικτη και να προσαρμόζεται κάθε φορά στην εξέλιξη του έργου. Για παράδειγμα, αρχικά η ομάδα ανάπτυξης εστιάζει σε ένα μικρό αλλά σημαντικό κομμάτι του έργου. Ξεκινάει από την ανάλυση και συνεχίζει με την σχεδίαση και την κατασκευή. Μέσα σε μερικές εβδομάδες παρουσιάζεται στον πελάτη το αποτέλεσμα της συγκεκριμένης εργασίας. Ο πελάτης έχοντας στα χέρια του το πρώτο αλλά σημαντικό δείγμα δουλειάς της ομάδας ανάπτυξης μπορεί να πει «Ναι, αυτό είναι το σύστημα που θέλουμε». Επαναλαμβάνοντας την διαδικασία αυτή αρκετές φορές η ομάδα έργου και προσθέτοντας κάθε φορά περισσότερη λειτουργικότητα στο υπό κατασκευή έργο, μπορεί σε σύντομο χρονικό διάστημα να πλησιάσει στο επιθυμητό αποτέλεσμα σύμφωνα πάντα με τις προδιαγραφές του έργου. Για το λόγο αυτό, η διαδικασία ανάπτυξης ενός σύγχρονου πληροφορικού συστήματος δεν είναι γραμμική αλλά επαναλαμβανόμενα κυκλική όπως φαίνεται στο παρακάτω σχήμα (βλέπε Σχήμα 2.3).

Σχήμα 2.3: *Κάθε στάδιο επαναλαμβάνεται αρκετές φορές*

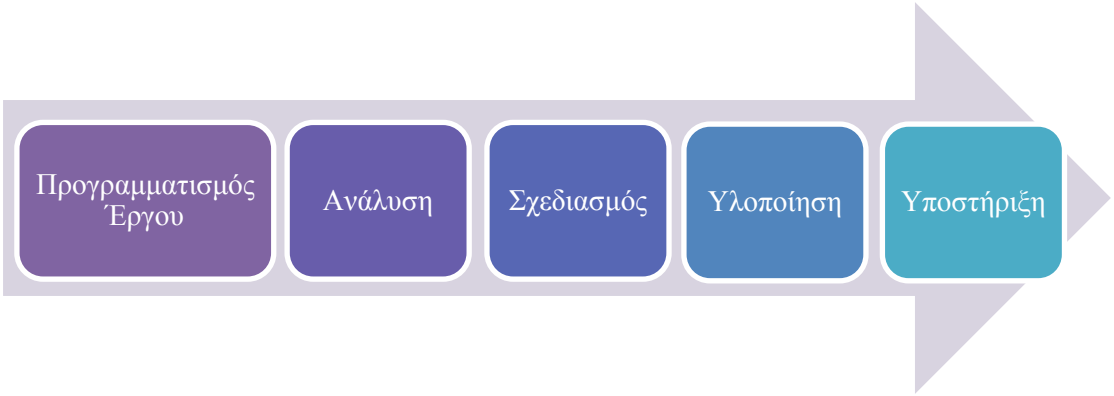
Πρακτικά, τα περισσότερα συστήματα ανάπτυξης ακολουθούν μία υβριδική προσέγγιση που συνδυάζει στοιχεία και από τις δύο προσεγγίσεις. Είναι και προβλέψιμα και προσαρμοστικά σε διαφορετικό βαθμό για το κάθε ένα. Οι προβλέψιμες προσεγγίσεις είναι οι πιο παραδοσιακές και είναι αυτές που αναπτύχθηκαν πρώτα. Τα μεγάλα συστήματα όμως χρειάζονται ενάμιση με τρία χρόνια ανάπτυξης και στο τέλος το παραγόμενο αποτέλεσμα συνήθως διαφέρει αισθητά από αυτό που είχαν στο νου τους οι πελάτες. Για το λόγο αυτό, η διαδικασία

ανάπτυξης ενός σύγχρονου πληροφορικού συστήματος πήρε άλλη μορφή λιγότερο προβλέψιμη και περισσότερο προσαρμοστική και αντικειμενοστραφής με τα επιμέρους στάδια του κύκλου ανάπτυξης να επαναλαμβάνονται κυκλικά είτε αυτά αναφέρονται σε ολόκληρο το έργο είτε σε επιμέρους τμήματά του τα οποία εμπλουτίζονται συνεχώς.

2.3. Κλασσικές προβλέψιμες προσεγγίσεις του Κύκλου ζωής Ανάπτυξης Συστήματος

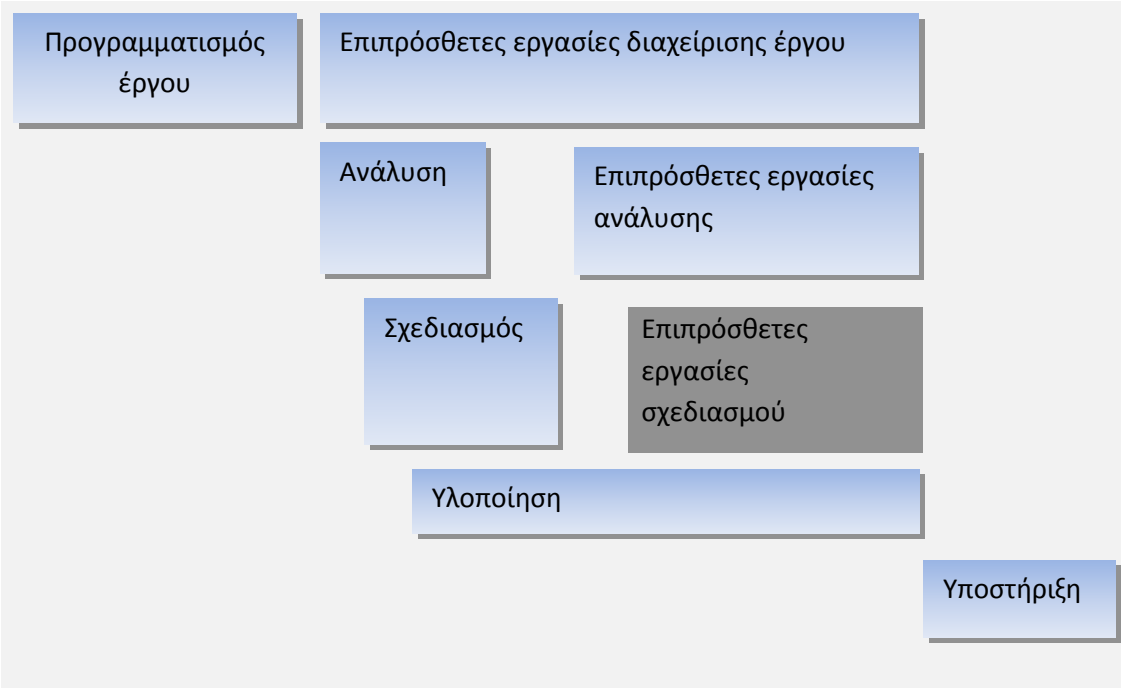
Η ανάπτυξη ενός νέου πληροφοριακού συστήματος συνεπάγεται την εκτέλεση πολλών διαφορετικών δραστηριοτήτων. Οι δραστηριότητες αυτές δεν είναι ανεξάρτητες μεταξύ τους, αλλά συσχετίζονται δηλαδή αποτελούν η μία συνέχεια της άλλης. Στις κλασσικές προβλέψιμες προσεγγίσεις ξεκινάμε αρχικά με τις *δραστηριότητες προγραμματισμού έργου* όπου σχεδιάζουμε, οργανώνουμε και προγραμματίζουμε το έργο. Στην φάση αυτή η επιχείρηση ή ο οργανισμός αναγνωρίζει ότι έχει ένα πρόβλημα να επιλύσει. Οι δραστηριότητες προγραμματισμού έργου χαράζουν τη συνολική δομή του έργου. Στη συνέχεια, μια ομάδα των δραστηριοτήτων που ονομάζονται *δραστηριότητες ανάλυσης* επικεντρώνονται στην κατανόηση του προβλήματος που πρέπει να επιλυθεί καθώς και στον προσδιορισμό των απαιτήσεων. Ο σκοπός είναι να καταλάβουμε ακριβώς τι πρέπει να κάνει το σύστημα για να υποστηρίξει τις λειτουργίες της επιχείρησης ή του οργανισμού. Μια τρίτη ομάδα δραστηριοτήτων εστιάζεται στον σχεδιασμό του νέου συστήματος δηλαδή στον λεπτομερή σχεδιασμό της επίλυσης. Οι δραστηριότητες αυτές, που ονομάζονται *δραστηριότητες σχεδιασμού*, βασίζονται στις απαιτήσεις που έχουν οριστεί νωρίτερα για να σχεδιάσουν τη δομή του προγράμματος και τους αλγόριθμους για το νέο πληροφοριακό σύστημα. Ακολουθούν οι *δραστηριότητες υλοποίησης* που κατασκευάζουν το νέο σύστημα που επιλύει το πρόβλημα. Κύριο μέρος των δραστηριοτήτων υλοποίησης είναι ο προγραμματισμός, ο έλεγχος και η εγκατάσταση του νέου συστήματος.

Αυτές οι τέσσερις παραπάνω ομάδες δραστηριοτήτων του προγραμματισμού, της ανάλυσης, του σχεδιασμού και της υλοποίησης, αναφέρονται συχνά ως **φάσεις (phases)** ή **στάδια** και συνθέτουν το πλαίσιο διαχείρισης του έργου. Μια άλλη φάση, που ονομάζεται *φάση υποστήριξης*, περιλαμβάνει εκείνες τις δραστηριότητες που απαιτούνται για την αναβάθμιση, ενίσχυση και συντήρηση του συστήματος μετά την ολοκλήρωσή του με σκοπό το νέο σύστημα να συνεχίσει να παρέχει τα προβλεπόμενα οφέλη. Η φάση υποστήριξης αποτελεί μέρος του συνολικού Κύκλου Ζωής Ανάπτυξης Συστήματος. Δεν θεωρείται όμως ότι αποτελεί μέρος του αρχικού σχεδίου ανάπτυξης λόγω της φύσης των δραστηριοτήτων που εμπεριέχει. Το παρακάτω σχήμα (βλέπε Σχήμα 2.4) απεικονίζει τα πέντε στάδια του κλασσικού Κύκλου Ζωής Ανάπτυξης Συστήματος.



***Σχήμα 2.4**: Φάσεις Ανάπτυξης Πληροφοριακού Συστήματος*

Ένα χαρακτηριστικό παράδειγμα της πιο κλασσικής προβλέψιμης προσέγγισης αποτελεί το μοντέλο καταρράκτη, όπου πριν μεταβούμε από το ένα στάδιο στο επόμενο πρέπει να έχει ολοκληρωθεί πλήρως το προηγούμενο στάδιο. Δεν υπάρχει δυνατότητα επιστροφής στο προηγούμενο στάδιο και εκ νέου αναπροσαρμογή. Το μοντέλο καταρράκτη αποτελεί ένα άκαμπτο σύστημα που προϋποθέτει την έλλειψη ανθρώπινων λαθών. Αν παραλείψουμε να κάνουμε κάτι, τότε δεν μπορούμε να ξαναγυρίσουμε πίσω και να το διορθώσουμε. Από το μοντέλο καταρράκτη οι σύγχρονες προσεγγίσεις ανάπτυξης κρατούν μόνο τις πέντε φάσεις ανάπτυξης αλλά όχι όμως την αυστηρά σειριακή διαδρομή εκτέλεσης αυτών. Προτείνουν ένα τροποποιημένο μοντέλο καταρράκτη με τον κατακερματισμό και την εν μέρει επικάλυψη των πέντε φάσεων του Κύκλου Ζωής Ανάπτυξης Συστήματος, όπως φαίνεται ενδεικτικά στο παρακάτω σχήμα (βλέπε Σχήμα 2.5).



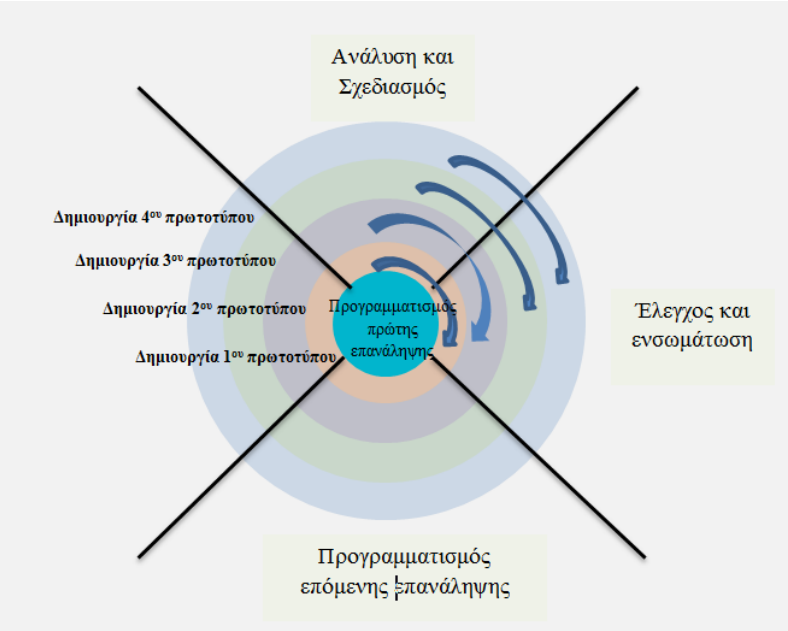
***Σχήμα 2.5**: Επικάλυψη των φάσεων του συστήματος ανάπτυξης*

Η επικάλυψη αφορά ένα μέρος μόνο των διαφόρων φάσεων, διότι η μία φάση εξαρτάται από την άλλη. Δεν μπορούμε για παράδειγμα να προχωρήσουμε εις βάθος στον σχεδιασμό χωρίς να έχουμε κατανοήσει πλήρως την φύση του προβλήματος. Άρα, ένα μέρος της ανάλυσης πρέπει να προηγηθεί πρώτα πριν περάσουμε στον σχεδιασμό. Και αυτός ο σχεδιασμός δεν θα είναι πλήρης. Θα αφορά μόνο το αντίστοιχο τμήμα της ανάλυσης που έχει ολοκληρωθεί προηγουμένως.

Από την άλλη πλευρά όμως, λόγω της αλληλεξάρτησης μεταξύ των συνιστωσών του συστήματος, ένα μέρος της ανάλυσης και του σχεδιασμού μπορούν να συμβαδίζουν ταυτόχρονα, όπως συμβαίνει στην περίπτωση της ανάλυσης των αναγκών των χρηστών. Δηλαδή, για να μπορέσουμε να αντιληφθούμε καλύτερα τι θέλουν οι χρήστες από το νέο σύστημα, ο σχεδιασμός ενός μέρος του συστήματος μπορεί να διενεργείται παράλληλα με την ανάλυση.

2.4. Ευέλικτες – Προσαρμοστικές προσεγγίσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος

Σε αντίθεση με τις προβλέψιμες προσεγγίσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος, οι προσαρμοστικές προσεγγίσεις υποθέτουν ότι οι δραστηριότητες που περιλαμβάνονται στην ανάπτυξη του συστήματος πρέπει να αναπροσαρμόζονται κάθε φορά και να ακολουθούν την εξέλιξη του έργου. Αυτό είναι απαραίτητο επειδή οι πτυχές του έργου δεν είναι πλήρως κατανοητές από την αρχή. Μια πρώιμη έκδοση του προσαρμοστικού Κύκλου Ζωής Ανάπτυξης Συστήματος αποτελεί το **σπειροειδές μοντέλο**, όπως φαίνεται στο παρακάτω σχήμα (βλέπε Σχήμα 2.6). Το σπειροειδές μοντέλο είναι μία επαναληπτική διαδικασία που τερματίζει με την ολοκλήρωση του έργου. Σε κάθε επανάληψη κάνουμε τα ίδια πράγματα ξεκινώντας όμως κάθε φορά από διαφορετική βάση και έχοντας ως στόχο να εμπλουτίσουμε το παραγόμενο προϊόν της προηγούμενης επανάληψης.



***Σχήμα 2.6**: Ο σπειροειδές κύκλος ανάπτυξης*

Στο τέλος κάθε κύκλου ή επανάληψης παράγεται ένα πρωτότυπο ως ένα προκαταρκτικό μοντέλο που δείχνει μία πτυχή του συστήματος. Για κάθε πρωτότυπο, η αναπτυξιακή διαδικασία ακολουθεί μια σειριακή διαδρομή μέσα από την ανάλυση, τον σχεδιασμό, την υλοποίηση, τον έλεγχο, την ενσωμάτωση με προηγούμενες συνιστώσες πρωτότυπων και τον προγραμματισμό για το επόμενο πρωτότυπο. Με την ολοκλήρωση του προγραμματισμού για το επόμενο πρωτότυπο ο κύκλος των δραστηριοτήτων αρχίζει ξανά.

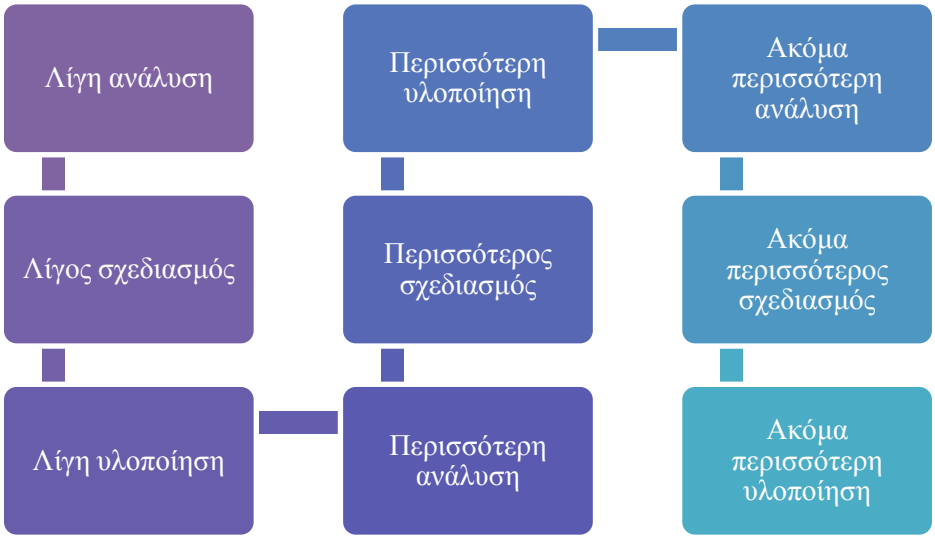
Το παράδειγμα στο Σχήμα 2.6 αποτελεί μία από τις υλοποιήσεις της σπειροειδής προσέγγισης με την δημιουργία τεσσάρων πρωτοτύπων. Αρχίζει με μια αρχική φάση προγραμματισμού, όπως φαίνεται στο κέντρο της σπείρας. Ο σκοπός της αρχικής αυτής φάσης είναι να συγκεντρωθούν αρκετές πληροφορίες για να μπορέσει να ξεκινήσει η ανάπτυξη του πρώτου πρωτότυπου. Η αρχική φάση προγραμματισμού των δραστηριοτήτων περιλαμβάνει την μελέτη σκοπιμότητας, την έρευνα σχετικά με τις απαιτήσεις των χρηστών, την παραγωγή εναλλακτικών προτάσεων εφαρμογής καθώς και την επιλογή του συνολικού σχεδιασμού και της στρατηγικής υλοποίησης. Μετά τον αρχικό προγραμματισμό η ομάδα ανάπτυξης αναλαμβάνει την δημιουργία του πρώτου πρωτότυπου και ακολουθούν στη συνέχεια και τα υπόλοιπα πρωτότυπα μέχρι την ολοκλήρωση του έργου.

Κατά την εξέλιξη του έργου εμφανίζονται μία σειρά από κίνδυνοι οι οποίοι αν δεν αντιμετωπιστούν έγκαιρα είναι δυνατό να τορπιλίσουν την επιτυχή έκβαση του έργου. Σε κάθε κύκλο ανάπτυξης ή επανάληψης καταγράφονται οι κίνδυνοι που εμπεριέχουν οι διαδικασίες και οι εναλλακτικές λύσεις όπου υπάρχουν. Κατά την πρώτη επανάληψη εστιάζουμε σε εκείνον τον τομέα ο οποίος φαίνεται να παρουσιάζει τους μεγαλύτερους κινδύνους, όπως για παράδειγμα η εφικτότητα της εφαρμογής της νέας τεχνολογίας. Στην περίπτωση αυτή η πρώτη επανάληψη θα μπορούσε να επικεντρωθεί σε ένα πρωτότυπο που αποδεικνύει ότι η συγκεκριμένη τεχνολογία μπορεί να λειτουργήσει έτσι όπως έχει σχεδιαστεί. Στη συνέχεια η δεύτερη επανάληψη θα μπορούσε να εστιαστεί στην δημιουργία ενός πρωτότυπου που θα διευθετεί τους κινδύνους που σχετίζονται με τις απαιτήσεις του συστήματος. Για μία άλλη εφαρμογή ο μεγαλύτερος κίνδυνος θα μπορούσε να ήταν η αποδοχή της αλλαγής εκ μέρους των χρηστών. Έτσι, η πρώτη επανάληψη θα μπορούσε να επικεντρωθεί στην δημιουργία ενός πρωτότυπου που θα αποδεικνύει την βελτίωση του ψηφιακού εργασιακού περιβάλλοντος με την εφαρμογή του νέου συστήματος.

Στο σπειροειδές μοντέλο χρησιμοποιείται ο όρος επανάληψη. Στην επίλυση προβλημάτων, οι επαναλήψεις χρησιμοποιούνται για να διαιρέσουμε ένα πολύ μεγάλο και σύνθετο πρόβλημα σε απλούστερα υποπροβλήματα τα οποία μπορούμε να διαχειριστούμε πιο εύκολα. Η επίλυση κάθε υποπροβλήματος οδηγεί στην επίλυση του αρχικού προβλήματος.

Η ανάπτυξη πληροφοριακών συστημάτων χρησιμοποιεί την επανάληψη για τον ίδιο σκοπό. Παίρνουμε ένα μεγάλο σύστημα και το κατακερματίζουμε σε μικρότερες συνιστώσες. Στη συνέχεια προγραμματίζουμε, αναλύουμε, σχεδιάζουμε και

υλοποιούμε κάθε μικρότερη συνιστώσα. Το τελευταίο βήμα είναι αυτό της ολοκλήρωσης όπου συνδυάζουμε τις μικρότερες συνιστώσες σε μια ολοκληρωμένη λύση. Η προσέγγιση αυτή συχνά ονομάζεται **επαναληπτική προσέγγιση (iterative approach)** του Κύκλου Ζωής Ανάπτυξης Συστήματος. Πολλές από τις πιο δημοφιλείς προσαρμοστικές προσεγγίσεις χρησιμοποιούν σήμερα την επανάληψη ως το θεμελιώδης στοιχείο της προσέγγισης. Το σχήμα 2.7 απεικονίζει πώς λειτουργεί μια επαναληπτική προσέγγιση.



***Σχήμα 2.7**: Οι δραστηριότητες ανάπτυξης συστήματος επαναλαμβάνονται*

Επανάληψη σημαίνει ότι οι δραστηριότητες του έργου ανάπτυξης, όπως η ανάλυση, ο σχεδιασμός και η υλοποίηση επαναλαμβάνονται συνεχώς. Ο αριθμός των επαναλήψεων εξαρτάται από την πολυπλοκότητα του έργου. Με κάθε επανάληψη, τα μέλη της ομάδας ανάπτυξης βελτιώνουν το αποτέλεσμα, έτσι ώστε να είναι πιο κοντά σε αυτό που τελικά χρειαζόμαστε. Η επανάληψη βασίζεται στην παραδοχή ότι δεν είναι δυνατό να πετύχουμε το σωστό αποτέλεσμα με την πρώτη φορά. Αρχικά η ανάλυση, ο σχεδιασμός και η υλοποίηση είναι μικρής έκτασης. Τα πρώτα αποτελέσματα όμως που παίρνουμε μας δίνουν την δυνατότητα να γνωρίζουμε αν πραγματικά το σύστημα θα λειτουργήσει και θα πετύχει τους στόχους του. Στη συνέχεια αναλύοντας, σχεδιάζοντας και υλοποιώντας κάθε φορά όλο και μεγαλύτερο τμήμα μπορούμε να κάνουμε βελτιώσεις. Όσο προχωράει η επαναληπτική αυτή διαδικασία τόσο πλησιάζουμε όλο και πιο κοντά στους στόχους που έχουμε θέσει και ταυτόχρονα μειώνουμε τους κινδύνους που μπορούν να εμφανιστούν κατά την ανάπτυξη όπως είναι η αλλαγή της τεχνολογίας.

Μπορούμε να οργανώσουμε τις επαναλήψεις με διάφορους τρόπους. Μία προσέγγιση είναι να ορίσουμε τις πιο βασικές λειτουργίες που πρέπει το σύστημα να περιλαμβάνει και στη συνέχεια να τις υλοποιήσουμε στην πρώτη επανάληψη. Με την ολοκλήρωση των βασικών αυτών λειτουργιών του συστήματος υλοποιούμε στη συνέχεια τις επόμενες και λιγότερο κρίσιμες λειτουργίες του συστήματος. Τέλος, οι

προαιρετικές λειτουργίες του συστήματος, για τις οποίες λέμε «καλό είναι να υπάρχουν» υλοποιούνται στην τελευταία επανάληψη. Μια άλλη προσέγγιση είναι να επικεντρωνόμαστε κάθε φορά σε ένα υποσύστημα. Η υλοποίηση του πρώτου υποσυστήματος περιέχει βασικές λειτουργίες και τα δεδομένα από τα οποία εξαρτώνται τα άλλα υποσυστήματα. Στη συνέχεια, η επόμενη επανάληψη περιλαμβάνει ένα πρόσθετο υποσύστημα και ούτω καθεξής.

Μερικές φορές οι επαναλήψεις ορίζονται σύμφωνα με την πολυπλοκότητα ή τον κίνδυνο ορισμένων συνιστωσών. Συχνά, οι πιο σύνθετες ή υψηλού κινδύνου συνιστώσες του συστήματος διευθετούνται πρώτα για να μπορέσουν οι στόχοι και οι αρχικές απαιτήσεις να αλλάξουν έγκαιρα χωρίς τεράστιες συνέπειες για την ανάπτυξη του έργου. Άλλες φορές, μερικά από τα απλούστερα συστατικά του συστήματος αντιμετωπίζονται πρώτα για να έχουμε όσο είναι δυνατό γρηγορότερα μεγαλύτερο μέρος του συστήματος ολοκληρωμένο. Το πώς οι επαναλήψεις ορίζονται, εξαρτάται από πολλούς παράγοντες και μπορεί να είναι διαφορετικές κάθε φορά σύμφωνα με το υπό ανάπτυξη έργο. Οι περισσότερες προσαρμοστικές προσεγγίσεις δίνουν προτεραιότητα στην αντιμετώπιση των πιο δύσκολων προβλημάτων με τον υψηλότερο κίνδυνο.

Μια σχετική προσέγγιση, η οποία είναι ένας τύπος επαναληπτικής προσέγγισης είναι αυτή της **αυξητικής ανάπτυξης (incremental development)**. Με την προσέγγιση αυτή, ολοκληρώνονται μέρη του συστήματος σε λίγες επαναλήψεις και τίθεται σε λειτουργία το σύστημα για τους χρήστες έτσι ώστε να μπορούν να επωφεληθούν από αυτό άμεσα. Στη συνέχεια με την μέθοδο των επαναλήψεων αναπτύσσεται ένα άλλο μέρος του συστήματος, το οποίο διασυνδέεται με το πρώτο μέρος και τίθεται ο τελικός συνδυασμός πάλι σε λειτουργία. Τέλος, ολοκληρώνεται το τελευταίο μέρος του συστήματος και ενσωματώνεται με τα υπόλοιπα. Σήμερα, οι περισσότερες διαδικασίες ανάπτυξης συστήματος χρησιμοποιούν ποικίλους βαθμούς επανάληψης. Μία από αυτές που περιγράφεται στην συνέχεια είναι η αντικειμενοστραφής προσέγγιση που χαρακτηρίζεται ως άκρως επαναληπτική.

2.5. Στάδια του Κύκλου ζωής Ανάπτυξης Συστήματος

Στις προηγούμενες ενότητες περιγράψαμε γενικά τις διάφορες φάσεις του Κύκλου Ζωής Ανάπτυξης Συστήματος που είναι ο προγραμματισμός, η ανάλυση, ο σχεδιασμός, η υλοποίηση και η υποστήριξη και εξηγήσαμε πώς οι δραστηριότητες της κάθε φάσης διενεργούνται επαναληπτικά. Στη συνέχεια, θα παρουσιάσουμε αναλυτικά τις δραστηριότητες της κάθε φάσης του Κύκλου Ζωής Ανάπτυξης Συστήματος.

Προγραμματισμός έργου

Οι πρωταρχικοί στόχοι του προγραμματισμού έργου είναι να προσδιορίσουμε το πεδίο εφαρμογής του νέου συστήματος, να διασφαλίσουμε ότι το έργο αυτό είναι εφικτό και υλοποιήσιμο, να σχεδιάσουμε τον χρονοπρογραμματισμό και τους πόρους

που θα χρειαστούμε και τέλος να κοστολογήσουμε το έργο. Οι δραστηριότητες που περιλαμβάνονται στον προγραμματισμό του έργου είναι οι παρακάτω:

- Καθορισμός του προβλήματος
- Χρονοπρογραμματισμός του έργου
- Επιβεβαίωση δυνατότητας υλοποίησης του έργου
- Στελέχωση του έργου
- Αρχικοποίηση του έργου

Η πιο σημαντική δραστηριότητα του προγραμματισμού του έργου είναι ο ακριβής προσδιορισμός του προβλήματος και το πεδίο εφαρμογής της λύσης του προβλήματος. Σε αυτό το στάδιο του έργου, δεν γνωρίζουμε όλες τις λειτουργίες ή τις διαδικασίες που θα πρέπει να συμπεριληφθούν στο σύστημα. Ωστόσο, είναι σημαντικό να εντοπίσουμε τις κύριες χρήσεις του νέου συστήματος και τα προβλήματα της επιχείρησης ή του οργανισμού που θα κληθεί το νέο σύστημα να αντιμετωπίσει.

Δραστηριότητες Ανάλυσης

Ο πρωταρχικός στόχος των δραστηριοτήτων ανάλυσης είναι να κατανοήσουμε και να τεκμηριώσουμε τις ανάγκες και τις απαιτήσεις του νέου συστήματος. Η ανάλυση είναι ουσιαστικά μια ανακαλυπτική διαδικασία. Οι λέξεις κλειδιά που χαρακτηρίζουν τις δραστηριότητες της ανάλυσης είναι η *ανακάλυψη* και η *κατανόηση*. Οι δραστηριότητες που περιλαμβάνονται σε αυτό το στάδιο είναι οι παρακάτω:

- Συλλογή πληροφοριών
- Προσδιορισμός απαιτήσεων συστήματος
- Κατασκευή πρωτοτύπων για την διερεύνηση των απαιτήσεων
- Ιεράρχηση απαιτήσεων
- Δημιουργία και αξιολόγηση εναλλακτικών λύσεων
- Επιλογή εναλλακτικής λύσης

Δραστηριότητες σχεδιασμού

Ο στόχος των δραστηριοτήτων σχεδιασμού είναι να σχεδιάσουμε το σύστημα με βάση τις καθορισμένες απαιτήσεις και τις αποφάσεις που έχουν ληφθεί κατά τη διάρκεια της ανάλυσης. Ο υψηλού επιπέδου σχεδιασμός συνίσταται στην ανάπτυξη μια αρχιτεκτονικής δομής για τα στοιχεία του λογισμικού, των βάσεων δεδομένων, των διεπαφών χρήστη και του λειτουργικού περιβάλλοντος. Ο σχεδιασμός χαμηλού επιπέδου συνεπάγεται την ανάπτυξη των λεπτομερών αλγορίθμων και δομών δεδομένων που είναι απαραίτητα για την ανάπτυξη λογισμικού. Επτά κύριες δραστηριότητες πρέπει να ολοκληρωθούν κατά τη διάρκεια της φάσης σχεδιασμού:

- Σχεδιασμός και ολοκλήρωση του δικτύου
- Σχεδιασμός της αρχιτεκτονικής της εφαρμογής

- Σχεδιασμός των διεπαφών χρήστη
- Σχεδιασμός των διασυνδέσεων συστήματος
- Σχεδιασμός και ενσωμάτωση της βάσης δεδομένων
- Δημιουργία πρωτοτύπου
- Σχεδιασμός και ενσωμάτωση των ελέγχου συστήματος

Δραστηριότητες Υλοποίησης

Με τις δραστηριότητες υλοποίησης παραδίδεται το τελικό σύστημα και είναι έτοιμο να χρησιμοποιηθεί. Στόχος δεν είναι μόνο να παραχθεί ένα αξιόπιστο, πλήρως λειτουργικό πληροφοριακό σύστημα, αλλά και να διασφαλίσει ότι οι όλοι χρήστες του συστήματος έχουν εκπαιδευτεί και η επιχείρηση ή ο οργανισμός είναι έτοιμοι να επωφεληθούν, όπως αναμενόταν από τη χρήση του συστήματος. Όλες οι προηγούμενες δραστηριότητες συγκλίνουν και οδηγούν στην δημιουργία ενός λειτουργικού πληροφοριακού συστήματος. Πέντε είναι οι κύριες δραστηριότητες που συνθέτουν το στάδιο υλοποίησης:

- Κατασκευή συνιστωσών λογισμικού
- Έλεγχος και δοκιμή
- Μετατροπή δεδομένων
- Εκπαίδευση χρηστών και τεκμηρίωση συστήματος
- Εγκατάσταση συστήματος

Δραστηριότητες Υποστήριξης

Ο στόχος των δραστηριοτήτων υποστήριξης είναι να διατηρήσουν το σύστημα σε πλήρη λειτουργία στο πέρασμα του χρόνου μετά την αρχική εγκατάσταση του. Το νέο σύστημα εγκαθίσταται και τίθεται σε λειτουργία. Από το σημείο εκείνο ξεκινούν οι δραστηριότητες υποστήριξης οι οποίες διαρκούν όσο και η «παραγωγική ζωή» του συστήματος. Οι περισσότερες επιχειρήσεις ή οργανισμοί προσδοκούν η «παραγωγική ζωή» του νέου συστήματος να έχει μεγάλη διάρκεια. Κατά την διάρκεια της υποστήριξης, αναβαθμίσεις ή βελτιώσεις είναι δυνατόν να διενεργηθούν με σκοπό την επέκταση των δυνατοτήτων του συστήματος. Αυτές οι αναβαθμίσεις ή βελτιώσεις αναπτύσσονται σύμφωνα με τον Κύκλο Ζωής Ανάπτυξης Συστήματος. Οι τρεις κύριες δραστηριότητες που λαμβάνουν χώρα κατά τη διάρκεια της υποστήριξης είναι οι παρακάτω:

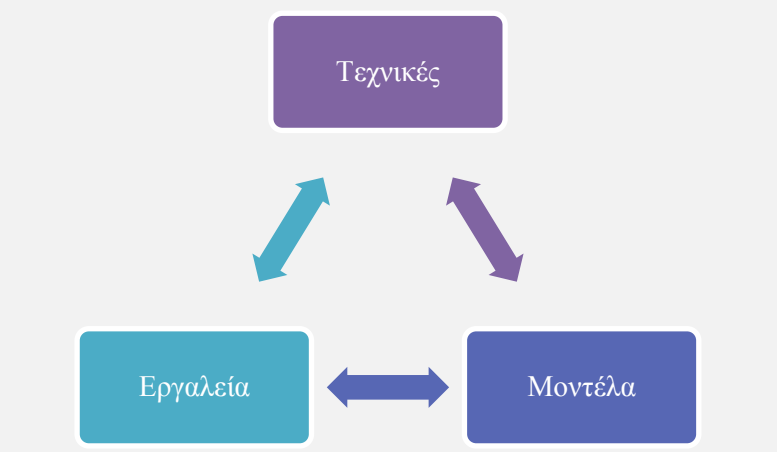
- Διατήρηση του συστήματος
- Βελτίωση του συστήματος
- Υποστήριξη χρηστών

2.6. Μεθοδολογίες, μοντέλα, εργαλεία και τεχνικές

Οι αναλυτές συστημάτων έχουν μια ποικιλία από βοηθήματα στην διάθεσή τους που τους βοηθούν να ολοκληρώσουν τις δραστηριότητές και τα καθήκοντα τους. Μεταξύ αυτών είναι οι μεθοδολογίες, τα μοντέλα, τα εργαλεία και οι τεχνικές.

Μεθοδολογίες

Μια μεθοδολογία ανάπτυξης συστήματος παρέχει οδηγίες για την ολοκλήρωση της κάθε δραστηριότητας στον Κύκλο Ζωής Ανάπτυξης Συστήματος και περιλαμβάνει συγκεκριμένες τεχνικές, εργαλεία και μοντέλα τα οποία συνδέονται μεταξύ τους όπως φαίνεται στο παρακάτω σχήμα (βλέπε Σχήμα 2.8).



***Σχήμα 2.8**: Σχέσεις μεταξύ συστατικών μιας μεθοδολογίας*

Επειδή μια μεθοδολογία περιέχει οδηγίες για το πώς θα χρησιμοποιήσουμε τα μοντέλα, τα εργαλεία και τις τεχνικές, ας προσπαθήσουμε να καταλάβουμε πρώτα τι είναι τα μοντέλα, τα εργαλεία και οι τεχνικές. Τα μοντέλα, τα εργαλεία και οι τεχνικές είναι αυτά που θα σας βοηθήσουν να διαχειριστείτε αποτελεσματικά το έργο που έχετε αναλάβει.

Μοντέλα (Models)

Κάθε φορά που θέλουμε να καταγράψουμε ή να επικοινωνήσουμε πληροφορίες σε οποιοδήποτε πλαίσιο, είναι πολύ χρήσιμο να δημιουργήσουμε ένα μοντέλο. Ένα μοντέλο στην ανάπτυξη πληροφοριακών συστημάτων έχει τον ίδιο σκοπό με οποιοδήποτε άλλο μοντέλο. Ένα μοντέλο είναι μια αναπαράσταση μιας σημαντικής πτυχής του πραγματικού κόσμου. Μερικές φορές ο όρος **αφαίρεση - γενίκευση (abstraction)** χρησιμοποιείται διότι διαχωρίζουμε από το υπόλοιπο σύνολο αυτό που έχει την μεγαλύτερη σημασία για εμάς και αυτό είναι που κρατάμε. Ας θεωρήσουμε το μοντέλο ενός αεροπλάνου. Αν θέλουμε να μιλήσουμε για την αεροδυναμική του αεροπλάνου, είναι χρήσιμο να έχουμε ένα μικρό μοντέλο που δείχνει το συνολικό σχήμα του αεροπλάνου στις τρεις διαστάσεις. Μερικές φορές ένα γράφημα που δείχνει λεπτομέρειες από την διατομή της πτέρυγας του αεροπλάνου είναι αυτό που χρειαζόμαστε. Σε άλλες περιπτώσεις, μια λίστα των μαθηματικών χαρακτηριστικών

του αεροπλάνου θα μπορούσε να χρησιμοποιηθεί για να κατανοήσουμε την συμπεριφορά του αεροπλάνου. Όλα αυτά είναι μοντέλα του ίδιου αεροπλάνου.

Ορισμένα μοντέλα είναι παρόμοια με το πραγματικό προϊόν. Άλλα μοντέλα αποτελούν γραφικές αναπαραστάσεις σημαντικών λεπτομερειών ή είναι αφηρημένοι μαθηματικοί συμβολισμοί. Κάθε μοντέλο δίνει έμφαση σε ένα διαφορετικό είδος των πληροφοριών. Στο σχεδιασμό του αεροπλάνου, οι μηχανικοί αεροσκαφών χρησιμοποιούν πολλά διαφορετικά μοντέλα. Το να είναι κάποιος μηχανικός αεροσκαφών σημαίνει πως γνωρίζει πώς να δημιουργεί και να χρησιμοποιεί όλα αυτά τα μοντέλα. Το ίδιο ακριβώς ισχύει και με τα μέλη της ομάδας ανάπτυξης. Τα μοντέλα για τα πληροφοριακά συστήματα όμως δεν είναι τόσο τυποποιημένα ή ακριβή. Αυτό οφείλεται στο γεγονός ότι ο τομέας που μελετάει τα μοντέλα των πληροφοριακών συστημάτων γνωρίζει ανάπτυξη τα τελευταία χρόνια καθώς και στο ότι τα πληροφοριακά συστήματα έχουν να κάνουν με καταστάσεις και έννοιες που δεν έχουν την φυσική υπόσταση των αεροπλάνων.

Τι είδους μοντέλα όμως χρησιμοποιούνται στα πληροφοριακά συστήματα; Τα μοντέλα που χρησιμοποιούνται στην ανάπτυξη συστημάτων περιλαμβάνουν αναπαραστάσεις εισόδων εξόδων, διαδικασιών, δεδομένων, αντικείμενων, αλληλεπιδράσεων μεταξύ αντικείμενων, τοποθεσιών, δικτύων και συσκευών μεταξύ άλλων. Τα περισσότερα από τα αυτά είναι γραφικά μοντέλα δηλαδή *διαγράμματα* και *γραφήματα*. Ένα άλλο είδος σημαντικού μοντέλου που αναπτύσσουμε και χρησιμοποιούμε είναι αυτό του σχεδιασμού ή προγραμματισμού του έργου. Στο μοντέλο αυτό παρουσιάζονται οι εργασίες που πρέπει να διεκπεραιωθούν καθώς και οι αντίστοιχες ημερομηνίες ολοκλήρωσης. Στα παραπάνω μοντέλα προστίθεται και το διάγραμμα που δείχνει το σύνολο των ατόμων που ασχολούνται με το έργο.

Εργαλεία (Tools)

Ένα εργαλείο στο πλαίσιο της ανάπτυξης του πληροφοριακού συστήματος είναι ένα υποστηρικτικό λογισμικό που βοηθά στη δημιουργία μοντέλων ή άλλα συνιστωσών που είναι απαραίτητες για την ανάπτυξη του έργου. Τα εργαλεία μπορεί να είναι απλά σχεδιαστικά προγράμματα για τη δημιουργία διαγραμμάτων. Θα μπορούσαν να περιλαμβάνουν μια εφαρμογή βάσης δεδομένων που να αποθηκεύει πληροφορίες αναφορικά με το έργο, όπως είναι οι ορισμοί ροών δεδομένων ή οι γραπτές περιγραφές των διαδικασιών. Ένα λογισμικό διαχείρισης έργου είναι ένα άλλο παράδειγμα εργαλείου που χρησιμοποιείται για να δημιουργήσει μοντέλα για τα καθήκοντα και τις εξαρτήσεις μεταξύ των καθηκόντων αυτών.

Τα εργαλεία έχουν σχεδιαστεί ειδικά για να βοηθήσουν τα μέλη της ομάδας ανάπτυξης του συστήματος. Οι προγραμματιστές θα πρέπει να είναι εξοικειωμένοι με τα ολοκληρωμένα περιβάλλοντα ανάπτυξης (integrated development environments - IDEs) που περιλαμβάνουν πολλά εργαλεία για την υποστήριξη του προγραμματισμού, όπως είναι οι έξυπνοι συντάκτες, η θεματική βοήθεια και τα εργαλεία εκσφαλμάτωσης. Μερικά εργαλεία μπορούν να δημιουργήσουν κώδικα

προγράμματος για τους προγραμματιστές ή να δημιουργήσουν μοντέλα από κώδικα προγράμματος έτσι ώστε ο προγραμματιστής να μπορεί να προσδιορίσει τι κάνει το πρόγραμμα σε περίπτωση που ο φάκελος τεκμηρίωσης απουσιάζει. Εργαλεία οπτικής μοντελοποίησης είναι διαθέσιμα στους αναλυτές συστημάτων για να τους βοηθήσουν να δημιουργήσουν και να επαληθεύσουν μοντέλα συστημάτων καθώς και να δημιουργήσουν κώδικα προγράμματος απευθείας από μοντέλα.

Τεχνικές (Techniques)

Μια τεχνική στην ανάπτυξη συστήματος είναι μια συλλογή από κατευθυντήριες γραμμές που υποστηρίζουν τον αναλυτή στην προσπάθειά του να ολοκληρώσει μια δραστηριότητα. Μια τεχνική μπορεί να περιλαμβάνει οδηγίες βήμα-προς-βήμα για τη δημιουργία ενός μοντέλου ή γενικές συμβουλές για τη συλλογή πληροφοριών από τους χρήστες του συστήματος. Συναντάμε τεχνικές μοντελοποίησης δεδομένων, ελέγχου λογισμικού, συνέντευξης καθώς και σχεδιασμού σχεσιακών βάσεων δεδομένων.

2.7. Δομημένη Ανάπτυξη Συστήματος (Structured Analysis and Design Technique - SADT)

Η δομημένη ανάλυση, η δομημένη σχεδίαση και ο δομημένος προγραμματισμός είναι τρείς τεχνικές που αποτελούν την δομημένη ανάπτυξη. Πολλές φορές αυτές οι τεχνικές αναφέρονται με τα αρχικά των λέξεων **Structured Analysis and Design Technique (SADT).**

Οι τεχνικές του **δομημένου προγραμματισμού** αναπτύχθηκαν στην δεκαετία του 60 και αποτέλεσαν μια προσπάθεια για την παροχή οδηγιών που θα βελτίωναν την ποιότητα των προγραμμάτων. Οι τεχνικές της **δομημένης σχεδίασης** αναπτύχθηκαν την δεκαετία του 70, για να συνδεθούν διαφορετικά προγράμματα σε ένα πληροφοριακό σύστημα, καθώς παρείχαν οδηγίες για το είδος των προγραμμάτων και την ιεράρχησή τους. Για το σκοπό αυτό αναπτύχθηκαν τα **Διαγράμματα Δομής (ΔΔ).** Οι τεχνικές της δομημένης ανάλυσης εξελίχθηκαν στις αρχές της δεκαετίας του 1980 για να διευκρινιστούν οι απαιτήσεις ενός πληροφοριακού συστήματος, πριν οι αναλυτές σχεδιάσουν τα προγράμματα. Τα **Διαγράμματα Ροής Δεδομένων (**ΔΡΔ) αποτελούν τη διαγραμματική αποτύπωση των απαιτήσεων του συστήματος. Από τη φάση της ανάλυσης τα ΔΡΔ μετατρέπονται σε ΔΔ για να χρησιμοποιηθούν στη Σχεδίαση.

Στη δεκαετία του 80, αναπτύχθηκαν τεχνικές σχεδίασης αρχείων και βάσεων δεδομένων. Τα Συστήματα Διαχείρισης βάσεων Δεδομένων χρησιμοποιούνται στην δομημένη σχεδίαση με σκοπό την αλληλεπίδραση των προγραμμάτων με τη βάση δεδομένων και για αυτό το λόγο αναπτύχθηκαν τα **Διαγράμματα Οντοτήτων Συσχετίσεων (ΔΟΣ).** Επιπρόσθετα, κατά την δεκαετία του 80, αναπτύχθηκαν τεχνικές σχεδίασης διεπαφής χρήστη (για παράδειγμα σχεδιασμός menus).

Σύμφωνα με τη δομημένη προσέγγιση το Πληροφοριακό Σύστημα θεωρείται ότι αποτελεί μια ενιαία και σχετικά αυτοτελή, μεγάλη και σύνθετη λειτουργική μονάδα. Πλεονεκτήματα της δομημένης ανάπτυξης αποτελεί η ανεξαρτησία των δεδομένων από το λογισμικό, η ευκολία στην κατανόηση του κώδικα και η συμβατότητα με τον δομημένο προγραμματισμό. Χαρακτηριστικό της δομημένης ανάπτυξης συστήματος είναι η αυστηρή πειθαρχία στην οργάνωση του κώδικα του προγράμματος και η σειριακή εκτέλεση των φάσεων ανάπτυξης που όμως πολλές φορές οδηγεί σε καθυστερήσεις στην υλοποίηση του έργου.

Μειονεκτήματα της δομημένης προσέγγισης

Η δομημένη ανάπτυξη ενός συστήματος με την πάροδο του χρόνου εξελίχθηκε σε αρκετές παραλλαγές. Οι αναλυτές που έμειναν στην αρχικές εκδόσεις δεν χρησιμοποίησαν πολλές βελτιώσεις της, ενώ κάποιοι χρησιμοποίησαν τμήματα των νέων τεχνικών χωρίς να τις κατανοήσουν σε όλη τους την έκταση και να τις μάθουν με όλες τις λεπτομέρειες. Πολλοί πιστεύουν ότι η δομημένη προσέγγιση έχει πολλές αδυναμίες καθώς οι τεχνικές που χρησιμοποιούνται δεν καλύπτουν όλες τις ενέργειες της ανάλυσης και σχεδίασης. Επίσης, πολλοί αναλυτές θεωρούν ότι η μετάβαση από τα Διαγράμματα Ροής Δεδομένων (ΔΡΔ) (δομημένη ανάλυση) στα Διαγράμματα Δομής (δομημένη σχεδίαση) στην πράξη δεν λειτουργεί. Μεγάλο μειονέκτημα, αποτελούν η πολύ αργή ανάπτυξη (κρατά μερικά χρόνια για μεγάλα Π.Σ.) γιατί έχει σειριακό χαρακτήρα (πρέπει πρώτα να τελειώσει η ανάλυση για να αρχίσει η σχεδίαση, κλπ) και οι αλλαγές στις επιχειρηματικές απαιτήσεις που αποτυπώνονται με μεγάλη δυσκολία και δεν επιτρέπουν την επαναχρησιμοποίηση τμημάτων λογισμικού (reuse).

2.8. Αντικειμενοστραφής ανάπτυξη συστήματος

Μια εντελώς διαφορετική προσέγγιση στην ανάπτυξη πληροφοριακών συστημάτων αποτελεί η αντικειμενοστραφής προσέγγιση. Το σύστημα εξετάζεται ως συλλογή από αλληλεπιδρώμενα αντικείμενα τα οποία συνεργάζονται για την πραγματοποίηση ενός έργου. Το σύστημα αποτελείται από αντικείμενα που μπορούν να απαντούν σε μηνύματα. Αυτό οδήγησε στην ανάπτυξη διαφορετικών προσεγγίσεων στην ανάλυση, στον σχεδιασμό και στον προγραμματισμό των συστημάτων. Η προσέγγιση αυτή αναπτύχθηκε όταν στη δεκαετία του 70 άρχισαν να αναπτύσσονται οι αντικειμενοστραφείς γλώσσες προγραμματισμού.

Επειδή η αντικειμενοστραφής προσέγγιση μελετά τα πληροφοριακά συστήματα ως συλλογές από αντικείμενα που αλληλεπιδρούν, η αντικειμενοστραφής ανάλυση **(Object-Oriented Analysis (OOA))** καθορίζει όλα τα είδη των αντικειμένων που αλληλεπιδρούν με το σύστημα και μελετά τις αλληλεπιδράσεις χρήστη που απαιτούνται για την ολοκλήρωση των εργασιών. Ο αντικειμενοστραφής σχεδιασμός (**Object-Oriented design (OOD**)) καθορίζει όλους τους τύπους των αντικειμένων που είναι απαραίτητοι, δείχνει πώς τα αντικείμενα αλληλεπιδρούν για να ολοκληρωθούν οι εργασίες και τελειοποιεί τον ορισμό του κάθε τύπου αντικειμένου

έτσι ώστε να μπορεί να υλοποιηθεί σε μια συγκεκριμένη γλώσσα. Ο Αντικειμενοστραφής Προγραμματισμός (**Object-Oriented Programming (OOP)**) αποτελείται από γραπτές δηλώσεις σε μια γλώσσα προγραμματισμού για να καθορίσει τι κάνει κάθε τύπος αντικειμένου.

Ένα αντικείμενο είναι ένας τύπος πράγματος — ένας πελάτης ή ένας υπάλληλος ή ένας μαθητής, καθώς επίσης και ένα κουμπί ή ένα μενού. Κάποια πράγματα, όπως οι πελάτες, υπάρχουν τόσο έξω από το σύστημα (αληθινός πελάτης) όσο και χωριστά στο εσωτερικό του συστήματος (μια αναπαράσταση στον υπολογιστή του πελάτη). Η αντικειμενοστραφής ανάπτυξη πληροφοριακών συστημάτων χρησιμοποιεί ένα διάγραμμα κλάσεων για την εμφάνιση όλων των κλάσεων των αντικειμένων στο σύστημα.

Η αντικειμενοστραφής προσέγγιση έχει διάφορα βασικά πλεονεκτήματα, μεταξύ των οποίων είναι η φυσικότητα και η επαναχρησιμοποίηση. Η προσέγγιση είναι φυσική — ή διαισθητική — για τους ανθρώπους, επειδή έχουν την τάση να σκέφτονται τον κόσμο ως αντικείμενα. Είναι λιγότερο φυσικό για έναν άνθρωπο να σκεφτεί πολύπλοκες διαδικασίες που αναπτύχθηκαν σε διαδικαστικές γλώσσες προγραμματισμού. Επίσης, επειδή η αντικειμενοστραφής προσέγγιση περιλαμβάνει κλάσεις αντικειμένων και πολλά συστήματα στην οργάνωση τους χρησιμοποιούν τα ίδια αντικείμενα, αυτές οι κλάσεις μπορούν να χρησιμοποιηθούν ξανά και ξανά κάθε φορά που χρειάζονται. Για παράδειγμα, σχεδόν όλα τα συστήματα χρησιμοποιούν μενού, πλαίσια διαλόγου, παράθυρα και κουμπιά, αλλά επίσης και πολλά συστήματα εντός της ίδιας εταιρείας χρησιμοποιούν κλάσεις όπως πελάτης, προϊόν και τιμολόγιο που μπορούν να επαναχρησιμοποιηθούν. Σαφώς, η αντικειμενοστραφής προσέγγιση είναι αρκετά διαφορετική από την δομημένη προσέγγιση.

Πολλά συστήματα που αναπτύσσονται σήμερα συνδυάζουν παραδοσιακές και αντικειμενοστραφείς προσεγγίσεις. Για το λόγο αυτό, είναι σημαντικό η κάλυψη τόσο των παραδοσιακών/δομημένων προσεγγίσεων όσο και των αντικειμενοστραφών προσεγγίσεων.

2.9. Σύγχρονες Τάσεις στην Ανάπτυξη Συστήματος

Ένα πράγμα που δεν αλλάζει ποτέ στο πεδίο των πληροφοριών συστημάτων είναι ότι τα πράγματα αλλάζουν πάντα. Πάντα εμφανίζονται νέα εργαλεία και τεχνικές - μερικές φορές με μεγάλη δημοσιότητα και αναμονή - και οι αναλυτές πάντα ψάχνουν για νέους και καλύτερους τρόπους εργασίας. Οι τεχνικές και οι κύκλοι ζωής που συζητήθηκαν προηγουμένως είναι παραδείγματα των εν εξελίξει αλλαγών στις μεθοδολογίες ανάπτυξης συστημάτων. Σε αυτήν την ενότητα θα παρουσιαστούν μερικές σημαντικές τρέχουσες τάσεις στην ανάπτυξη του συστήματος. Κάθε μία από αυτές τις τάσεις θα μπορούσε να κυριαρχήσει στην ανάπτυξη των συστημάτων στο μέλλον.

2.9.1 Ενοποιημένη διαδικασία (UP)

Η **ενιαία διαδικασία (ΕΔ)** είναι μια αντικειμενοστραφής μεθοδολογία ανάπτυξης συστήματος που προσφέρεται από την εταιρεία λογισμικού IBM και προέρχεται από τούς τρεις υποστηρικτές της **Unified Modeling Language(UML**): Grady Booch, James Rumbaugh και τον Ivar Jacobson. Αποτελεί την προσπάθειά τους να καθορίσουν μια ολοκληρωμένη μεθοδολογία που εκτός από την παροχή πολλών μοναδικών χαρακτηριστικών, χρησιμοποιεί την UML για τα μοντέλα του συστήματος. Αν και θα μάθετε πολλά για την UML επειδή πρόκειται για ένα πρότυπο μοντελοποίησης για την **αντικειμενοστραφή προσέγγιση (ΟΟ),** η UP δεν είναι μία κλασσική ΟΟ μεθοδολογία ανάπτυξης.

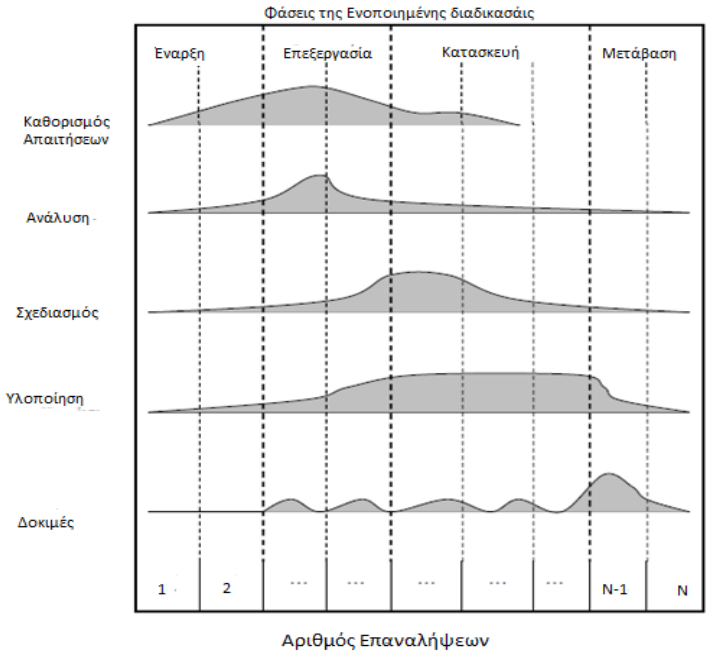
Η UP στοχεύει στην ενίσχυση των έξι "βέλτιστων πρακτικών" ανάπτυξης συστήματος που είναι κοινές σε πολλές μεθοδολογίες ανάπτυξης:

- Επαναληπτική ανάπτυξη
- Ορισμός και διαχείριση απαιτήσεων
- Χρήση συστατικών αρχιτεκτονικής
- Δημιουργία εικονικών μοντέλων
- Έλεγχος ποιότητας
- Έλεγχος αλλαγών

Η UP ορίζει τέσσερις φάσεις του κύκλου ζωής:

- έναρξη
- επεξεργασία
- κατασκευή
- μετάβαση

Ο κύκλος ζωής UP παρουσιάζεται στο σχήμα 2.9.



***Σχήμα 2.9**: Φάσεις της ενοποιημένης διαδικασίας*

Η εναρκτήρια φάση ορίζει το πεδίο εφαρμογής του έργου καθορίζοντας περιπτώσεις χρήσης. Στην επόμενη φάση της επεξεργασίας, καθορίζονται οι περισσότερες από τις περιπτώσεις χρήσης με λεπτομέρειες και σχεδιάζεται η αρχιτεκτονική του συστήματος. Κατά την φάση της κατασκευής το προϊόν κατασκευάζεται, ενώ στη φάση της μετάβασης το προϊόν βρίσκεται σε κατάσταση δοκιμής.

2.9.2 Ακραίος Προγραμματισμός eXtreme Programming (XP)

Ο **ακραίος προγραμματισμός (XP)** είναι μία προσέγγιση ανάπτυξης συστήματος που πρόσφατα διαδόθηκε από τον Kent Beck. Στον XP προσαρμόζονται τεχνικές από πολλές πηγές και προσθέτονται μερικές νέες ιδέες. Μερικές φορές αναφέρεται ως μία "ελαφριά" μεθοδολογία ανάπτυξης συστήματος, δηλαδή διατηρείται απλή και επικεντρώνεται περισσότερο στην αποτελεσματικότητα της διαδικασίας ανάπτυξης. Είναι ένα παράδειγμα μιας σε μεγάλο βαθμό προσαρμοζόμενης προσέγγισης για τον SDLC.

Οι αναλυτές ξεκινούν τον προγραμματισμό του έργου αφήνοντας τους χρήστες να περιγράφουν ιστορίες χρηστών, που είναι παρόμοιες με περιπτώσεις χρήσης. Οι ιστορίες χρηστών είναι περιγραφές για το πώς θέλουν την υποστήριξη από το σύστημα — με άλλα λόγια περιγράφουν την απαιτούμενη λειτουργικότητα του συστήματος. Οι αναλυτές ετοιμάζουν ένα έγγραφο με αυτές τις ιστορίες με άτυπα περιγραφικά μοντέλα. Εκτός από τις ιστορίες του χρήστη, οι χρήστες περιγράφουν ένα σύνολο δοκιμών αποδοχής που θα αποδεικνύουν ότι το σύστημα παρέχει την απαιτούμενη λειτουργικότητα μετά την ολοκλήρωσή του.

Στη συνέχεια οι αναλυτές προγραμματίζουν μια σειρά εκδόσεων του έργου, όπου σε κάθε έκδοση συμπεριλαμβάνεται ένα μέρος του τελικού συστήματος, όπως ισχύει και με την προσέγγιση της σταδιακής ανάπτυξης. Το έργο προχωρά με την πρώτη έκδοση, που συνήθως παίρνει αρκετές επαναλήψεις για να ολοκληρωθεί. Όταν ολοκληρωθεί η πρώτη έκδοση, η δεύτερη έκδοση ξεκινά και το ίδιο συμβαίνει και με τις επόμενες εκδόσεις.

Ο XP είναι λίγο πολύ όπως και οι άλλες επαναληπτικές και επαυξητικές προσεγγίσεις. Όμως, ο XP περιέχει μερικά πρόσθετα χαρακτηριστικά γνωρίσματα που τον καθιστούν δημοφιλή. Για παράδειγμα απαιτεί συνεχή δοκιμή, συνεχή ολοκλήρωση και μεγάλη συμμετοχή των χρηστών. Απαιτεί επίσης, ο προγραμματισμός να γίνεται από ομάδες των δύο προγραμματιστών που εργάζονται μαζί σε ένα σταθμό εργασίας όταν γράφουν και δοκιμάζουν τον κώδικα. Αυτό και άλλα χαρακτηριστικά γνωρίσματα δίνουν έμφαση στην ανοιχτή και αποτελεσματική επικοινωνία μεταξύ των μελών της ομάδας. Ένα τελικό χαρακτηριστικό γνώρισμα είναι ότι οι προγραμματιστές δεν θα πρέπει να εργάζονται περισσότερο από 40 ώρες την εβδομάδα, για να αποδειχθεί ότι το σύστημα μπορεί να ολοκληρωθεί σύμφωνα με το χρονοδιάγραμμα χωρίς να απαιτούνται υπερβολικές ώρες εργασίας.

2.9.3 Μεθοδολογία SCRUM

Η **Scrum** είναι μία νέα μεθοδολογία ευέλικτης/προσαρμοστικής ανάπτυξης. Ο όρος Scrum αναφέρεται στο παιχνίδι του ράγκμπι. Το όνομα δόθηκε καθώς υπάρχουν πολλές ομοιότητες μεταξύ του αθλήματος και της προσέγγισης ανάπτυξης συστήματος. Και οι δύο είναι γρήγορες, προσαρμοστικές και αυτό-ρυθμιζόμενες. Η βασική ιδέα πίσω από την Scrum είναι ότι η μεθοδολογία πρέπει να ανταποκριθεί σε μια τρέχουσα κατάσταση γρήγορα και όσο το δυνατόν θετικότερα.

Η φιλοσοφία του scrum εστιάζεται στην γρήγορη ανταπόκριση σε αλλαγές ενός δυναμικού περιβάλλοντος στο οποίο οι χρήστες δεν γνωρίζουν ακριβώς τι χρειάζεται και μπορεί να αλλάζουν συχνά προτεραιότητες. Σε αυτό το είδος του περιβάλλοντος, οι αλλαγές είναι τόσες πολλές που τελικά το έργο τελματώνει και δεν φτάνει ποτέ σε ολοκλήρωση. Η scrum υπερέχει σε αυτές τις καταστάσεις έναντι των άλλων προσεγγίσεων επειδή επικεντρώνεται κατά κύριο λόγο στην ομάδα ανάπτυξης και στο έργο τους. Δίνει έμφαση στα άτομα περισσότερο από όσο στις διεργασίες και περιγράφει πώς οι ομάδες των προγραμματιστών μπορούν να εργαστούν μαζί για να οικοδομήσουν το λογισμικό σε μια σειρά από σύντομα υποέργα. Κλειδί σε αυτή η φιλοσοφία είναι ότι ασκεί τον απόλυτο έλεγχο μια ομάδα ειδικών.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Δημιουργήστε ένα σχεδιάγραμμα του δωματίου σας και στη συνέχεια προβείτε στην περιγραφή του. Αποτελούν και τα δύο μοντέλα του δωματίου σας; Που συναντάται περισσότερο η ακρίβεια και η λεπτομέρεια; Ποιο από τα δύο θα ακολουθούσε με ευκολία κάποιος που δεν σας έχει ξαναεπισκεφτεί;

2. Περιγράψτε μία «τεχνική» που θα σας βοηθήσει να ολοκληρώσετε την δραστηριότητα «Είμαι στην ώρα μου στο μάθημα». Ποια εργαλεία θα μπορούσατε να χρησιμοποιήσετε;

3. Περιγράψτε μία «τεχνική» που θα σας βοηθήσει να ολοκληρώσετε την δραστηριότητα «Παραδίδω τις εργασίες έγκαιρα». Ποια εργαλεία θα μπορούσατε να χρησιμοποιήσετε;

4. Ποιες άλλες τεχνικές χρησιμοποιείτε για να ολοκληρώσετε τις καθημερινές σας δραστηριότητες;

5. Υπάρχουν τουλάχιστον δύο προσεγγίσεις για την ανάπτυξη συστήματος, μια ποικιλία των κύκλων ζωής ανάπτυξης συστήματος και ένας μακρύς κατάλογος τεχνικών και μοντέλων που χρησιμοποιούνται σε ορισμένες προσεγγίσεις και σε άλλες όχι. Σκεφτείτε γιατί συμβαίνει αυτό. Συζητήστε τους παρακάτω πιθανούς λόγους και προσδιορίστε ποιός είναι πιο σημαντικός τεκμηριώνοντας την άποψή σας: το πεδίο εφαρμογής είναι αρκετά νέο; η τεχνολογία αλλάζει πολύ γρήγορα; διαφορετικοί οργανισμοί έχουν διαφορετικές ανάγκες; υπάρχουν πολλά διαφορετικά είδη συστημάτων; στην ανάπτυξη συστημάτων εμπλέκονται άτομα με διαφορετικά υπόβαθρα;

6. Περιηγηθείτε στο διαδίκτυο για να βρείτε εταιρείες ανάπτυξης λογισμικού. Αναζητήστε την προσέγγιση που ακολουθούν στην ανάπτυξη συστήματος καθώς και τα χρησιμοποιούμενα εργαλεία. Περιγράφουν τον Κύκλο Ζωής Ανάπτυξης Συστήματος; Κάνουν αναφορά σε IDE ή σε εργαλεία οπτικής μοντελοποίησης;

7. Θεωρείστε ότι ξεκινάτε τις σπουδές σας στην Α’ ΕΠΑΛ. Στόχος σας είναι να ολοκληρώσετε τις σπουδές σας στο προβλεπόμενο χρονικό διάστημα αποκτώντας εκείνα τα εφόδια που θα σας βοηθήσουν στην μετέπειτα εξέλιξή σας. Αποφασίζετε να αναπτύξετε μία μεθοδολογία «ολοκλήρωσης της εκπαίδευσης». Στο πλαίσιο αυτό προσπαθείτε να απαντήσετε στα παρακάτω ερωτήματα: Ποια είναι τα στάδια του προσωπικού σας κύκλου ζωής ολοκλήρωσης της εκπαίδευσης; Ποιες δραστηριότητες περιέχονται σε κάθε στάδιο; Ποιες τεχνικές, μοντέλα και εργαλεία θα μπορούσατε να χρησιμοποιήσετε για να υλοποιήσετε τις παραπάνω δραστηριότητες;

8. Σας έχει ανατεθεί η ανάπτυξη του συστήματος αυτοματοποίησης παρακολούθησης των εργασιών που βρίσκονται σε εξέλιξη καθώς και του τελικού αποθέματος εμπορευμάτων. Έρχεστε αντιμέτωποι με μία νέα πρόκληση. Πώς θα αντιμετωπίσουν οι εργαζόμενοι τις νέες αλλαγές; Από την άλλη μεριά χρειάζεστε την συνεργασία των εργαζομένων όσον αφορά το κομμάτι της τεχνολογίας. Για το δεύτερο κομμάτι αυτό της λογιστικής γνωρίζετε αρκετά πράγματα. Μπορείτε να διακρίνετε ποια είναι τα δύο συστήματα που θα κληθείτε να αναπτύξετε; Ποιες παραλλαγές του Κύκλου Ζωής Ανάπτυξης Συστήματος είναι πιο κατάλληλες για το υπάρχον πλαίσιο; Θα επιλέγατε τις προσεγγίσεις UP, XP ή Scrum; Σε ποιες

δραστηριότητες της ανάλυσης και του σχεδιασμού θα έπρεπε να συμμετέχουν οι εργαζόμενοι και τα στελέχη;

9. Επιλέξτε ένα επιχειρησιακό έργο της προτίμησής σας για να το μηχανογραφήσετε. Πώς θα μπορούσατε να χρησιμοποιούσατε τα στάδια του Κύκλου Ζωής Ανάπτυξης Συστήματος για να το υλοποιήσετε. Παραθέστε ενδεικτικά παραδείγματα.

10. Ποια από τις δύο προσεγγίσεις για την ανάπτυξη του συστήματος είναι η πιο πρόσφατη;

11. Δώστε μερικά χαρακτηριστικά της ενοποιημένης διαδικασίας (UM).

12. Συζητήστε τα πλεονεκτήματα του ακραίου προγραμματισμού (XP).

13. Δώστε μερικά χαρακτηριστικά της προσέγγισης Scrum

Μελέτη Περίπτωσης

Σχολική Βιβλιοθήκη

Η Βιβλιοθήκη του σχολείου σας είναι αρκετά μεγάλη, έχει πολλά βιβλία σχολικά, λογοτεχνικά, ιστορικά, ποιήματα κ.α. Παρόλα αυτά δεν σας έχει απασχολήσει ποια βιβλία περιλαμβάνονται καθώς δεν είναι εύκολη η πρόσβαση σε αυτά. Σκέφτεστε ότι μια σχολική βιβλιοθήκη στην σύγχρονη εποχή θα έπρεπε να περιλαμβάνει και e-books, e-magazines και γενικότερα να υπάρχει πρόσβαση μέσω εφαρμογών στα περιεχόμενα βιβλίων, στους τίτλους των βιβλίων καθώς και δυνατότητα δανεισμού. Θα ήταν ωραίο όταν χρησιμοποιείτε τον υπολογιστή ή το κινητό σας να μπορείτε να βρείτε το βιβλίο που χρειάζεστε, να μάθετε περισσότερες πληροφορίες για αυτό και να έχετε τη δυνατότητα να το δανειστείτε. Επίσης, να μπορείτε να κατεβάσετε τμήματα ή και ολόκληρα ψηφιακά βιβλία που χρειάζεστε στα μαθήματά σας.

Ώρα να ξεκινήσετε την ανάλυση του έργου. Είστε οι **αναλυτές του συστήματος** που θα υλοποιηθεί. Σκεφτείτε αυτά που μάθατε σε αυτό το κεφάλαιο (τον κύκλο ζωής λογισμικού) και αναλογιστείτε τις ενέργειες που πρέπει να γίνουν. Χωριστείτε σε ομάδες. Κάθε ομάδα θα αναλάβει να επεξεργαστεί μια προσέγγιση ανάπτυξης συστήματος από αυτές που μάθατε. Παρουσιάστε τις προτάσεις σας στις υπόλοιπες ομάδες. Στη συνέχεια η ολομέλεια της τάξης θα αποφασίσει ποια προσέγγιση θα ακολουθηθεί στην ανάπτυξη τους συστήματος σχολικής βιβλιοθήκης.

Βιβλιογραφία

Στα Ελληνικά

Βεσκούκης, Β.(2000) *Τεχνολογία Λογισμικού Ι*. Πάτρα: Ε.Α.Π.

Στα Αγγλικά

O'Brien A. J. & Marakas. G. (2008). *Introduction to Information Systems*, 14th/edition, New York: McGraw-Hill.

Satzinger, J. W. & Jackson B. R., Burd D. S. (2009). *Systems Analysis and Design in a Changing World*, Boston: MA: Thomson Course Technology.

Ανάλυση Απαιτήσεων και Καθορισμός Προδιαγραφών

Περιεχόμενα

3.1 Κατηγορίες απαιτήσεων
3.1.1 Η απαίτηση
3.1.2 Η απαίτηση από το λογισμικό
3.1.3 Λειτουργικές και μη λειτουργικές απαιτήσεις
3.1.4 Οι απαιτήσεις του πελάτη
3.2 Διαδικασία προσδιορισμού απαιτήσεων
3.2.1 Εξαγωγή απαιτήσεων
3.3 Μοντέλα ανάλυσης απαιτήσεων
3.3.1 Δομημένη προσέγγιση
3.3.2 Αντικειμενοστραφής προσέγγιση
3.4 Καθορισμός Προδιαγραφών

Διδακτικοί Στόχοι

- Να περιγράφουν τα βήματα που γίνονται για την προδιαγραφή απαιτήσεων.
- Να κατατάσσουν σε κατηγορίες τις απαιτήσεις ενός πληροφοριακού συστήματος.
- Να χρησιμοποιούν τεχνικές για την εξαγωγή απαιτήσεων, τη μοντελοποίηση και την προδιαγραφή απαιτήσεων.
- Να δημιουργούν ένα έγγραφο προδιαγραφής απαιτήσεων.

3.1 Κατηγορίες απαιτήσεων

Στο 2ο κεφάλαιο περιγράφηκε ο κύκλος ζωής ανάπτυξης ενός συστήματος. Η ανάλυση απαιτήσεων είναι η πρώτη διαδικασία της ανάπτυξης ενός συστήματος και είναι ανεξάρτητη από το μοντέλο του κύκλου ζωής που ακολουθείται. Αποτελεί μια πολύ κρίσιμη διεργασία καθώς εκφράζει την προσπάθεια κατανόησης των αναγκών και των απαιτήσεων του πελάτη. Η έλλειψη οργανωμένης και υπεύθυνης προσπάθειας στην ανάλυση και στον καθορισμό των απαιτήσεων είναι σίγουρο ότι θα οδηγήσει σε προβλήματα που θα εμφανιστούν σε μεταγενέστερα στάδια του κύκλου ζωής και θα κοστίσουν σε χρήματα και σε ώρες εργασίας.

3.1.1 Η απαίτηση

Απαίτηση είναι η περιγραφή μιας υπηρεσίας που θα πρέπει να παρέχει ένα σύστημα, μιας διεργασίας που θα πραγματοποιεί ή μιας συνθήκης που θα πρέπει να ικανοποιεί. Η απαίτηση απαντάει στο ερώτημα «τι κάνει το σύστημα;» και δεν περιλαμβάνει τον τρόπο που θα γίνει αυτό. Αποτελεί μια προσπάθεια αποσαφήνισης του προβλήματος που στοχεύει να λύσει το πληροφοριακό σύστημα που θα δημιουργηθεί και όχι στην περιγραφή κάποιας λύσης. Η απαίτηση καταγράφεται και τεκμηριώνεται με τέτοιο τρόπο ώστε να είναι κατανοητή από όλα τα ενδιαφερόμενα μέρη στην ανάπτυξη του συστήματος. Επιπλέον η απαίτηση είναι μια σημαντική σταθερά που παραμένει ως ένας παράγοντας αξιολόγησης καθ ‘όλη τη διάρκεια του κύκλου ζωής ενός προϊόντος. Είναι δηλαδή ένας είδος συμβολαίου που ελέγχεται ακόμα και μετά το τέλος της ανάπτυξης του προϊόντος.

Απαίτηση για ένα σύστημα ηλεκτρονικής βιβλιοθήκης 1*: Το σύστημα θα παρουσιάζει τον κατάλογο των βιβλίων στον χρήστη και θα του δίνει τη δυνατότητα να αναζητήσει ένα βιβλίο με βάση τον τίτλο ή τον συγγραφέα.*

3.1.2 Η απαίτηση από το λογισμικό

Ένα πληροφοριακό σύστημα αποτελείται από τρία μέρη: τον άνθρωπο, το υλικό και το λογισμικό. Ο άνθρωπος είναι ο χρήστης που αλληλεπιδρά και χειρίζεται το σύστημα ή ο κατασκευαστής του συστήματος. Το υλικό είναι οι υπολογιστές και οι συσκευές που το απαρτίζουν ενώ το λογισμικό είναι οι εφαρμογές που το καθιστούν λειτουργικό και ρυθμίζουν τη συμπεριφορά του. Έτσι ο όρος απαίτηση από το πληροφοριακό σύστημα αφορά την απαίτηση από το σύστημα στην ολότητά του ή από οποιοδήποτε από τα μέρη του.

Ωστόσο το βασικό θεμέλιο ενός πληροφοριακού συστήματος είναι το λογισμικό καθώς ουσιαστικά είναι αυτό που προσαρμόζεται στις ανάγκες του χρήστη και καθορίζει την υπόσταση του συστήματος. Ο ρόλος αυτός του λογισμικού αντικατοπτρίζεται πλήρως και στις απαιτήσεις που το αφορούν. Έτσι ο καθορισμός των απαιτήσεων από το λογισμικό αποτελεί κατά κανόνα την πιο κρίσιμη και εξέχουσα εργασία. Εξάλλου οι απαιτήσεις από τον άνθρωπο και το υλικό συνήθως σχετίζονται έμμεσα ή άμεσα με τις απαιτήσεις του λογισμικού.

3.1.3 Λειτουργικές και μη λειτουργικές απαιτήσεις

Οι απαιτήσεις διακρίνονται σε δυο μεγάλες κατηγορίες. Στις λειτουργικές και στις μη λειτουργικές.

Οι **λειτουργικές απαιτήσεις** περιγράφουν αναλυτικά την αλληλεπίδραση του συστήματος και του περιβάλλοντος. Πολλές φορές οι λειτουργικές απαιτήσεις αποδίδουν την έννοια του επεξεργαστή πληροφορίας σε αυτό που περιγράφουν. Έτσι αναφέρονται στην είσοδο ή αλλιώς στα ερεθίσματα που μπορεί να δεχτεί το σύστημα και στον τρόπο που αντιδρά σε αυτά τα ερεθίσματα. Το σύστημα μπορεί να διενεργεί κάποιες επεξεργασίες ή να μεταβάλλει την κατάστασή του. Μετά την επεξεργασία προκύπτει η έξοδος του συστήματος δηλαδή τα επιθυμητά αποτελέσματα που παράγει σαν απόκριση στο ερέθισμα - είσοδο που δέχθηκε.

Οι απαιτήσεις που δεν αναφέρονται σε κάποια λειτουργία του συστήματος ονομάζονται **μη λειτουργικές.** Οι μη λειτουργικές απαιτήσεις περιγράφουν κάποιες προδιαγραφές που πρέπει να έχει το σύστημα που ουσιαστικά πλαισιώνουν τις λειτουργικές απαιτήσεις. Είναι πολύ σημαντικές καθώς θέτουν κάποιους περιορισμούς στις επιλογές που έχουν οι κατασκευαστές στα στάδια σχεδιασμού και υλοποίησης. Οι μη λειτουργικές απαιτήσεις αναλύονται επιμέρους σε κατηγορίες. Οι πιο σημαντικές κατηγορίες είναι η ασφάλεια, η απόδοση, η αξιοπιστία, η χρηστικότητα και η υποστήριξη.

Απαίτηση για ένα σύστημα ηλεκτρονικής βιβλιοθήκης 2*: Το σύστημα θα πρέπει να ικανοποιεί όλες τις προδιαγραφές ασφαλείας για την προστασία των χρηστών και του συστήματος. Θα πρέπει να απαγορεύεται η πρόσβαση σε μη εξουσιοδοτημένους χρήστες και οι εξουσιοδοτημένοι χρήστες θα πρέπει να έχουν διαφορετικά επίπεδα πρόσβασης και ανάλογα με το επίπεδο να έχουν συγκεκριμένα δικαιώματα. Τα προσωπικά δεδομένα των χρηστών θα πρέπει να φυλάσσονται στη βάση δεδομένων. Ανά τακτά χρονικά διαστήματα ο διαχειριστής του συστήματος θα πρέπει να δημιουργεί αντίγραφο ασφαλείας που θα αποθηκεύει σε ασφαλή τοποθεσία για να ελαχιστοποιήσει την απώλεια δεδομένων σε περίπτωση αστοχίας υλικού ή επιθέσεων από χάκερς.*

Ερώτηση: Τι είδος απαίτησης αποτελούν η απαίτηση 1 και η απαίτηση 2 του συστήματος ηλεκτρονικής βιβλιοθήκης που αναφέρονται παραπάνω;

3.1.4 Οι απαιτήσεις του πελάτη

Ο πελάτης είναι ο πρώτος που εντοπίζει ένα πρόβλημα και εκτιμά ότι υπάρχει ανάγκη λύσης του. Το πρόβλημα μπορεί να αφορά τον τρόπο με τον οποίο γίνονται κάποια πράγματα καθημερινά ή ένα παλαιότερο πληροφοριακό σύστημα που θα πρέπει να επεκταθεί ή να αντικατασταθεί. Ο πελάτης δεν είναι πάντα και χρήστης του συστήματος.

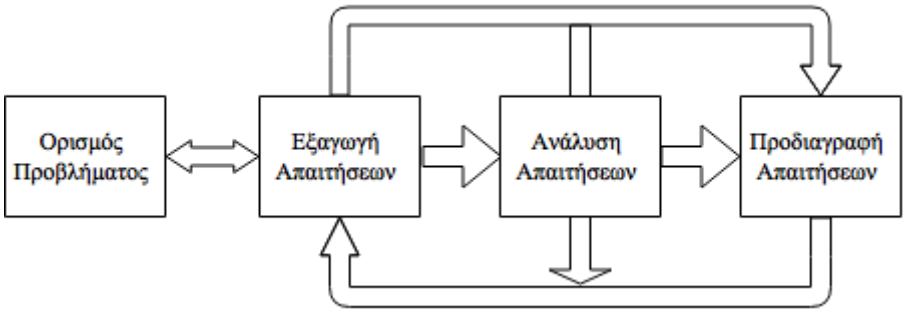
Ο πελάτης χρησιμοποιεί την φυσική γλώσσα για να περιγράψει το πρόβλημα και τις υπηρεσίες που θα παρέχει το νέο πληροφοριακό σύστημα καθώς και τις συνθήκες που θα πρέπει να ικανοποιεί. Ουσιαστικά αποτελεί το πρώτο από τα ενδιαφερόμενα μέρη

που περιγράφει τις απαιτήσεις από το σύστημα και το λογισμικό. Οι απαιτήσεις του πελάτη βέβαια διαφέρουν πάρα πολύ από τις απαιτήσεις του συστήματος. Συχνά πολυάριθμες απαιτήσεις εκφράζονται σαν μία απαίτηση και συγχέονται οι λειτουργικές με τις μη λειτουργικές. Πολλές φορές οι απαιτήσεις περιγράφονται σε συντομία, δεν είναι ακριβείς και εμπεριέχουν πολλές ασάφειες. Επίσης οι πελάτες τείνουν από νωρίς να προτείνουν κάποιες λύσεις και υποδείξεις σε θέματα σχεδιασμού του συστήματος.

Παράδειγμα απαίτησης πελάτη: «Για να είναι το σύστημα εύκολο στη χρήση για εμάς θα πρέπει να διαβάζει τα στοιχεία των βιβλίων από το αρχείο Excel που έχουμε ήδη φτιάξει. Δηλαδή θα ανεβάσετε το αρχείο excel στην ιστοσελίδα μας στο Internet και όταν μπαίνει ο μαθητής θα βλέπει τα βιβλία από το Excel και θα διαλέγει όποιο θέλει. Όταν θέλουμε να προσθέσουμε κάποιο βιβλίο θα μπαίνουμε στην ιστοσελίδα και θα το αλλάζουμε απευθείας στο αρχείο του Excel». Η παραπάνω απαίτηση διατυπώθηκε από ένα πελάτη στη διαδικασία συνέντευξης. Πως την κρίνετε;

3.2 Διαδικασία προσδιορισμού απαιτήσεων

Η διαδικασία του προσδιορισμού απαιτήσεων αποτελεί ένα δύσκολο έργο. Η κρισιμότητα αυτής της εργασίας έχει οδηγήσει στη δημιουργία και εφαρμογή μια σειράς διακριτών βημάτων που φαίνονται στο σχήμα 3.1.



Σχήμα 3.1: *Τα βήματα της διαδικασίας προσδιορισμού απαιτήσεων*

Ο διαχωρισμός της διαδικασίας του προσδιορισμού απαιτήσεων σε επιμέρους δραστηριότητες διευκολύνει την εκτέλεσή της. Η πραγματοποίηση κάθε βήματος προϋποθέτει την καταγραφή πληροφοριών και την τελική παραγωγή του **εγγράφου προδιαγραφής απαιτήσεων**. Μέχρι την τελική παραγωγή αυτής της έκθεσης τα βήματα που φαίνονται στο σχήμα 3.1 επαναλαμβάνονται πολλές φορές. Έτσι επιτυγχάνεται η αναθεώρηση και η σταδιακή εκλέπτυνση των απαιτήσεων που ορίστηκαν στην αρχή της διαδικασίας. Στη συνέχεια παρουσιάζονται συνοπτικά οι δραστηριότητες που εκτελούνται σε κάθε βήμα.

3.2.1 Εξαγωγή απαιτήσεων

Στην δραστηριότητα της εξαγωγής απαιτήσεων οι αναλυτές του συστήματος επικοινωνούν με τα ενδιαφερόμενα μέρη. Η επικοινωνία αυτή έχει στόχο την

ανάλυση της υφιστάμενης κατάστασης ώστε να βγάλουν οι αναλυτές τα δικά τους συμπεράσματα και την αποσαφήνιση της εκδοχής των απαιτήσεων των πελατών. Τα ενδιαφερόμενα μέρη είναι ο πελάτης και όλοι οι χρήστες του συστήματος. Συχνά ο πελάτης και οι χρήστες εκφράζουν με ένα δικό τους τρόπο τι περιμένουν από το νέο σύστημα. Οι αναλυτές εργάζονται με τους ενδιαφερόμενους για να εκμαιεύσουν πληροφορίες για να κατανοήσουν το πρόβλημα, να καταλάβουν τις ανάγκες που ωθούν στη δημιουργία του νέου συστήματος και τις προσδοκίες των ενδιαφερομένων από αυτό. Γι' αυτό το σκοπό οργανώνουν συναντήσεις με σαφής ξεκάθαρους στόχους και συνεντεύξεις με σαφείς και στοχευμένες ερωτήσεις.

Όπως έχει ήδη αναφερθεί τα λάθη που θα γίνουν σε αυτό το στάδιο θα κοστίσουν σε χρόνο και σε χρήμα. Οι αναλυτές θα πρέπει να είναι ιδιαίτερα προσεκτικοί και να διασφαλίζουν ότι τα στοιχεία που εκμαιεύουν από τους πελάτες είναι και τα σωστά. Έτσι χρησιμοποιούν τεχνικές για να ελέγξουν τις πληροφορίες που λαμβάνουν από τους ενδιαφερομένους Για παράδειγμα απευθύνουν το ίδιο ερώτημα με διαφορετικό τρόπο ή διατύπωση στη συνέντευξη και ελέγχουν αν θα πάρουν την ίδια απάντηση. Ομοίως απευθύνουν το ίδιο ερώτημα σε όλες τις ενδιαφερόμενες μεριές και εξετάζουν τις αντιλήψεις που υπάρχουν για ένα συγκεκριμένο θέμα.

Βασική προϋπόθεση για τη σωστή εκτέλεση αυτής της φάσης είναι η άριστη συνεργασία και επικοινωνία ανάμεσα στον αναλυτή και στους ενδιαφερομένους Ωστόσο πολλές φορές η καλή επικοινωνία δεν είναι πλήρως εφικτή. Οι αναλυτές είναι αναμενόμενο να μην έχουν ιδιαίτερες γνώσεις πάνω στον τομέα που δραστηριοποιείται ο ενδιαφερόμενος ούτε να γνωρίζουν την ορολογία που χρησιμοποιείται σε αυτό το πεδίο. Κρίνεται λοιπόν σκόπιμο να προετοιμαστούν κατάλληλα πριν αρχίσει ο κύκλος επαφών με τον πελάτη και τους χρήστες του συστήματος. Θα πρέπει να ενημερωθούν και να αποκτήσουν κάποιες ειδικές γνώσεις πάνω στον συγκεκριμένο τομέα. Έτσι θα πετύχουν βαθύτερη κατανόηση του υπάρχοντος συστήματος ενώ θα δημιουργηθεί ένα καλύτερος κώδικας επικοινωνίας.

Συμπληρωματικά οι αναλυτές μπορούν να χρησιμοποιήσουν την παρατήρηση για να μελετήσουν την υπάρχουσα κατάσταση. Η υπάρχουσα κατάσταση μπορεί να είναι μια χειρωνακτική εργασία που πρέπει να γίνει πιο αυτοματοποιημένη ή ένα παλιό σύστημα που δεν είναι πλέον λειτουργικό ή υπάρχει ανάγκη επέκτασης. Μέσα από την παρατήρηση και την καταγραφή του τρόπου λειτουργίας υπάρχει η δυνατότητα καλύτερης κατανόησης του συστήματος και διάγνωσης ενδεχόμενων δυσλειτουργιών και δυσκολιών.

Στα αρχικά στάδια της εξαγωγής απαιτήσεων οι αναλυτές επιχειρούν να αποκρυσταλλώσουν τον τρόπο που λειτουργεί η υπάρχουσα κατάσταση για να κατανοήσουν τις απαιτήσεις και τις ανάγκες του πελάτη. Έτσι κάνουν τις εξής διεργασίες:

- Δημιουργία μιας λίστας με τους κανόνες που διέπουν την υφιστάμενη κατάσταση:

Κανόνες βιβλιοθήκης

- Δικαίωμα για δανεισμό έχουν μόνο οι μαθητές και οι εκπαιδευτικοί
- Ο κάθε δικαιούχος δεν μπορεί να έχει χρεωμένα πάνω από τέσσερα βιβλία
- Ο κάθε δικαιούχος έχει δικαίωμα να κρατήσει ένα βιβλίο ως και δύο εβδομάδες. Αν δεν υπάρχει εκδήλωση ενδιαφέροντος από άλλους χρήστες για το συγκεκριμένο βιβλίο τότε μπορεί να ανανεώσει τον δανεισμό.
- Όταν παρατηρούνται επανειλημμένα καθυστερήσεις στην επιστροφή βιβλίων τότε γίνονται συστάσεις από τους υπεύθυνους της βιβλιοθήκης και από τη διεύθυνση του σχολείου.
- Ένας δικαιούχος μπορεί να κάνει κράτηση ένα βιβλίο που έχει δανειστεί κάποιος άλλος.
- Ένας δικαιούχος που έχει κάνει κράτηση ένα βιβλίο, ειδοποιείται από τους υπεύθυνους της βιβλιοθήκης όταν αυτό επιστραφεί στη βιβλιοθήκη και έχει προτεραιότητα έναντι άλλων ενδιαφερομένων.

- Καταγραφή των κύριων εννοιών του συστήματος και των χαρακτηριστικών τους:

Έννοιες

Βιβλίο: Κάθε βιβλίο χαρακτηρίζεται από τον «Αύξων αριθμό», τον «Συγγραφέα», τον «Εκδότη» και την «Έκδοση».

Δανειζόμενος: Είναι ο μαθητής ή ο εκπαιδευτικός που έχει δικαίωμα να δανειστεί από τη βιβλιοθήκη. Περιγράφεται από το «Ονοματεπώνυμο», τον «Αύξων αριθμό», το «Τηλέφωνο» και το «Τμήμα».

Βιβλίο δανεισμών: Αναγράφονται οι δανεισμοί των δικαιούχων. Κάθε καταχώρηση περιγράφεται από το όνομα του δανειζόμενου, το όνομα του βιβλίου, την ημερομηνία του δανεισμού, την ημερομηνία επιστροφής και τις παρατηρήσεις.

Καρτέλα βιβλίου: Είναι μία καρτέλα που αντιστοιχεί σε ένα συγκεκριμένο βιβλίο. Στην καρτέλα βιβλίου αναγράφονται πληροφορίες για το βιβλίο, αν είναι δανεισμένο ή όχι, η ημερομηνία δανεισμού, το όνομα του δανειζόμενου, η ημερομηνία επιστροφής και κάποιες παρατηρήσεις.

Βιβλίο κρατήσεων: Αναγράφονται οι κρατήσεις των δικαιούχων. Κάθε καταχώρηση περιγράφεται από το όνομα του δανειζόμενου, το όνομα του βιβλίου και την ημερομηνία κράτησης.

- Καταγραφή σεναρίων που περιγράφουν με λεπτομέρειες τις λειτουργίες του συστήματος:

Περιγραφή λειτουργιών

Λειτουργία του δανεισμού

Για να γίνει ένας δανεισμός ακολουθούνται δύο βήματα: ο έλεγχος και η διαδικασία του δανεισμού. Κατά τον έλεγχο ο υπεύθυνος της βιβλιοθήκης βρίσκει το όνομα και τον αύξων αριθμό του δικαιούχου και του βιβλίου αντίστοιχα. Στη συνέχεια ελέγχει στο βιβλίο δανεισμών τον αριθμό των βιβλίων που έχει ο δικαιούχος και στο βιβλίο κρατήσεων αν υπάρχει κράτηση για το βιβλίο ή όχι. Αν ο αριθμός βιβλίων που έχει ο δικαιούχος δεν υπερβαίνει τον αριθμό βιβλίων που επιτρέπεται να έχει και δεν υπάρχει κράτηση για το βιβλίο τότε μπορεί να προχωρήσει στο βήμα του δανεισμού. Σε αυτό το βήμα καταγράφεται ο δανεισμός στο βιβλίο και στην καρτέλα του βιβλίου.

Κατά τη διάρκεια αυτής φάσης επικαιροποιούν και εμπλουτίζουν διαρκώς αυτές τις πληροφορίες αξιοποιώντας την ανατροφοδότηση που λαμβάνουν στις συναντήσεις με τους ενδιαφερόμενους και τα ευρήματα των παρατηρήσεών τους και την ανάλυσης. Επίσης διαμορφώνουν μια αρχική άποψη για την δομή του συστήματος σε υψηλό επίπεδο χωρίς όμως να προχωρούν σε ιδιαίτερες λεπτομέρειες.

Παράλληλα με την συλλογή και καταγραφή των πληροφοριών που αναφέρθηκαν προηγουμένως ο αναλυτής προσδιορίζει τις απαιτήσεις. Δημιουργεί μία λίστα με τον τρόπο που τις περιγράφει ο πελάτης και συνεχίζει την εργασία του με τον έλεγχο, τον εντοπισμό αντιφάσεων και την αναθεώρησή τους. Στο τελικό στάδιο της διαδικασίας οι αναλυτές ομαδοποιούν και ταξινομούν αυτές τις απαιτήσεις.

Άσκηση 1: *Για το σύστημα που περιγράφεται πως σκέφτεστε ότι μπορεί να είναι η λειτουργία της κράτησης; Να την περιγράψετε.*

Άσκηση 2: *Από τις πληροφορίες που έχετε ως τώρα για το σύστημα μπορείτε να εντοπίσετε κάποια κατάσταση που κρίνεται προβληματική; Πως θα μπορούσε να λυθεί με την ανάπτυξη της νέας εφαρμογής;*

Σημείωση: *Σκεφθείτε αν καταχωρείται πολλές φορές η ίδια πληροφορία, αν οι χρήστες του συστήματος βρίσκουν εύκολα την πληροφορία που ψάχνουν και αν γίνεται εύκολα αντιληπτό ένα λάθος ή μια παράλειψη.*

3.3 Μοντέλα ανάλυσης απαιτήσεων

Υπάρχουν πολλοί τρόποι ορισμού των απαιτήσεων. Ο πιο απλός τρόπος είναι στη φυσική γλώσσα με τη χρήση απλών προτάσεων. Ωστόσο αυτός ο τρόπος έχει ως βασικό μειονέκτημα τον κίνδυνο οι απαιτήσεις να μη γίνουν αντιληπτές από όλα τα ενδιαφερόμενα μέρη με τον ίδιο τρόπο. Αυτό συμβαίνει λόγω ασαφειών στο κείμενο ή χρήση ορολογίας που δεν είναι πλήρως αντιληπτή από όλους. Γι' αυτό το λόγο

δημιουργήθηκαν οι διαγραμματικοί τρόποι περιγραφής του συστήματος και αποτύπωσης των απαιτήσεων του. Τα μοντέλα αυτά περιγράφουν το λογισμικό του συστήματος από διαφορετικές οπτικές γωνίες. Έτσι κάποια μοντέλα περιγράφουν τη λειτουργία του συστήματος από την οπτική του χρήστη, άλλα δίνουν έμφαση στον τρόπο διαχείρισης των δεδομένων και άλλα περιγράφουν τον τρόπο που αλλάζει η κατάσταση του συστήματος ως συνάρτηση της εισόδου που δέχεται. Τα μοντέλα αυτά, εκτός από την οπτική που υιοθετούν απέναντι στο σύστημα, χωρίζονται και με βάση την προσέγγιση που ακολουθούν. Έτσι κάποια ακολουθούν τη δομημένη προσέγγιση ενώ άλλα ακολουθούν την αντικειμενοστραφή προσέγγιση. Οι διαφορές των δύο αυτών προσεγγίσεων αποτυπώνονται στον παρακάτω πίνακα.

Πινάκας 3.1. *Διαφορές της Δομημένης και της αντικειμενοστραφούς προσέγγισης*

Δομημένη προσέγγιση	Αντικειμενοστραφής προσέγγιση
Το σύστημα είναι μια συλλογή διαδικασιών	Το σύστημα είναι μια συλλογή αντικειμένων που αλληλεπιδρούν
Οι διαδικασίες αλληλεπιδρούν με δεδομένα	Τα αντικείμενα αλληλεπιδρούν με τους ανθρώπους και μεταξύ τους
Οι διαδικασίες δέχονται εισόδους και παράγουν εξόδους	Τα αντικείμενα στέλνουν και δέχονται μηνύματα.

3.3.1 Δομημένη προσέγγιση

Σκοπός της δομημένης ανάλυσης είναι η πλήρης κατανόηση του προβλήματος. Μόνο όταν το πρόβλημα καθοριστεί ακριβώς είναι δυνατή η εγγύηση της αποτελεσματικότητας του συστήματος. Αν οι απαιτήσεις του συστήματος δεν είναι κατανοητές τότε πιθανότατα η ανάπτυξη λογισμικού θα δώσει λύση σε ένα άλλο πρόβλημα, όχι αυτό του πελάτη.

Οι τεχνικές της δομημένης ανάλυσης βοηθούν τον αναλυτή στο να ορίσει τι ακριβώς πρέπει να κάνει το σύστημα, ποια είναι τα δεδομένα που θα αποθηκευτούν, ποιες είσοδοι και ποιές έξοδοι απαιτούνται και ποιες διαδικασίες απαρτίζουν το σύστημα. Για αυτό το λόγο αναπτύχθηκαν διαγραμματικές τεχνικές όπως τα Διαγράμματα Ροής Δεδομένων (ΔΡΔ), οι πίνακες αποφάσεων/ δένδρα αποφάσεων, τα διαγράμματα οντοτήτων συσχετίσεων και τα λεξικά δεδομένων.

Διαγράμματα Ροής Δεδομένων (ΔΡΔ)

Το Διάγραμμα Ροής Δεδομένων (ΔΡΔ), αποτελεί την γραφική απεικόνιση της συγκεκριμένης σχεδίασης. Η ανάλυση προχωρά από πάνω προς τα κάτω (Top Down) για να οδηγήσει σε πληρέστερη και λεπτομερέστερη αναπαράσταση του συστήματος.

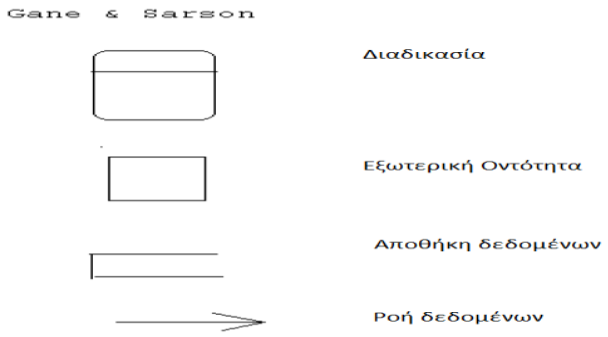
- **Ένα ΔΡΔ παρέχει στοιχεία για** :
    - Τη διάσπαση ενός συστήματος σε υποσυστήματα
    - Τις ροές δεδομένων στο σύστημα
    - Τα δεδομένα εισόδου (Input Data), και τα δεδομένα εξόδου (Output Data) και την εισαγωγή τους σε αρχεία αποθήκευσης
    - Τις πηγές και τους προορισμούς του συστήματος
- **Ένα ΔΡΔ <u>δεν</u> παρέχει στοιχεία για :**
    - Τις αποφάσεις που λαμβάνονται στο σύστημα
    - Τις επαναληπτικές διαδικασίες και τους υπολογισμούς του συστήματος

Τα ΔΡΔ βοηθούν στην κατανόηση της λογικής του συστήματος, μέσω της γραφικής απεικόνισης των διαδικασιών και της ροής των πληροφοριών σε ένα Π.Σ. Κατά συνέπεια τα ΔΡΔ αναπαριστούν ένα Π.Σ. ως ένα σύνολο εξωτερικών οντοτήτων, ροών δεδομένων, διαδικασιών και χώρων αποθήκευσης δεδομένων Το ΔΡΔ είναι ένα γραφικό μοντέλο και για αυτό το λόγο εύκολα αναγνώσιμο.

Οι εξωτερικές οντότητες

Οι εξωτερικές οντότητες είναι στοιχεία που αποτελούν τις πηγές ή τους προορισμούς των ροών των δεδομένων (terminators). Συνήθως είναι εκτός του συστήματός μας (πελάτες, προμηθευτές, τράπεζες, μαθητές, καθηγητές κλπ). Έχουν ένα όνομα (ουσιαστικό) και αποστέλλουν δεδομένα προς το σύστημα ή δέχονται δεδομένα από αυτό, και οποιαδήποτε σχέση μεταξύ τους δεν αφορά την συγκεκριμένη σχεδίαση. Τα σύμβολα που χρησιμοποιούνται σε ένα Διάγραμμα Ροής Δεδομένων παρουσιάζονται στον Πίνακα 3.2.

Πίνακας 3.2. *Απεικονίσεις συμβόλων ενός Διαγράμματος Ροής Δεδομένων*

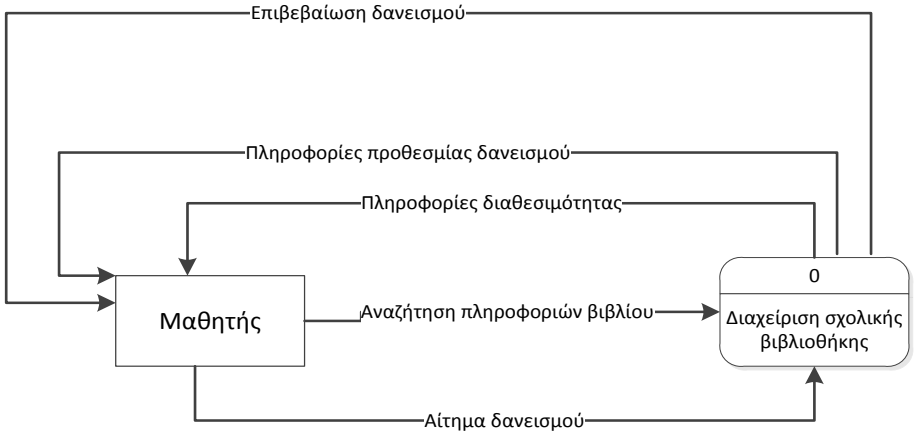


Οι διαφορετικές απεικονίσεις σχετίζονται με την μεθοδολογία που χρησιμοποιήθηκε διαχρονικά για την απεικόνιση του ΔΡΔ. Στην παρούσα ενότητα θα χρησιμοποιήσουμε την απεικόνιση Gane and Sarson.

Διαδικασίες

Μια διαδικασία έχει όνομα που συνήθως αποτελείται από μια ενέργεια και ένα αντικείμενο όπου η ενέργεια επιδρά για παράδειγμα «Μετέφερε Κωδικό». Η διαδικασία επεξεργάζεται δεδομένα και παράγει επεξεργασμένα δεδομένα ή πληροφορίες. Συνήθως μια διαδικασία κάνει μια μεμονωμένη και αυτόνομη επεξεργασία. Όταν η επεξεργασία μας έχει πολύπλοκες και σύνθετες διεργασίες, αναλύεται σε απλούστερες.

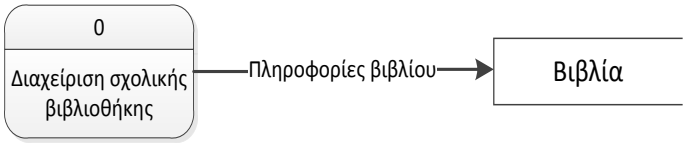
Το όνομα των δεδομένων είναι διαφορετικό από αυτό των δεδομένων εξόδου, η δε επιλογή ονόματος της διαδικασίας πρέπει να είναι προσεκτική για να αποτυπώνει ακριβώς την λειτουργία που εκτελεί. Στο Σχήμα 3.2, παρουσιάζεται η Λειτουργία Δανεισμού Βιβλίου σε μια σχολική βιβλιοθήκη. Για παράδειγμα, η εξωτερική οντότητα **Μαθητής** αναζητά πληροφορίες ενός βιβλίου από τη διαδικασία **Διαχείριση σχολικής βιβλιοθήκης**. Η διαδικασία αυτή επεξεργάζεται τα δεδομένα και επιστρέφει στην οντότητα Μαθητής πληροφορίες διαθεσιμότητας καθώς και πληροφορίες προθεσμίας δανεισμού του βιβλίου



Σχήμα 3.2: *Μηδενικό επίπεδο ΔΡΔ*

Αποθήκες Δεδομένων

Είναι χώροι αποθήκευσης δεδομένων από το σύστημα (βιβλία. μαθητές, προϊόντα, πελάτες), και αποτελούν το Σύστημα Βάσεων Δεδομένων του συστήματος. Επικοινωνούν μόνο με διαδικασίες, από τις οποίες λαμβάνουν και αποστέλλουν δεδομένα. Δεν είναι οντότητες επεξεργασίας δεδομένων, αλλά απλά αποθηκευτικά μέσα (Σχήμα 3.3).



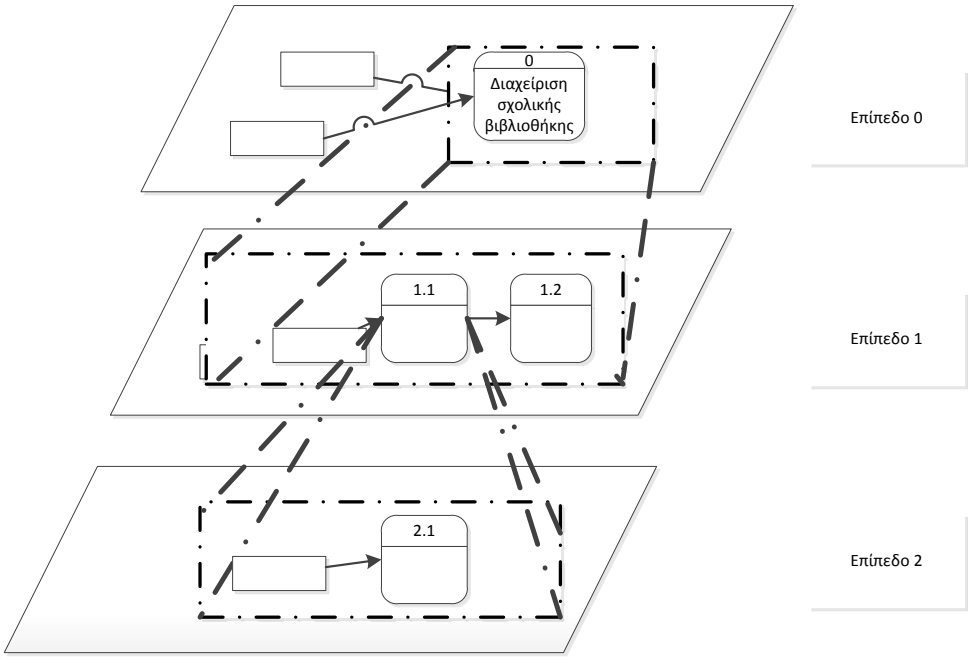
Σχήμα 3.3: *Αποθήκες δεδομένων*

Ροή Δεδομένων

Η ροή δεδομένων απεικονίζεται με τα βέλη τα οποία δείχνουν την πορεία των δεδομένων (μονόδρομη ή αμφίδρομη), ενώ συνοδεύονται από μια ετικέτα η οποία δείχνει το είδος των δεδομένων που μετακινούνται. Συνήθως είναι ηλεκτρονικές μετακινήσεις όπως εγγραφή ή ανάγνωση από μια βάση δεδομένων, αλλά μπορεί να είναι και φυσικά αντικείμενα όπως χρήματα, παραστατικά, παραγγελίες, κουτιά. Για παράδειγμα, στο σχήμα 3.3 παρουσιάζεται η ροή δεδομένων που αφορά πληροφορίες βιβλίων από την διαδικασία **Διαχείριση σχολικής βιβλιοθήκης** προς την αποθήκη **Βιβλία**.

Βήματα για την ανάπτυξη ενός ΔΡΔ

1. Αρχικά καθορίζουμε τις πηγές και τους προορισμούς και συνθέτουμε το γενικό διάγραμμα του συστήματος (Διάγραμμα περιεχομένων – **Επίπεδο 0**)
2. Αναλύουμε το γενικό διάγραμμα του βήματος 1, σε βασικές επεξεργασίες με μια είσοδο και μια έξοδο και καθορίζουμε τις ροές και συλλογές δεδομένων (**Επίπεδο 1**), όπως φαίνεται στο Σχήμα 3.4.
3. Ξεκινώντας από αριστερά προς τα δεξιά, διασπάμε κάθε μια επεξεργασία σε μικρότερες επεξεργασίες, μέχρι να περιγραφούν όλες.
4. Ελέγχουμε τη συμμετοχή των χρηστών, την ορθότητα της αναπαράστασης και τροποποιούμε το προηγούμενο διάγραμμα.
5. Το ΔΡΔ, καλό θα είναι, να μην υπερβαίνει τις 6 διαδικασίες για λόγους ευκολότερης κατανόησης, να μην υπερβαίνει την μια σελίδα χωροταξικά και η διάσπαση των λειτουργιών ή διαδικασιών να μην αναλύεται σε υπερβολικό βάθος.



Σχήμα 3.4: *Επίπεδα Διαγραμμάτων Ροής Δεδομένων*

Κανόνες Σχεδίασης Διαγραμμάτων Ροής Δεδομένων

Ο σχεδιασμός ενός ΔΡΔ γίνεται σύμφωνα με τους παρακάτω κανόνες:

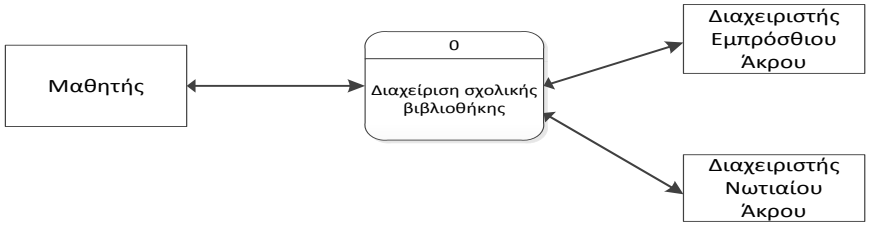
- Μία διαδικασία δεν επιτρέπεται να έχει μόνον εξόδους.
- Μία διαδικασία δεν επιτρέπεται να έχει μόνον εισόδους.
- Δεν επιτρέπεται ροή δεδομένων από μία αποθήκη δεδομένων σε μία άλλη αποθήκη δεδομένων. Η μετακίνηση αυτή των δεδομένων πρέπει να γίνει μέσω διαδικασίας.
- Δεν επιτρέπεται ροή δεδομένων από μία εξωτερική οντότητα σε μία άλλη εξωτερική οντότητα. Η μετακίνηση αυτή των δεδομένων πρέπει να γίνει μέσω διαδικασίας.
- Δεν επιτρέπεται ροή δεδομένων από μία εξωτερική οντότητα σε μία αποθήκη δεδομένων. Η μετακίνηση αυτή των δεδομένων πρέπει να γίνει μέσω διαδικασίας

Παράδειγμα Σχεδίασης Λειτουργιών Σχολικής Βιβλιοθήκης

Στο παράδειγμα που ακολουθεί, παρουσιάζεται το ΔΡΔ μιας σχολικής βιβλιοθήκης με 3 επίπεδα διάταξης (**3-Tier**). Το ένα επίπεδο είναι το **Client** του συστήματος που στην περίπτωσή μας είναι ο μαθητής, ο οποίος δανείζεται κάποιο βιβλίο ή το αναζητά. Στο δεύτερο επίπεδο είναι ο **Διαχειριστής Εμπρόσθιου Άκρου (Front End Side).** Ο Διαχειριστής αυτός διαχειρίζεται τον δανεισμό των βιβλίων, εγκρίνει αιτήσεις δανεισμού ή απαγορεύει κάποιες από αυτές. Τέλος υπάρχει και ο **Διαχειριστής Νωτιαίου Άκρου (Back End Side)**. Αυτός διαχειρίζεται την βάση των βιβλίων, εισάγει, διαγράφει ή τροποποιεί τα στοιχεία που αφορούν τα βιβλία, όπως ο Συγγραφέας, η περίληψη του βιβλίου, ή η εικόνα του εξώφυλλού του. Στην ουσία ο διαχειριστής αυτός έχει και τον ρόλο του διαχειριστή της Βάσης Δεδομένων του συστήματος (**DBMS Administartor**)

Σχεδιασμός Επιπέδου 0 (Context Diagram).

Η διαγραμματική απεικόνιση του παραδείγματος παρουσιάζεται στο σχήμα 3.5.

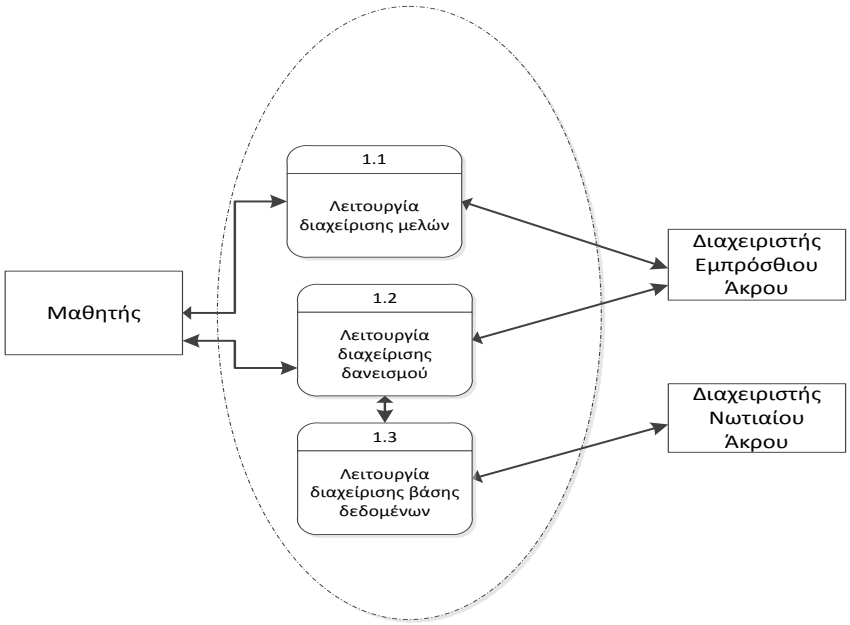


Σχήμα 3.5: *Επίπεδο 0 ΔΡΔ της δανειστικής βιβλιοθήκης*

Σχεδιασμός Επιπέδου 1

Στο πρώτο επίπεδο αναλύεται η διαδικασία Διαχείριση σχολικής βιβλιοθήκη, στην Λειτουργία διαχείρισης μελών, στην λειτουργία διαχείρισης δανεισμού και στην λειτουργία διαχείρισης της βάσης δεδομένων. Κάθε διαδικασία έχει έναν αριθμό στο

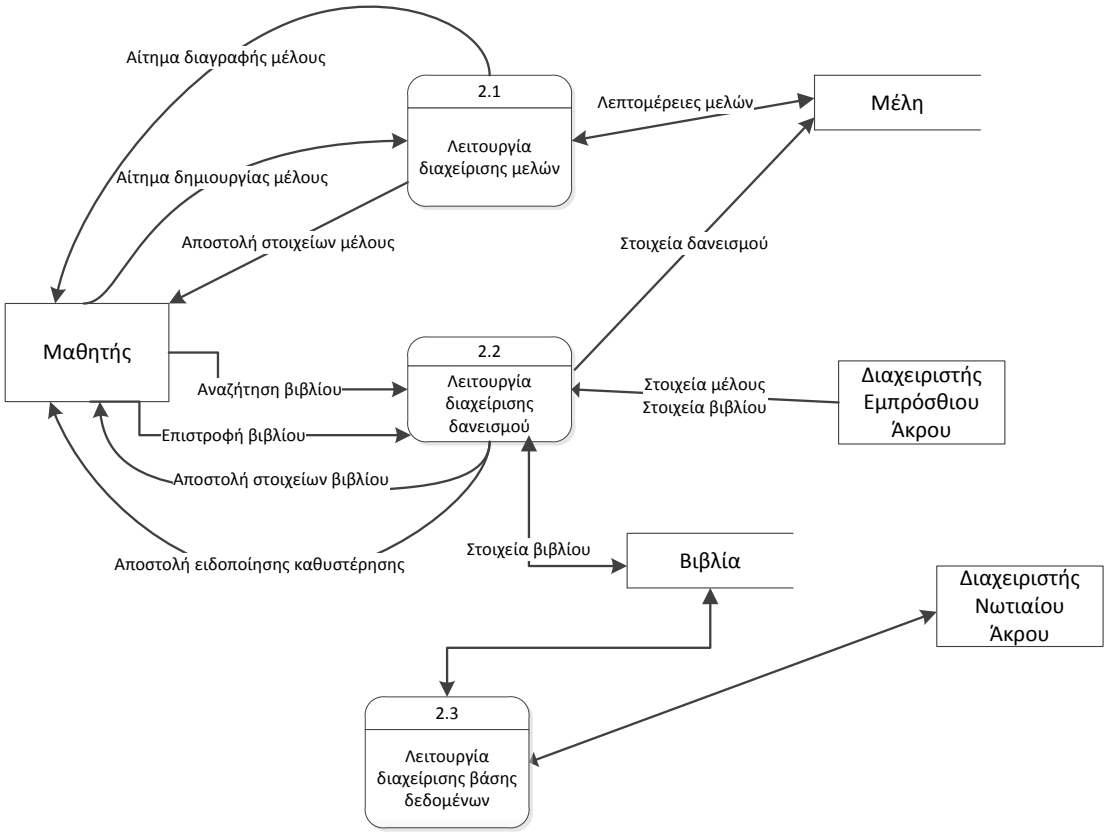
πάνω μέρος του σχήματος, που αποτελείται από το επίπεδο ανάλυσης και τον αύξοντα αριθμό της διαδικασίας. Για παράδειγμα η Λειτουργία διαχείρισης δανεισμού έχει αρίθμηση 1.2 καθώς παρουσιάζεται στο πρώτο επίπεδο και είναι η δεύτερη διαδικασία του επιπέδου (Σχήμα 3.6).



Σχήμα 3.6*:Επίπεδο 1 ΔΡΔ της δανειστικής βιβλιοθήκης*

Σχεδιασμός Επιπέδου 2

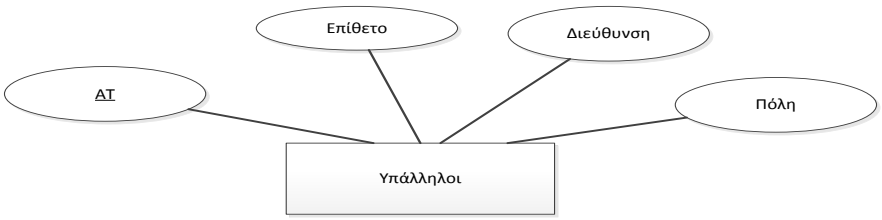
Στο δεύτερο επίπεδο αναλύονται περισσότερο οι διαδικασίες και οι ροές δεδομένων. Η ανάλυση των λειτουργιών του σχήματος 3.6 σε μεγαλύτερη λεπτομέρεια παρουσιάζεται στο σχήμα 3.7.



***Σχήμα 3.7**: Επίπεδο 2 ΔΡΔ της δανειστικής βιβλιοθήκης*

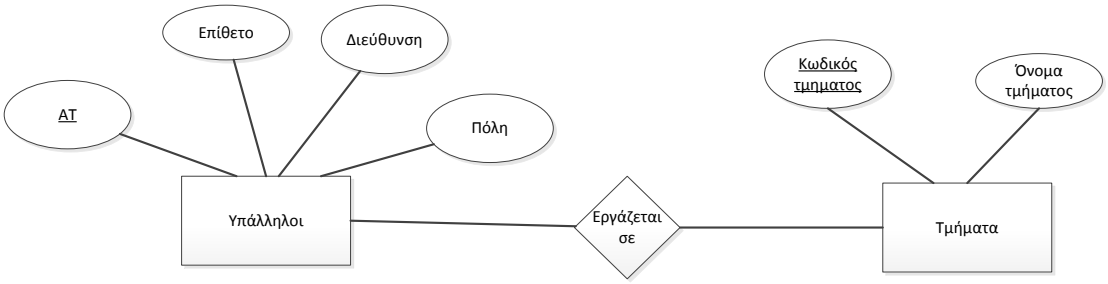
Διαγράμματα Οντοτήτων Συσχετίσεων

Το μοντέλο οντοτήτων συσχετίσεων αποτελεί το στάδιο του εννοιολογικού σχεδιασμού μιας βάσης δεδομένων και περιγράφει το λογικό (εννοιολογικό) σχήμα της βάσης δεδομένων. Το μοντέλο οντοτήτων συσχετίσεων περιγράφεται από το διάγραμμα οντοτήτων συσχετίσεων που περιγράφεται λεπτομερώς στην ενότητα 5. Παράδειγμα διαγραμματικής απεικόνισης του μοντέλου παρουσιάζεται στο σχήμα 3.8 που αφορά το σύνολο οντοτήτων Υπάλληλοι με γνωρίσματα όνομα, αριθμός Αστυνομικής Ταυτότητας (ΑΤ), επίθετο, διεύθυνση και πόλη κατοικίας.



***Σχήμα 3.8:** Σχηματική αναπαράσταση της οντότητας Υπάλληλοι*

Οι συσχετίσεις ανάμεσα στις οντότητες αναπαριστώνται με ρόμβο. Παράδειγμα παρουσιάζεται στο σχήμα 3.9.



Σχήμα 3.9: *Σχηματική αναπαράσταση της συσχέτισης*

Στο παραπάνω σχήμα περιγράφονται οι οντότητες Υπάλληλοι και Τμήματα. Τα τμήματα αντιπροσωπεύουν τα τμήματα που εργάζονται οι υπάλληλοι στον φυσικό κόσμο (Λογιστήριο, Τμήμα προσωπικού, τμήμα πωλήσεων κ.α). Η σχέση ανάμεσα στις δύο οντότητες περιγράφεται από τη συσχέτιση Εργάζεται σε. Το μοντέλο οντοτήτων συσχετίσεων χρησιμοποιείται ευρέως στην ανάλυση των απαιτήσεων μιας βάσης δεδομένων.

Πίνακες Αποφάσεων και Δένδρα αποφάσεων

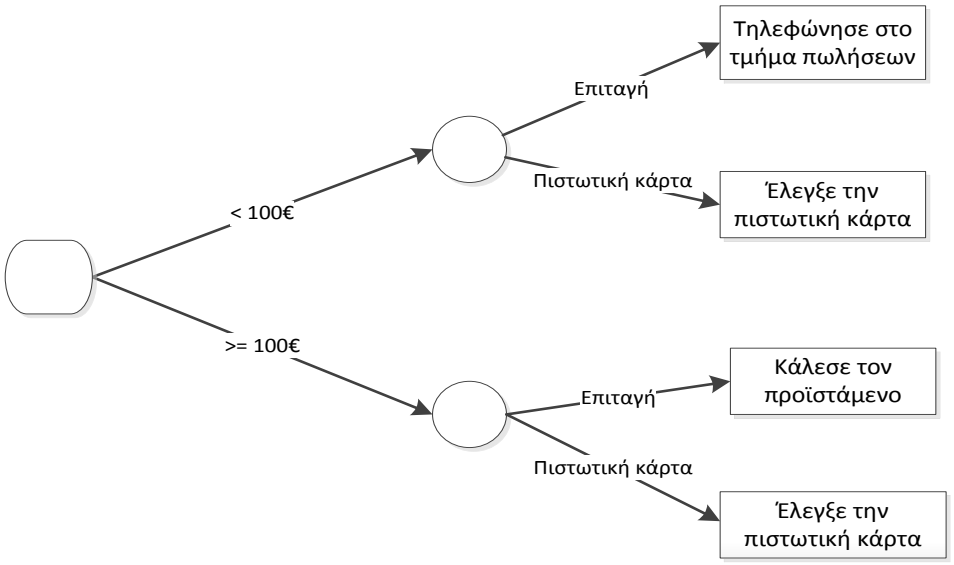
Οι πίνακες απόφασης και τα δέντρα απόφασης μπορούν να συνοψίζουν τη λογική μιας σύνθετης απόφασης πιο ουσιαστικά. Ο πίνακας απόφασης είναι πιο συμπαγής, αλλά το δέντρο αποφάσεων είναι πιο ευανάγνωστο. Μερικές φορές ο αναλυτής πρέπει να περιγράψει μια διαδικασία με όλους τους δυνατούς τρόπους προτού αποφασίσει ποια προσέγγιση περιγράφει μια συγκεκριμένη διαδικασία καλύτερα.

Στο παρακάτω παράδειγμα περιγράφεται με τις δύο αυτές μεθόδους οι ενέργειες που πρέπει κάνουν οι ταμίες ενός πολυκαταστήματος όταν οι πελάτες δεν πληρώνουν τοις μετρητοίς. Στο παρακάτω πίνακα αποφάσεων (Πίνακας 3.3) Ν σημαίνει ότι ικανοποιείται η συνθήκη, Ο σημαίνει ότι δεν ικανοποιείται η συνθήκη και η ενέργεια που πρέπει να εκτελέσει ο ταμίας σημειώνεται με √.

Πίνακας 3.3:*Πίνακας αποφάσεων*

Κάτω από 100 €	N	N	O	O
Πληρωμή με επιταγή	N	O	N	O
Πληρωμή με πιστωτική κάρτα	O	N	O	N
Τηλεφώνησε στο τμήμα πωλήσεων	√			
Έλεγξε την πιστωτική κάρτα		√		√
Κάλεσε τον προϊστάμενο			√	

Στο παρακάτω δένδρο αποφάσεων (Σχήμα 3.10), με κύκλο σημειώνονται τα γεγονότα που πρέπει να ελέγξει ο ταμίας (π.χ. αν το ποσό πληρωμής είναι μεγαλύτερο ή μικρότερο από 100 €) και με τετράγωνο η τελική του απόφαση.



Σχήμα 3.10: *Δένδρο αποφάσεων*

Λεξικά Δεδομένων

Το **Λεξικό Δεδομένων** είναι μια οργανωμένη συλλογή δεδομένων των σχετιζομένων στοιχείων του συστήματος, με μεγάλη λεπτομέρεια, έτσι ώστε να είναι διαθέσιμα τόσο στον αναλυτή του συστήματος όσο και στους χρήστες του.

Ο παρακάτω πίνακας (Πίνακας 3.4), περιέχει ένα παράδειγμα από λεξικό δεδομένων για μια υποθετική εφαρμογή τράπεζας.

Πίνακας 3.4:*Λεξικό δεδομένων μιας τράπεζας*

Όνομα	Τύπος	Περιγραφή
Πελάτης	Δομή	Η εγγραφή με τα στοιχεία ενός πελάτη. Περιέχει όνομα, κωδικό λογαριασμού και υπόλοιπο λογαριασμού.
Κωδικός λογαριασμού	INT(20)	Ο μοναδικός κωδικός του λογαριασμού
Όνομα	CHAR(30)	Το ονοματεπώνυμο ενός πελάτη
Υπόλοιπο λογαριασμού	INT(12)	Το καθαρό υπόλοιπο του λογαριασμού σε Ευρώ

3.3.2 Αντικειμενοστραφής προσέγγιση

Η αντικειμενοστραφής προσέγγιση για την ανάλυση απαιτήσεων έχει σκοπό την ανάπτυξη ενός αντικειμενοστραφούς μοντέλου που θα αναπαριστά το πεδίο της εφαρμογής. Σε αυτό το μοντέλο τα αντικείμενα συμβολίζουν τις έννοιες του συστήματος και αποτελούνται από ιδιότητες και μεθόδους.

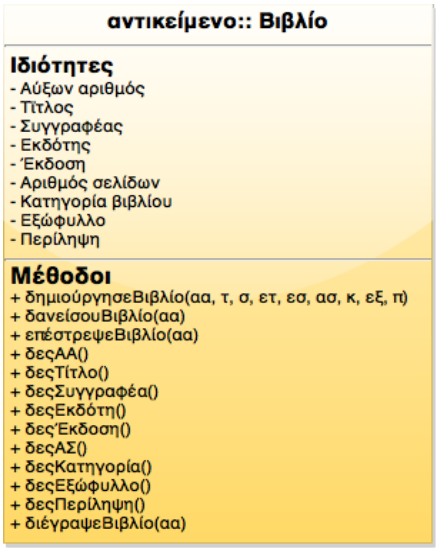
Σχήμα 3.11: *Το αντικείμενο*

Οι ιδιότητες είναι μια σειρά από μεταβλητές και χαρακτηρίζουν την κατάσταση των αντικειμένων. Όταν ο αναλυτής θα πρέπει να χρησιμοποιήσει περισσότερες από μία λέξεις για να ορίσει το όνομα μιας ιδιότητας τότε οι λέξεις ενώνονται αφήνοντας το πρώτο γράμμα με μικρό και τα πρώτα γράμματα των υπόλοιπων λέξεων με κεφαλαία ή χρησιμοποιείται ο χαρακτήρας κάτω παύλα π.χ. «κατηγορίαΒιβλίου» ή «κατηγορία_βιβλίου». Επίσης ο αναλυτής θα πρέπει να έχει κάποια ιδέα για τον τύπο των δεδομένων της ιδιότητας π.χ. αν είναι ακέραιος αριθμός, αλφαριθμητικό, κείμενο ή λογική τιμή.

Οι μέθοδοι είναι συναρτήσεις. Ουσιαστικά αποτελούν τις διεπαφές των αντικειμένων που τις επιτρέπουν να επικοινωνούν με άλλες οντότητες του συστήματος και ορίζουν την συμπεριφορά τους. Η κάθε μέθοδος χαρακτηρίζεται από ένα όνομα που ακολουθείται από μια παρένθεση που μπορεί να περιέχει προαιρετικά ένα ή περισσότερα ορίσματα. Όταν καλούμε τις μεθόδους επιστρέφουν μια τιμή της οποίας τον τύπο ο αναλυτής θα πρέπει να έχει προσδιορίσει. Για την περίπτωση του βιβλίου ένα παράδειγμα μεθόδου θα μπορούσε να είναι το «δανείσουΒιβλίο()».

Ένα σύστημα που περιγράφεται με βάση την αντικειμενοστραφή προσέγγιση μεταφράζεται σε ένα σύνολο αντικειμένων που επικοινωνούν μέσω των μεθόδων-διεπαφών μεταξύ τους και με τους χρήστες του συστήματος.

Η αντικειμενοστραφής προσέγγιση είναι ιδιαίτερα δημοφιλής καθώς προσφέρει έναν ρεαλιστικό τρόπο αναπαράστασης διακριτών αντικειμένων με σαφείς ιδιότητες και μεθόδους καθορισμένες μέσα στα πλαίσια του συστήματος. Για παράδειγμα ένα βιβλίο μιας σχολικής βιβλιοθήκης μπορεί να έχει τις ιδιότητες και μεθόδους που φαίνονται στο παρακάτω σχήμα:



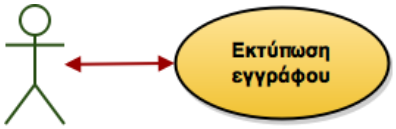
Σχήμα 3.12: *Το αντικείμενο βιβλίο: Ιδιότητες και Μέθοδοι*

Πως όμως αναγνωρίζουμε ποιες έννοιες πρέπει να αντιστοιχίσουμε ως αντικείμενα στο πεδίο ενός συστήματος; Ο ορισμός των αντικειμένων του πεδίου ενός συστήματος αποτελεί μια δύσκολη διαδικασία για την οποία δεν έχει αναπτυχθεί κάποια κοινά αποδεκτή αυτοματοποιημένη διαδικασία. Στηρίζεται στην ικανότητα, στην πείρα και στις γνώσεις του τομέα των αναλυτών του συστήματος. Παρόλα αυτά έχουν προταθεί κάποιες μεθοδολογίες που βοηθούν τον αναλυτή στην διάκριση των αντικειμένων σε ένα πεδίο. Έτσι ο αναλυτής μπορεί να εργαστεί πάνω σε κάποιες περιγραφές που αναφέρονται στο σύστημα. Τα ουσιαστικά μπορεί να είναι αντικείμενα ή ιδιότητες αντικειμένων ενώ τα ρήματα ενδέχεται να αναφέρονται στις μεθόδους τους. Επίσης ο αναλυτής μπορεί να εντοπίσει κάποια συμβάντα που συμβαίνουν στο σύστημα. Ένα συμβάν μπορεί να έχει άμεση σχέση με το κάλεσμα μιας μεθόδου σε ένα αντικείμενο. Η οντότητα που καλεί αυτό το συμβάν και η οντότητα που το δέχεται μπορεί να αποτελούν αντικείμενα.

Δραστηριότητα: Στην περίπτωση της σχολικής βιβλιοθήκης ποια αντικείμενα εντοπίζετε; Πως εργαστήκατε για να τα εντοπίσετε;

Περιπτώσεις χρήσης

Οι περιπτώσεις χρήσης είναι μια τεχνική περιγραφής απαιτήσεων για την αναπαράσταση μοντέλων συστημάτων που ακολουθούν την αντικειμενοστραφή προσέγγιση. Περιλαμβάνει διαγράμματα για την απεικόνιση της χρήσης του συστήματος συνοδευόμενα με κάποια τεκμηρίωση σε φυσική γλώσσα. Ουσιαστικά αποτελούν διηγήσεις της λειτουργικότητας του συστήματος όπως γίνεται αντιληπτή από την πλευρά του τελικού χρήστη και απεικονίζουν την αλληλεπίδραση του συστήματος με τους χρήστες ή άλλα συστήματα.



Σχήμα 3.13: *Περίπτωση χρήσης εκτύπωσης εγγράφου*

Τα διαγράμματα των περιπτώσεων χρήσης αποτελούνται από τους πράκτορες - ηθοποιούς (actors) που απεικονίζονται με μια ανθρώπινη φιγούρα και τις περιπτώσεις χρήσης που απεικονίζονται με μια έλλειψη. Οι πράκτορες είναι χρήστες του συστήματος ή κάποια εξωτερική οντότητα που αλληλεπιδρά με αυτό για να εκπληρώσει μια περίπτωση χρήσης. Όταν ο πράκτορας αντιπροσωπεύει κάποιο χρήστη τότε αυτός είναι κάποιο πρόσωπο που παίζει ρόλο στη λειτουργία του συστήματος π.χ. υπεύθυνος βιβλιοθήκης, διευθυντής, πωλητής κτλ.

Η περίπτωση χρήσης είναι κάποια ενέργεια του συστήματος που έχει εκφραστεί ως μια λειτουργική απαίτηση. Συνδέεται με μια γραμμή με τον πράκτορα του συστήματος που παρουσιάζει την αλληλεπίδρασή τους δηλαδή μια σχέση επικοινωνίας μεταξύ τους. Στο παράδειγμα του σχήματος 3.13 παρουσιάζεται μια περίπτωση χρήσης κατά την οποία ένας χρήστης κάποιου συστήματος εκτυπώνει ένα έγγραφο π.χ. ο υπεύθυνος βιβλιοθήκης μια λίστα με τα βιβλία που έχει δανειστεί ένας χρήστης.

Μια συλλογή από διαγράμματα περίπτωσης χρήσης μπορούν να αντιπροσωπεύουν όλες τις πιθανές λειτουργικές απαιτήσεις του συστήματος. Θα πρέπει να σημειωθεί ότι ο όρος χρήστης του συστήματος και ο πράκτορας δεν ταυτίζονται σε όλα τα διαγράμματα ενός συστήματος αφού ένα χρήστης μπορεί να παίξει πολλούς ρόλους πράκτορα-ηθοποιού. Στο παράδειγμα της σχολικής βιβλιοθήκης σε μια περίπτωση χρήσης ένας μαθητής είναι ο πράκτορας που δανείζεται ένα βιβλίο ενώ παράλληλα σε μια διαφορετική περίπτωση χρήσης μπορεί να είναι ο πράκτορας που το επιστρέφει.

Τα διαγράμματα περίπτωσης χρήσης συνοδεύονται από μία περιγραφή σε φυσική γλώσσα όπως αυτή που ακολουθεί για την περίπτωση χρήσης του δανεισμού ενός βιβλίου:

Ο υπεύθυνος της βιβλιοθήκης εισάγει το όνομα του χρήστη στο σύστημα και ελέγχει αν μπορεί να δανειστεί ένα βιβλίο. Στη συνέχεια εισάγει το αναγνωριστικό του βιβλίο και το όνομα του χρήστη στο σύστημα και καταχωρεί τον δανεισμό. Ενημερώνει τον χρήστη για την ημερομηνία που θα πρέπει να επιστραφεί το βιβλίο.

Ωστόσο κατά τον δανεισμό ενός βιβλίου θα μπορούσε να εξελιχθεί και ένα διαφορετικό σενάριο:

Ο υπεύθυνος της βιβλιοθήκης εισάγει το όνομα του χρήστη στο σύστημα και ελέγχει αν μπορεί να δανειστεί ένα βιβλίο. Ο χρήστης έχει φτάσει το όριο των βιβλίων που μπορεί να δανειστεί και έτσι θα πρέπει να ενημερωθεί ότι πρέπει να κάνει μια

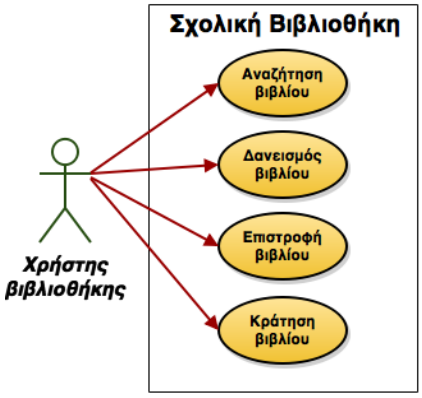
επιστροφή για να μπορέσει να δανειστεί το βιβλίο.

Οι διαφορετικές αυτές περιπτώσεις που μπορούν να εξελιχθούν κατά την εκτέλεση μίας λειτουργίας του συστήματος ονομάζονται σενάρια.

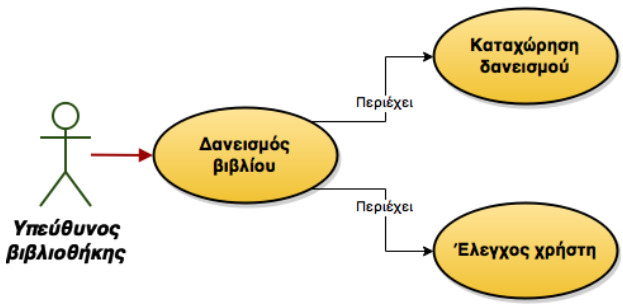
Δραστηριότητα: Μπορείς να γράψεις κάποια εναλλακτικά σενάρια που μπορεί να συμβούν κατά τη διάρκεια της λειτουργίας δανεισμού ενός βιβλίου από τη σχολική βιβλιοθήκη;

Σημείωση: Τι γίνεται αν ο χρήστης ή το βιβλίο δεν είναι εγγεγραμμένα στο σύστημα; Τι γίνεται αν δεν είναι διαθέσιμη η σύνδεση στο διαδίκτυο;

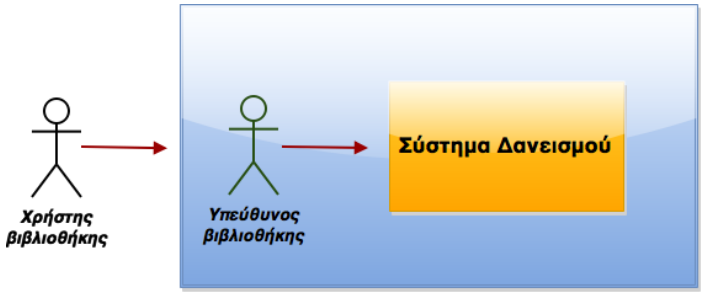
Δραστηριότητα: Μελέτησε τα παραδείγματα περίπτωσης χρήσης που απεικονίζονται στα σχήματα 3.14, 3.15 και 3.16. Να γράψεις την περιγραφή που πρέπει να συνοδεύει αυτά τα παραδείγματα.



Σχήμα 3.14: *Περίπτωση χρήσης χρήστη βιβλιοθήκης*



Σχήμα 3.15: *Περίπτωση χρήσης δανεισμού βιβλίου*



Σχήμα 3.16: *Περίπτωση χρήσης συστήματος δανεισμού*

3.4 Καθορισμός Προδιαγραφών

Η ανάπτυξης πληροφορικών συστημάτων και συγκεκριμένα η έννοια της ανάλυσης αναφέρεται στην μελέτη ενός τομέα της οικονομίας ή μιας εφαρμογής, η οποία οδηγεί στην δημιουργία προδιαγραφών για ένα νέο σύστημα, το οποίο θα οδηγήσει στην υλοποίηση του αντίστοιχου πληροφοριακού συστήματος. Όπως αναφέρθηκε στην ενότητα 3.2 το τελευταίο στάδιο της διαδικασίας προσδιορισμού απαιτήσεων είναι η προδιαγραφή απαιτήσεων. Σε αυτό το στάδιο καθορίζονται οι προδιαγραφές του υπό μελέτη συστήματος και καταγράφονται στο πιο σημαντικό παραδοτέο υλικό της ανάλυσης, το λεγόμενο **Κείμενο Προδιαγραφών** των **Απαιτήσεων** ή **Κείμενο του Στόχου** (Target Document). Επίσης απαραίτητο είναι να συνοδεύεται από ένα σύνολο μοντέλων αποτύπωσης του συστήματος σε μορφή Διαγραμμάτων Ροής Δεδομένων (Data Flow Diagrams), Οντοτήτων – Συσχετίσεων (Entity Relation) μαζί με τους Πίνακες Απόφασης/ Δένδρα Απόφασης και το Λεξικό Δεδομένων (Data Dictionary).

Προδιαγραφή

Προδιαγραφή αποτελεί τη δομημένη και λεπτομερή περιγραφή των απαιτήσεων του ΠΣ, η οποία γίνεται με τη μορφή γραπτού λόγου και συνοδεύεται με τη μορφή διαγραμμάτων και πινάκων.

Το κείμενο προδιαγραφών των απαιτήσεων που προκύπτει στο τέλος της διαδικασίας προσδιορισμού των απαιτήσεων είναι αναμφίβολα το σημαντικότερο από τα έγγραφα τεκμηρίωσης των εφαρμογών λογισμικού. Είναι σημαντικό να αναφερθεί ότι ελλείψεις, αστοχίες και λάθη όσων αναφέρονται σε αυτό θα μεταφερθούν σε όλη την υπόλοιπη διαδικασία κατασκευής του λογισμικού και ασφαλώς θα έχουν επιπτώσεις στο τελικό προϊόν. Σε πολλές περιπτώσεις έχει παρατηρηθεί ότι το τελικό προϊόν είναι άχρηστο, καθώς δεν ικανοποιούσε τις απαιτήσεις των χρηστών, όμως όπως αναφέρθηκε και στο κύκλο ζωής λογισμικού η διόρθωση αυτών είναι μια αρκετά χρονοβόρος και πολυέξοδη διαδικασία.

Όπως αναφέρθηκε ένα κείμενο προδιαγραφών απαιτήσεων είναι ένα επίσημο έγγραφο το οποίο ο αναλυτής και η ομάδα του θα το στείλει σε αυτούς που έχουν παραγγέλλει το λογισμικό. Βασικός σκοπός είναι η καταγραφή των απαιτήσεων και η αποδοχή αυτών από τους πελάτες. Σε περίπτωση που οι πελάτες έχουν αντιρρήσεις,

ζητούνται προτάσεις οι οποίες θα οδηγήσουν στην αλλαγή κάποιων απαιτήσεων και στην τελική συμφωνία. Η χρήση τέτοιων κειμένων είναι καθολική στις κυβερνητικές συμβάσεις και πολύ συχνά στον ιδιωτικό τομέα. Πολλές φορές είναι ένα νομικό έγγραφο και οι πελάτες βασίζονται σε αυτό.

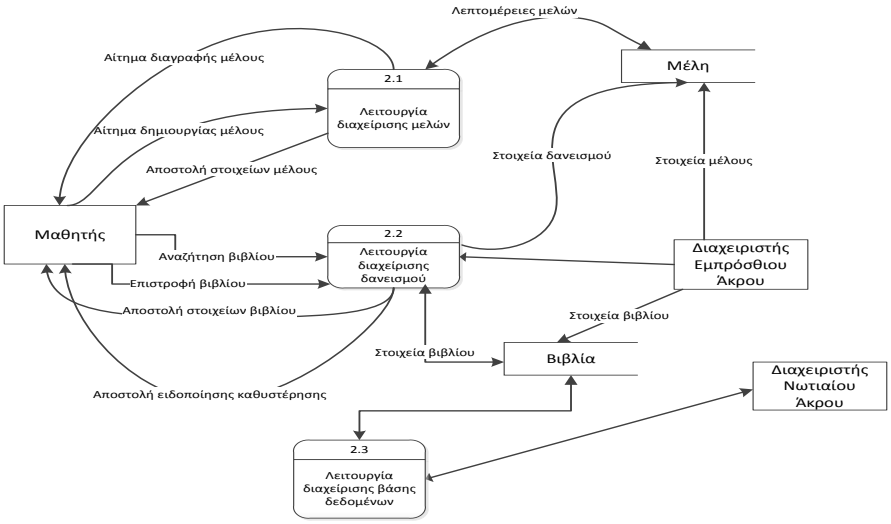
Για να είναι καλό ένα τέτοιο κείμενο θα πρέπει να έχει γίνει σε βάθος ανάλυση του συστήματος. Επίσης, τα άτομα που θα συμμετάσχουν στην ομάδα ανάλυσης θα πρέπει να είναι έμπειρα και ειδικά ο αναλυτής, του οποίου ο ρόλος είναι ιδιαίτερα σημαντικός στην δημιουργία του κειμένου.

Υπάρχουν αρκετά πρότυπα για τη δόμηση αυτού του εγγράφου, όμως αυτό που μας ενδιαφέρει είναι να έχει τα ακόλουθα επιθυμητά χαρακτηριστικά:

- Θα πρέπει να περιγράφει τις οργανωτικές επιπτώσεις (επανεκπαίδευση του προσωπικού)
- Θα πρέπει να υπολογίζεται το αναμενόμενο κόστος μετατροπής δεδομένων.
- Θα πρέπει να περιγράφει τη συμπεριφορά του λογισμικού προς το εξωτερικό του περιβάλλον (χρήστης, άλλες εφαρμογές λογισμικού).
- Θα πρέπει να καταγράφει όλους τους περιορισμούς που αφορούν την ανάπτυξη του λογισμικού.
- Θα πρέπει να είναι εύκολο να αλλαχτεί στην περίπτωση που οι πελάτες δεν συμφωνήσουν και στείλουν άλλες προτάσεις.
- Θα πρέπει να χρησιμεύει στη συντήρηση του λογισμικού. Για το λόγο αυτό καλό είναι να συνοδεύεται από διαγράμματα και πίνακες που περιγράφουν τις απαιτήσεις έτσι όπως προέκυψαν στη φάση της ανάλυσης απαιτήσεων.
- Θα πρέπει να περιγράφει τη συμπεριφορά των χρηστών/ προγραμματιστών όταν υπάρξουν προβλήματα και ανεπιθύμητες περιπτώσεις.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Περιγράψτε τα τέσσερα σύμβολα ενός ΔΡΔ και τι αναπαριστά το κάθε ένα.
2. Για ποιους λόγους ένας αναλυτής θα περιγράψει μια διαδικασία με έναν πίνακα ή ένα δένδρο αποφάσεων;
3. Σας δίνεται το ΔΡΔ. Υπάρχουν λάθη και γιατί αποτελούν λάθη κατά τη γνώμη σας;



4. Αφήνεται σαν άσκηση ο εμπλουτισμός της σχεδίασης του ΔΡΔ με περισσότερες λειτουργίες για μεγαλύτερη λειτουργικότητα του συστήματος της Σχολικής Βιβλιοθήκης. Η σχεδίαση θα μπορούσε να περιλαμβάνει :

    - Αποθήκη με τις ποινές που θα υπήρχαν για την καθυστέρηση των βιβλίων.

    - Λειτουργία Εκτύπωσης καταστάσεων Μελών και Βιβλίων.

5. Λειτουργία Εύρεσης των Δημοφιλέστερων Βιβλίων και σε συνεργασία με την λειτουργία των εκτυπώσεων να εκτυπώνεται σχετική αναφορά. Να γίνει ανάλυση απαιτήσεων και στη συνέχεια ο σχεδιασμός ΔΡΔ ενός πληροφοριακού συστήματος για ένα σχολείο. Θα υπάρχουν ως εξωτερικές οντότητες οι μαθητές, οι καθηγητές, και οι γονείς. Λειτουργίες μπορούν να είναι η εγγραφή/διαγραφή του μαθητή, η διδασκαλία από τους καθηγητές, η ενημέρωση των γονέων κλπ. Ως αποθήκες μπορούν να είναι το αρχείο των μαθητών, των καθηγητών, των γονέων κλπ.

Βιβλιογραφία

Στα Ελληνικά

Βεσκούκης, Β. (2000). *Τεχνολογία Λογισμικού.* Πάτρα: Ι. Ε.Α.Π,

Γιακουμάκης Μ. & Διαμαντίδης Ν. (2009). *Τεχνολογία λογισμικού*. Αθήνα: Σταμούλη Α.Ε.

Στα Αγγλικά

O'Brien J. A. & Marakas G. (2008). *Introduction to Information Systems*. 14th/edition, New York: McGraw-Hill.

Pfleeger L.S. (2012). *Τεχνολογία λογισμικού Θεωρία και πράξη.* Μτφρ. Φρυσήρας Κ., Επιμ Σταμέλος Γ, Αθήνα: Κλειδάριθμος.

Sommerville I. (2009). *Βασικές αρχές τεχνολογίας λογισμικού. Μτφρ*. Τσιλογιάννης Δ. Αθήνα: Κλειδάριθμος.

ΚΕΦΑΛΑΙΟ 4

Εισαγωγή στις Αρχιτεκτονικές Σχεδίασης

Περιεχόμενα

4.1. Εισαγωγή στις Αρχιτεκτονικές Σχεδίασης
4.2. Σχεδιασμός Προσανατολισμένος στις διαδικασίες (Function Oriented)
4.3. Σχεδιασμός Προσανατολισμένος στα Αντικείμενα (Object Oriented)
4.4. Σχεδιασμός Διεπαφής Χρήστη

Διδακτικοί Στόχοι

Στόχοι του κεφαλαίου αυτού είναι:

- Να περιγράφουν τα στάδια της φάσης σχεδιασμού συμπεριλαμβανομένης της αρχιτεκτονικής σχεδίασης και της σχεδίασης διεπαφών.
- Να χρησιμοποιούν τους διάφορους τύπους αρχιτεκτονικών στον σχεδιασμό πληροφοριακών συστημάτων.
- Να αξιοποιούν συγκεκριμένες αρχές και να υιοθετούν καλές πρακτικές για να σχεδιάζουν αποτελεσματικά τη γραφική διεπαφή χρήστη.
- Να εκτιμούν την αναγκαιότητα του σχεδιασμού αρχιτεκτονικής και λογισμικού.

4.1. Εισαγωγή στις Αρχιτεκτονικές Σχεδίασης

Όσο το μέγεθος και η πολυπλοκότητα των πληροφοριακών συστημάτων μεγαλώνει ραγδαία, το πρόβλημα της σχεδίασης του λογισμικού και δομών δεδομένων δεν αποτελούν πλέον κύρια προβλήματα σχεδίασης. Όταν τα συστήματα αποτελούνται από πολλά δομικά στοιχεία (components), η οργάνωση του συνολικού συστήματος αποτελεί την νέα σχεδιαστική πρόκληση.

Για την σχεδίαση τέτοιων συστημάτων επιστρατεύονται πολλοί και διάφοροι τρόποι, τεχνοτροπίες και μέθοδοι, οι οποίες περιλαμβάνουν άτυπα διαγράμματα και περιγραφικούς όρους, γλώσσες παραμέτρων σύνδεσης των τμημάτων, πρότυπα και τυπικά μοντέλα μηχανισμών ολοκλήρωσης των διαφορετικών τμημάτων.

Όταν αναφερόμαστε στα δομικά προβλήματα σχεδίασης εννοούμε την δυναμική οργάνωση και την ολική δομή επικοινωνίας των μερών του συστήματος, πρωτόκολλα επικοινωνίας, θέματα συγχρονισμού των μερών, ανάθεση λειτουργικότητας ξεχωριστά σε κάθε τμήμα του συστήματος, κατανομή φυσικών πόρων, σύνθεση των τμημάτων και τέλος αξιολόγηση των διαφόρων εναλλακτικών αρχιτεκτονικών σχεδίασης.

Όλα αυτά συνιστούν το επίπεδο της αρχιτεκτονικής σχεδίασης του συστήματος. Είναι προφανές ότι η σωστή κατασκευή ενός συστήματος προϋποθέτει την σωστή αρχιτεκτονική της σχεδίασης του, η οποία θα αποτελέσει την βάση. Επίσης :

1. Πρέπει να κατηγοριοποιηθούν τα υπάρχοντα συστήματα των οποίων τα κοινά χαρακτηριστικά θα μπορούν να χρησιμοποιηθούν ως πρότυπα για την σχεδίαση νέων συστημάτων.

2. Η σωστή επιλογή αρχιτεκτονικής συστήματος πρέπει να γίνει από την αρχή γιατί μπορεί να οδηγήσει σε καταστροφικά αποτελέσματα.

3. Λεπτομερής κατανόηση των αρχιτεκτονικών σχεδίασης των συστημάτων επιτρέπει στον υπεύθυνο σχεδιασμού να κάνει τις κατάλληλες επιλογές μεταξύ εναλλακτικών επιλογών σχεδίασης.

4. Η παρουσίαση της αρχιτεκτονικής ενός συστήματος είναι ουσιαστική για την ανάλυση και περιγραφή του υψηλότερου επιπέδου ενός σύνθετου συστήματος.

Η Αρχιτεκτονική Σχεδίαση (ΑΣ) περιγράφει τον τρόπο λειτουργίας και χρήσης τμημάτων του συστήματος, μας βοηθάει να καταλάβουμε πως θα έπρεπε να οργανωθεί και να σχεδιασθεί η συνολική δομή του πληροφοριακού συστήματος. Είναι το πρώτο πράγμα που πρέπει να κάνουμε κατά την διαδικασία της σχεδίασης και αποτελεί την γέφυρα μεταξύ της σχεδίασης και της διαδικασίας κατασκευής του λογισμικού. Το αποτέλεσμα της αρχιτεκτονικής σχεδίασης είναι ένα αρχιτεκτονικό

μοντέλο το οποίο περιγράφει πως το όλο σύστημα είναι οργανωμένο ως ένα σύνολο από κομμάτια που επικοινωνούν μεταξύ τους.

Κατά την Ευέλικτη Διαδικασία (Agile Process) είναι γενικά αποδεκτό ότι το πρώτο στάδιο, όπως είναι η σχεδίαση, θα έπρεπε να ασχολείται με το να δώσει μια συνολική εικόνα του συστήματος. Αντίθετα η μέθοδος της Σταδιακής Ανάπτυξης, με την οποία ξεκινάμε την σχεδίαση από ένα αρχικό τμήμα και στην πορεία συνδέουμε καινούργια τμήματα, αποδείχθηκε στην πορεία ένας λάθος τρόπος προσέγγισης. Η τρίτη επιλογή που έχουμε είναι να γίνει μια αρχική σχεδίαση και στην πορεία να κάνουμε διορθωτικές επεμβάσεις. Ο τρόπος αυτός αν και είναι εύκολος, έχει αρκετά μεγάλο κόστος.

Στην πράξη, υπάρχει σημαντική επικάλυψη μεταξύ της διαδικασίας εύρεσης των απαιτήσεων του συστήματος και της αρχιτεκτονικής σχεδίασης. Κανονικά η **καταγραφή προδιαγραφών** του συστήματος **δεν πρέπει** να περιέχει στοιχεία σχεδίασης. Αυτό επιτρέπεται μόνον στον σχεδιασμό μικρών συστημάτων. Η αρχιτεκτονική σχεδίασης έχει σαν αποστολή να βοηθήσει στην οργάνωση και δόμηση των προδιαγραφών (System Specifications). Συνεπώς ο αναλυτής του συστήματος, θα μπορούσε να προτείνει ως μέρος του ορισμού των απαιτήσεων του συστήματος, μια ΑΣ, όπου θα υπάρχουν οι συσχετισμοί μεταξύ των λειτουργιών του συστήματος ή χαρακτηριστικών μεταξύ των μεγάλων ή μικρότερων τμημάτων.

Ο σχεδιασμός της αρχιτεκτονικής σχεδίασης ανάλογα με το μέγεθος της, μπορεί να είναι δύο κατηγοριών.

- **Αρχιτεκτονική Σχεδίαση Μικρής Κλίμακας.** Αυτή ασχολείται με την ΑΣ μικρών τμημάτων του συστήματος. Το επίπεδο αυτό αφορά τα μικρότερα τμήματα από τα οποία αποτελείται ένα βασικό τμήμα του συστήματος. Η βάση δίνεται στην αρχιτεκτονική προγραμμάτων και τοπικών λειτουργιών.

- **Αρχιτεκτονική Σχεδίαση Μεγάλης Κλίμακας.** Αυτή ασχολείται με πολύ-σύνθετα συστήματα επιχειρήσεων και οργανισμών τα οποία περιέχουν και άλλα συστήματα, προγράμματα και κομμάτια άλλων προγραμμάτων. Αυτά τα διάφορα συστήματα είναι κατανεμημένα σε διαφορετικούς υπολογιστές και ενδεχομένως να τα υποστηρίζουν διαφορετικές και ανεξάρτητες μεταξύ τους εταιρείες ή οργανισμοί.

4.1.1. Τεχνοτροπίες Σχεδίασης

Οι τεχνοτροπίες αυτές είναι προσανατολισμένες στις εξής δυο βασικές κατηγορίες :

1. **Σχεδίαση βασισμένη στις Διαδικασίες (Function Oriented)**, όπου γίνεται διάκριση δεδομένων από τις διαδικασίες (Δομημένη Σχεδίαση).

2. **Σχεδίαση βασισμένη στα αντικείμενα (Object Oriented)**, όπου οι οντότητες και οι διαδικασίες ενσωματώνονται στο ίδιο κέλυφος. Η συλλογή των

οντοτήτων που περικλείουν τα δεδομένα και τις διαδικασίες παρέχουν το σύνολο των υπηρεσιών.

Ο σχεδιαστής και στις δυο περιπτώσεις ασχολείται και με τα δεδομένα και με τις διαδικασίες. Αναλυτική παρουσίαση των δύο αυτών τεχνοτροπιών παρουσιάζονται στα παρακάτω κεφάλαια.

Οι παραπάνω τεχνοτροπίες που εξετάζονται αναλυτικά στη συνέχεια, συχνά αναφέρονται και σαν ιεραρχικά διαγράμματα, γιατί έχουν τη μορφή δέντρου. Για τον ίδιο λόγο και οι τεχνικές αυτές ονομάζονται και δομημένες διαγραμματικές τεχνικές.

4.1.2. Είδη Σχεδίασης

Σχετικά με τα είδη σχεδίασης, έχουμε τα εξής:

- **Αρχιτεκτονική Σχεδίαση**

  Στη σχεδίαση αυτή γίνεται καθορισμός μονάδων του λογισμικού και διάταξής τους στο σύστημα

- **Σχεδίαση Διαπροσωπειών (Interfaces)**

  Εδώ κάνουμε τον καθορισμό της επικοινωνίας των μονάδων του συστήματος με :

  - Άλλα συστήματα
  - Συσκευές
  - Χρήστες

- **Λεπτομερής Σχεδίαση Μονάδων**

  Στο επίπεδο αυτό γίνεται καθορισμός της δομής κάθε μονάδας του συστήματος

- **Σχεδίαση Δεδομένων**

  Λεπτομερή περιγραφή και σχεδίαση των δεδομένων

4.2. Αρχιτεκτονική Σχεδίασης προσανατολισμένη στις Διαδικασίες (Function Oriented)

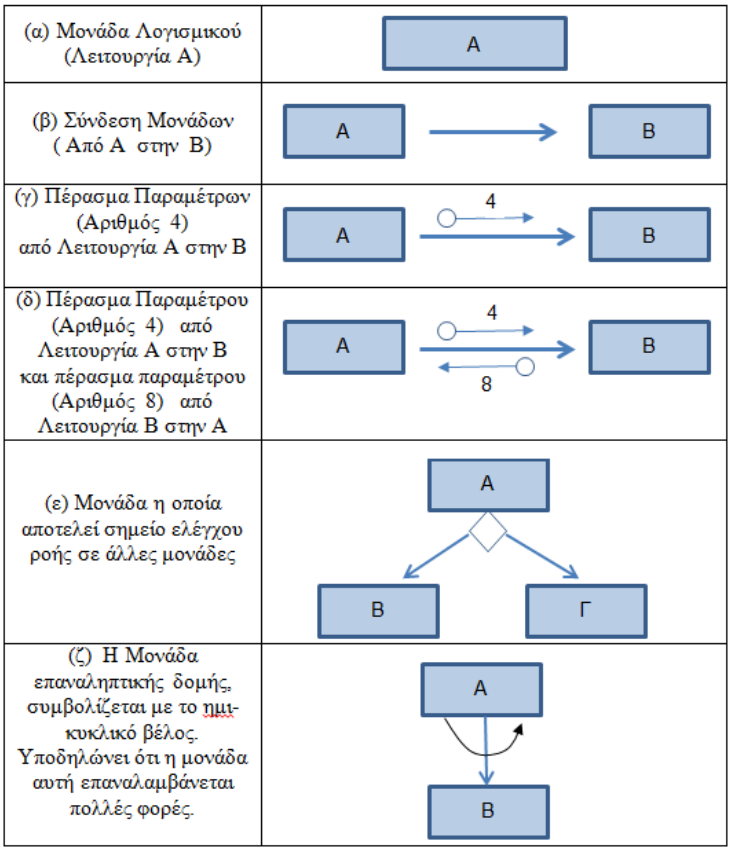
Ας δούμε την διαδικασία κατασκευής του αρχιτεκτονικού σχεδίου ενός συστήματος μιας σχολικής βιβλιοθήκης, βασιζόμενοι στο διάγραμμα ροής δεδομένων του προηγούμενου κεφαλαίου.

4.2.1 Διάγραμμα Δομής Συστήματος

Η διαγραμματική απεικόνιση είναι ένα πολύ σημαντικό βήμα για την αποτύπωση του σχεδιαστικού μοντέλου του συστήματός μας.

Το διάγραμμα δομής βασίζεται στο διάγραμμα ροής δεδομένων (ΔΡΔ), από το οποίο μέσα από κάποια διαδοχικά βήματα μπορούμε να οδηγηθούμε στο διάγραμμα δομής (ΔΔ). Στα διαγράμματα αυτά οι μονάδες λογισμικού παριστάνονται με ένα παραλληλόγραμμο (α), όπως φαίνεται στο Σχήμα 4.1.

Η κάθε μονάδα του συστήματος μπορεί να καλέσει κάποια άλλη, και η αποτύπωση της κλήσης αυτής είναι το βέλος (β). Η αποτύπωση της αποστολής ή παραλαβής δεδομένων από τις μονάδες γίνεται με το βέλος με έναν κύκλο στο άκρο της αρχής του (γ) όταν γίνεται με μια κατεύθυνση ή με τα διπλά βέλη όταν υπάρχει αμφίδρομη αποστολή δεδομένων.



Σχήμα 4.1: *Συμβολισμοί Διαγράμματος Δομής*

Στα διαγράμματα αυτά υπάρχουν δύο βασικοί τύποι χαρακτηριστικών περιοχών. Αυτά είναι οι **κεντρικοί μετασχηματισμοί** και τα **κέντρα δοσοληψιών.**

Κεντρικός Μετασχηματισμός είναι ένα σύνολο επεξεργασιών που εκτελεί την επεξεργασία δεδομένων εισόδου και παράγει δεδομένα εξόδου. Τα δεδομένα εισόδου ετοιμάζονται από ένα σύνολο επεξεργασιών που βρίσκονται πριν από τον κεντρικό μετασχηματισμό. Τα δεδομένα εξόδου ετοιμάζονται από επεξεργασίες που ακολουθούν τον κεντρικό μετασχηματισμό Σχήμα 4.2.

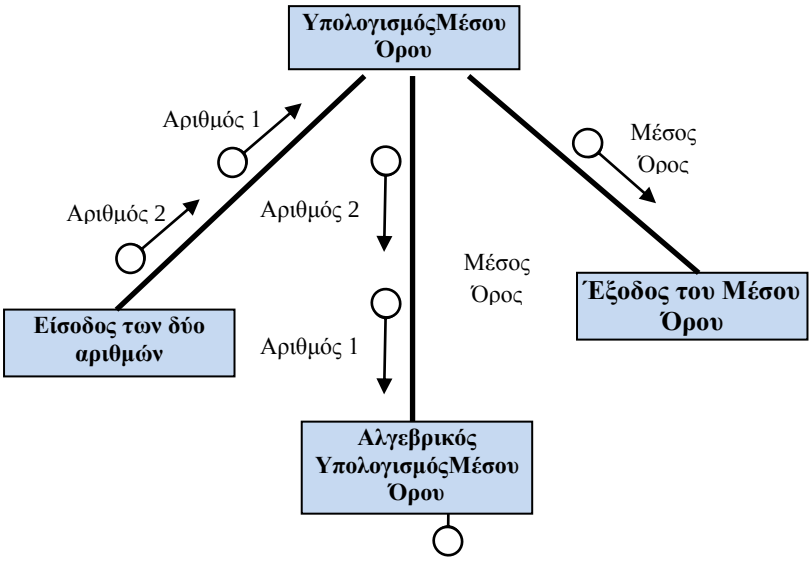
Σχήμα 4.2: *Κεντρικός Μετασχηματισμός Σχολικής Βιβλιοθήκης*

Να αναφέρουμε ότι οι μετασχηματισμοί (λειτουργίες) του παραπάνω σχήματος, ενδέχεται να είναι θύλακες άλλων εσωτερικών μετασχηματισμών. Συνεπώς, η Είσοδος και αρχική επεξεργασία Δεδομένων Εισόδου Μαθητή, όπως και η Επεξεργασία και η έξοδος Δεδομένων Εισόδου Μαθητή, μπορούν να διαιρεθούν σε άλλα σύνθετα τμήματα του διαγράμματος ροής δεδομένων και όχι σε απλούς, αυτόνομους μετασχηματισμούς.

Ο εντοπισμός και η πιθανή ανάλυση των μετασχηματισμών, δεν είναι μια αυτοματοποιημένη διαδικασία βημάτων την οποία θα μπορούσαμε να ακολουθήσουμε. Για να γίνει αυτό πρέπει να επιστρατεύσουμε την αναλυτική μας ικανότητα, την εμπειρία και τον αυτοσχεδιασμό μας. Το βασικό για την καλή δημιουργία των διαγραμμάτων δομής είναι η σύλληψη και η λεπτομέρεια του διαγράμματος ροής στο οποίο στηρίζεται.

4.2.2 Πρώτο Παράδειγμα

Ας θεωρήσουμε ένα απλό παράδειγμα για να καταλάβουμε την δομή ενός διαγράμματος δομής λειτουργιών ενός συστήματος. Έστω ότι θέλουμε να σχεδιάσουμε ένα σύστημα το οποίο να δέχεται δύο αριθμούς και να εξάγει τον μέσο όρο τους.



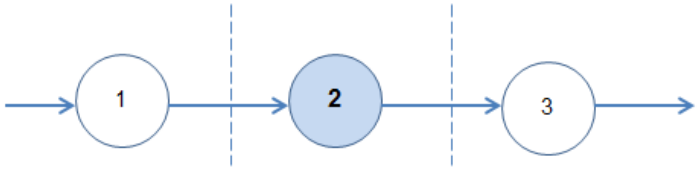
Σχήμα 4.3: *Διάγραμμα Δομής Υπολογισμού Μέσου Όρου*

Αν και το παραπάνω παράδειγμα αφορά την δομή μιας προγραμματιστικής μονάδας, και όχι ενός πληροφοριακού συστήματος, η θεώρησή του είναι βασική για την κατανόηση των συμβόλων και του τρόπου λειτουργίας του ΔΔ. Βλέπουμε στο σχήμα 4.3, ότι έχουμε έναν κεντρικό μετασχηματισμό (Υπολογισμός Μέσου Όρου) και δυο υπό-μονάδες για την είσοδο των δύο αριθμών, και την έξοδο του δεδομένου εξόδου. Είναι εύκολο να καταλάβουμε και τον συμβολισμό της μετακίνησης των δεδομένων.

4.2.3 Δεύτερο Παράδειγμα

Στο παράδειγμα αυτό θα δούμε την σχεδίαση ενός διαγράμματος δομής δεδομένων το οποίο αφορά την σχολική βιβλιοθήκη προηγούμενου κεφαλαίου.

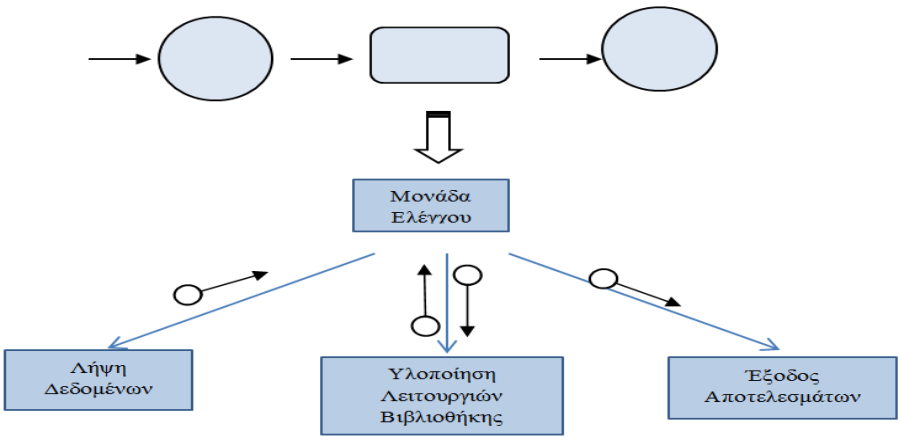
Ας αρχίσουμε με μια εναλλακτική απεικόνιση του κεντρικού μετασχηματισμού του σχήματος 4.4.



Σχήμα 4.4: *Διάγραμμα Κεντρικού Μετασχηματισμού*

Το διάγραμμα του σχήματος 4.4. αποτυπώνει την εντελώς βασική απεικόνιση του κεντρικού μετασχηματισμού. Η μονάδα ελέγχου εκτέλεσης (2), καλεί την μονάδα που απαιτείται για την είσοδο των δεδομένων (1) και την μονάδα (3) για την έξοδο των δεδομένων.

Στο σχήμα 4.5. βλέπουμε την μετατροπή του βασικού κεντρικού μετασχηματισμού σε διάγραμμα δομής, όπου φαίνονται οι απλοί μετασχηματισμοί που αντιστοιχούν στις μονάδες του συστήματος.



Σχήμα 4.5: *Αντιστοίχιση Κεντρικού Μετασχηματισμού σε διάγραμμα δομής*

Κέντρο Δοσοληψιών (transaction center). Σε ένα διάγραμμα ροής δεδομένων μια επεξεργασία χαρακτηρίζεται ως κέντρο δοσοληψιών όταν δέχεται δεδομένα από επεξεργασίες που τα προετοιμάζουν και βρίσκονται πριν από αυτό και διανέμει τη ροή σε διάφορες επεξεργασίες ανάλογα με την λογική του συστήματος.

Η διαφορά του με τον κεντρικό μετασχηματισμό είναι στο ότι είναι αδιαίρετη μονάδα όσον αφορά την λειτουργία του. Δηλαδή, ενώ ένας κεντρικός μετασχηματισμός μπορεί να αναλυθεί σε περισσότερους του ενός μετασχηματισμούς του διαγράμματος ροής δεδομένων, ένα κέντρο δοσοληψιών περιλαμβάνει μόνο μια λειτουργία η οποία δεν αναλύεται σε άλλες απλούστερες. Ένα κέντρο δοσοληψιών φαίνεται στο Σχήμα 4.6.

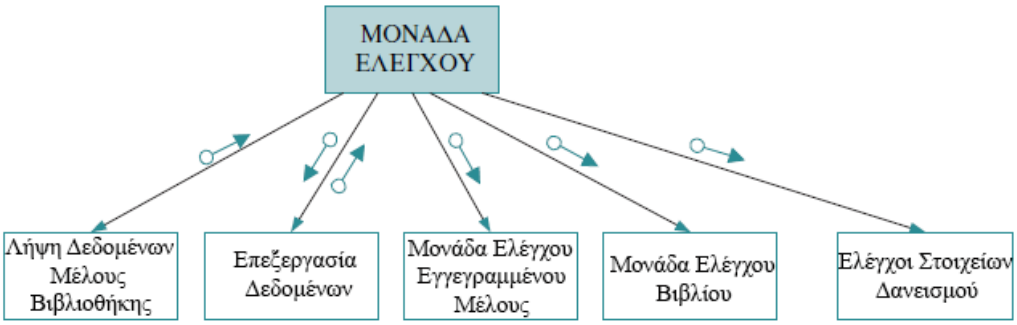
Σχήμα 4.6: *Ανάλυση Κεντρικού Μετασχηματισμού*

Χαρακτηριστική περίπτωση κέντρου δοσοληψιών είναι ένας μετασχηματισμός ελέγχου ροής προγράμματος, όπου, ανάλογα με την επιλογή του χρήστη, η ροή μεταφέρεται σε διαφορετικούς μετασχηματισμούς, όπως σε ένα μενού (βλέπε Σχήμα 4.7).

Στο σχήμα 4.7. η μονάδα ελέγχου του κέντρου δοσοληψιών καλεί δύο μονάδες για τη λήψη των δεδομένων εισόδου, και επεξεργασία αντίστοιχα, και μεταφέρει τον έλεγχο σε τόσες μονάδες, όσες είναι οι περιπτώσεις των δεδομένων εξόδου.

4.2.4 Βήματα κατασκευής διαγραμμάτων δομής

Για να μετατρέψουμε ένα διάγραμμα ροής σε ένα διάγραμμα δομής κάνουμε διαδοχικό διαμελισμό των λειτουργιών, με διάσχιση κατά πλάτος (BreathFirst), μέχρι να προσδιοριστούν και να απεικονιστούν όλοι οι μετασχηματισμοί που περιέχονται στο διάγραμμα ροής του συστήματος.



Σχήμα 4.7: *Κέντρο Δοσοληψίας σε διάγραμμα δομής*

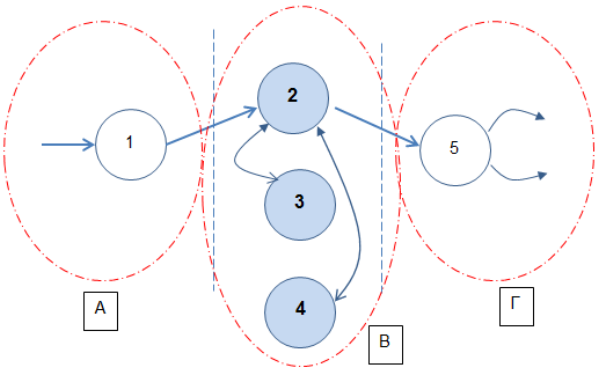
Τα βήματα αυτά είναι :

- **Εντοπίζουμε τον κεντρικό μετασχηματισμό.** Για κάθε τμήμα του ΔΡΔ, εντοπίζουμε τον κεντρικό μετασχηματισμό, και τους μετασχηματισμούς εισόδου και εξόδου.

- **Μετατρέπουμε τον κεντρικό μετασχηματισμό σε διάγραμμα δομής.** Δημιουργούμε το πρώτο επίπεδο του διαγράμματος δομής (ΔΔ) που αντιστοιχεί στον κεντρικό μετασχηματισμό.

- **Παραγοντοποίηση (factoring).** Κατασκευάζουμε τα διαγράμματα δομής για τους μετασχηματισμούς εισόδου και εξόδου, αριστερά, και δεξιά του κεντρικού. Κάθε ένας από αυτούς περιέχει μια μονάδα ελέγχου η οποία παραλαμβάνει τα δεδομένα εισόδου και μια άλλη μονάδα ελέγχου η οποία τα εξάγει στους αποδέκτες των δεδομένων, όπως είναι ο χρήστης, τα εξωτερικά συστήματα, οι συσκευές εξόδου (εκτυπωτής) και τα αρχεία. Η παραγοντοποίηση είναι μια αυτο-επαναλαμβανόμενη λειτουργία (recursive) για κεντρικούς μετασχηματισμούς που θα αναγνωριστούν στην πορεία με αποτέλεσμα να γίνει η ίδια διαδικασία ανάλυσης και για αυτούς. Η διαδικασία επαναλαμβάνεται μέχρι την πλήρη ανάλυση του συστήματος.

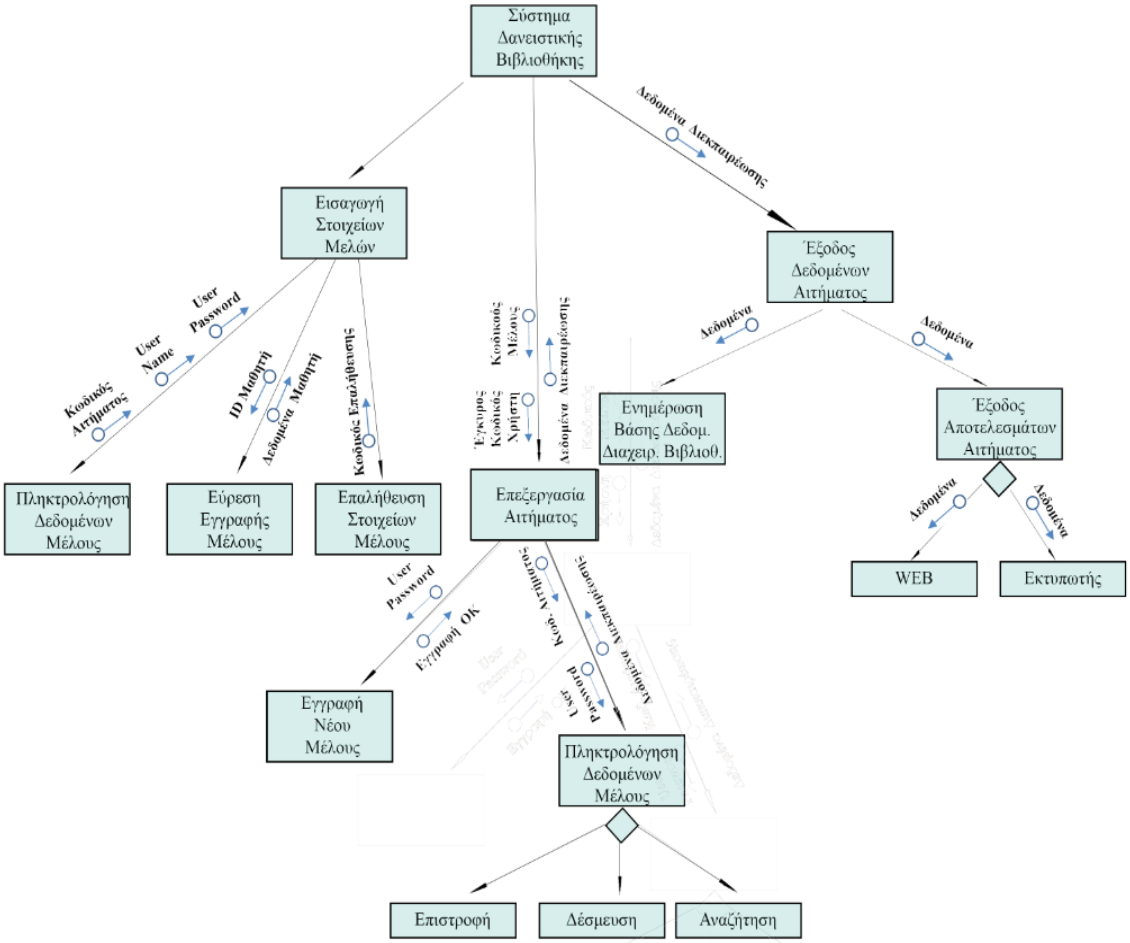
- **Συνένωση.** Για περιπτώσεις όπου τα δεδομένα εισόδου ή εξόδου δεν λαμβάνονται από τον χρήστη ή εξωτερικά συστήματα, τότε οι μονάδες αυτές αντικαθίστανται από μονάδες έλεγχου οι οποίες παρέχουν τα δεδομένα αυτά. Αν ο καλύτερος σχεδιασμός του διαγράμματος δομής απαιτεί διορθώσεις στο επίπεδο της ανάλυσης, το οποίο είναι το διάγραμμα ροής, τότε αυτό είναι χρήσιμο να γίνει τώρα και όχι αργότερα.



Σχήμα 4.8: *Κεντρικός Μετασχηματισμός για Σχολική Βιβλιοθήκη*

Τα τμήματα 2, 3 και 4 αποτελούν μέρη του κεντρικού μετασχηματισμού. Στη συνέχεια βασιζόμενοι στη σχήμα 4.8 δημιουργούμε την απεικόνιση του διαγράμματος δομής. Στο τμήμα Α, γίνεται εισαγωγή των στοιχείων του υποψήφιου χρήστη ή του υπάρχοντος μέλους της βιβλιοθήκης. Στη συνέχεια πληκτρολογούνται τα στοιχεία του και μεταφέρονται από την μονάδα ελέγχου Εισαγωγή Στοιχείων Μελών για εύρεση αν είναι ήδη μέλος της βιβλιοθήκης, και επιστρέφονται τα δεδομένα του (Όνομα, Βιβλία που δανείστηκε κλπ). Στη συνέχεια γίνεται επαλήθευση των στοιχείων του με χρήση της Αποθήκης Δεδομένων (ΒΔ).

Στο επόμενο βήμα, γίνεται επεξεργασία του αιτήματος του χρήστη/μέλους, η οποία είτε οδηγεί σε έγγραφή νέου μέλους, είτε οδηγεί σε διεκπεραίωση αιτήματος ήδη υπάρχοντος μέλους (Επιστροφή βιβλίου, Δέσμευση βιβλίου, Αναζήτηση Τίτλου κλπ). Όταν τελειώσει, τα δεδομένα – αποτελέσματα, προωθούνται από την κεντρική μονάδα ελέγχου στη μονάδα εξόδου. Γίνεται ενημέρωση της αποθήκης της βιβλιοθήκης και ενημερώνεται η εγγραφή του νέου ή παλιού μέλους με το αίτημα που είχε, και τα αποτελέσματά του. Επίσης η μονάδα εξόδου προωθεί τα αποτελέσματα του αιτήματος (Transaction), στο web μέσα από κάποια διαδικτυακή διεπαφή, είτε τα προωθεί στον εκτυπωτή αν κριθεί από τον διαχειριστή εμπρόσθιου άκρου (Front End Administrator).



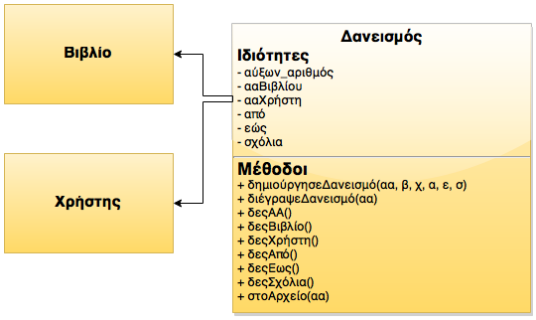
Σχήμα 4.9: *Συνολική Απεικόνιση του Διαγράμματος Δομής μιας Σχολικής Βιβλιοθήκης*

Η παραπάνω σχεδίαση του διαγράμματος δομής του συστήματος είναι καθαρά ενδεικτική. Σε καμία περίπτωση δεν είναι η μοναδική που μπορεί να γίνει, ούτε μπορεί να θεωρηθεί πλήρης από τον κάθε αναγνώστη. Το διάγραμμα αυτό μπορεί να τροποποιηθεί, να εμπλουτιστεί και να μεγαλώσει τόσο κατά πλάτος όσο και κατά ύψος. Αυτό εξαρτάται από πως θέλουμε να οργανώσουμε την λειτουργία της βιβλιοθήκης μας.

4.3 Σχεδιασμός προσανατολισμένος στα αντικείμενα

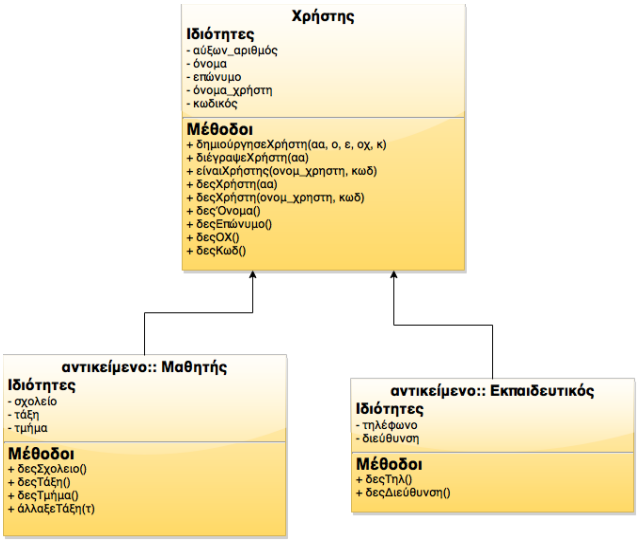
Στο 3ο κεφάλαιο παρουσιάσθηκε η αντικειμενοστραφής προσέγγιση στην ανάλυση ενός συστήματος. Συγκεκριμένα προσδιορίστηκε από τι αποτελούνται τα αντικείμενα, τι ακριβώς αντιπροσωπεύουν και πως μπορούν να εντοπιστούν στο πεδίο μιας εφαρμογής. Η φάση του σχεδιασμού ξεκινάει με αφετηρία τα ευρήματα της ανάλυσης που αφορούν τα αντικείμενα που έχουν εντοπιστεί σε ένα σύστημα και τις περιπτώσεις χρήσης. Στη συνέχεια τα αντικείμενα που έχουν εντοπιστεί σκιαγραφούνται με σκοπό να προσδιοριστεί το πως ακριβώς θα ικανοποιήσουν τις απαιτήσεις που υπάρχουν από το σύστημα και μοντελοποιούνται οι κλάσεις των

αντικειμένων. Λεπτομερώς ανατίθενται οι λειτουργίες, που εξάγονται από τις περιπτώσεις χρήσης, στα αντικείμενα και έτσι συνάγονται οι μέθοδοι των κλάσεων. Παράλληλα ορίζονται οι πληροφορίες που είναι απαραίτητο να διαχειρίζονται τα αντικείμενα και προσδιορίζονται οι ιδιότητές τους. Δημιουργούνται τα διαγράμματα των κλάσεων και ταξινομούνται. Αποτυπώνονται οι σχέσεις μεταξύ τους και εφαρμόζονται οι κανόνες της αντικειμενοστραφής σχεδίασης όπως η κληρονομικότητα.



Σχήμα 4.10 :*Η κλάση του δανεισμού*

Στο σχήμα 4.10 αποτυπώνεται η κλάση του «Δανεισμού» όπου δείχνει ότι ένα αντικείμενο δανεισμού έχει δύο σαφείς αναφορές. Η πρώτη είναι στο αντικείμενο που αντιπροσωπεύει το βιβλίο που θέλει να δανειστεί ο χρήστης και η δεύτερη στο αντικείμενο που αντιπροσωπεύει τον ίδιο τον χρήστη.



Σχήμα 4.11: *Η κλάση του χρήστη*

Η κληρονομικότητα χρησιμοποιείται στην αντικειμενοστραφή προσέγγιση όταν υπάρχει μια ομάδα από κλάσεις που έχουν κοινές ιδιότητες και μεθόδους. Στο παράδειγμα της σχολικής βιβλιοθήκης μια τέτοια ομάδα είναι ο χρήστης-μαθητής, ο χρήστης-εκπαιδευτικός κ.α. Σε αυτή την περίπτωση ορίζεται μια γενική κλάση, που ονομάζεται υπερκλάση και από την οποία κληρονομούν τις κοινές μεθόδους και ιδιότητες όλες οι κλάσεις της ομάδας χωρίς να χρειάζεται να τις ορίσουν ή να τις υλοποιήσουν εκ νέου. Στο σχήμα 4.11 φαίνονται η υπερκλάση «*Χρήστης*» και οι

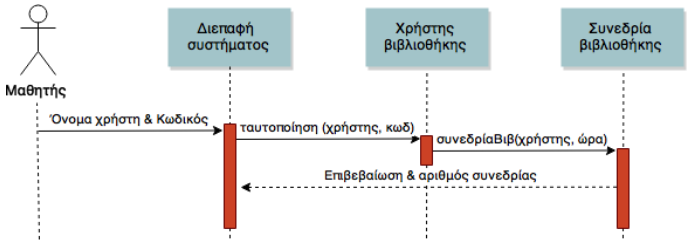
υποκλάσεις «*Μαθητής*» και «*Εκπαιδευτικός*» που κληρονομούν από την πρώτη κλάση. Η κλάση «*Χρήστης*» ορίζει τις κοινές ιδιότητες και μεθόδους που έχουν ο «*Μαθητής*» και ο «*Εκπαιδευτικός*» κι έτσι ο σχεδιαστής δεν χρειάζεται να τις ορίσει εκ νέου στις δυο υποκλάσεις.

Τελικό προϊόν της φάσης αυτής του σχεδιασμού είναι οι μέθοδοι και λειτουργίες των αντικειμένων, το πεδίο ευθύνης τους και ο τρόπος που συσχετίζονται και οργανώνονται.

Δραστηριότητα: Σε ένα ηλεκτρονικό σύστημα βιβλιοθήκης κρίνεται σκόπιμο να υπάρχει η δυνατότητα της κράτησης ενός βιβλίου. Να σχεδιάσετε την κλάση της κράτησης. Με ποιες κλάσεις συνδέεται;

4.3.1 Διαγράμματα ακολουθίας

Τα διάγραμμα ακολουθίας χρησιμοποιούνται κατά τη διάρκεια της φάσης σχεδιασμού που υιοθετεί το αντικειμενοστραφή μοντέλο προσέγγισης. Απεικονίζουν τον τρόπο με τον οποίο τα αντικείμενα του συστήματος αλληλεπιδρούν μεταξύ τους. Γενικά απεικονίζουν τον κύκλο ζωής των αντικειμένων και τον τρόπο που διεξάγονται τα σενάρια λειτουργικότητας. Δίνουν έμφαση στην χρονική στιγμή που δημιουργούνται αντικείμενα και στην ακολουθία που ανταλλάσουν μηνύματα για να επικοινωνήσουν μεταξύ τους. Έτσι γίνεται καλύτερη κατανόηση της λειτουργίας των αντικειμένων και επανεξετάζονται τυχόν θέματα που προκύπτουν.



Σχήμα 4.12: *Διάγραμμα ακολουθίας είσοδος μαθητή στο σύστημα*

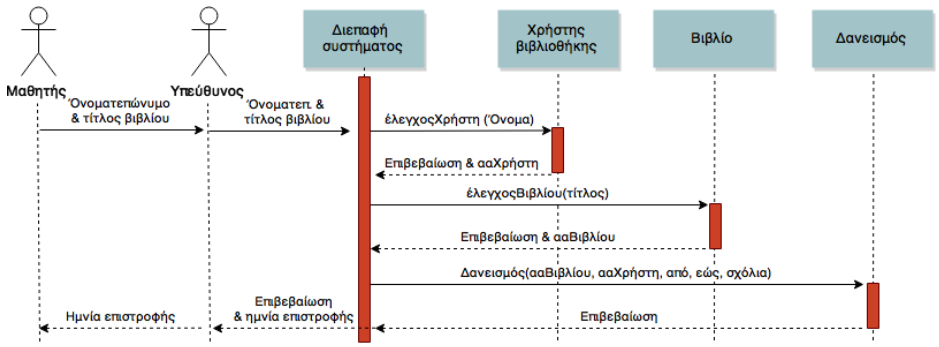
Το διάγραμμα ακολουθίας του σχήματος 4.12 δείχνει την λειτουργία εισαγωγής του μαθητή στο σύστημα της βιβλιοθήκης. Οι κάθετες διακεκομμένες γραμμές αναπαριστούν τον χρόνο ενώ οριζόντια διατάσσονται τα διάφορα αντικείμενα και οι πράκτορες-ηθοποιοί του συστήματος. Στο παράδειγμα φαίνονται ο μαθητής και τα αντικείμενα «*Διεπαφή συστήματος*», «*Χρήστης βιβλιοθήκης*», «*Συνεδρία βιβλιοθήκης*». Τα μικρά ορθογώνια κουτάκια πάνω στις γραμμές χρόνου αναπαριστούν την διάρκεια εκτέλεσης των λειτουργιών από τα αντικείμενα. Τα αντικείμενα επικοινωνούν μεταξύ τους με ανταλλαγή μηνυμάτων που αναπαριστάνονται με τα βελάκια.

Ο μαθητής αλληλεπιδρά με το αντικείμενο που διαχειρίζεται την διεπαφή χρήστη και εισάγει τα στοιχεία όνομα χρήστη και κωδικό. Το σύστημα δημιουργεί το αντικείμενο

του χρήστη και ελέγχει την ορθότητα των στοιχείων. Αν τα στοιχεία είναι σωστά τότε δημιουργείται ένα αντικείμενο συνεδρίας με το όνομα του χρήστη και την ώρα εισαγωγής στο σύστημα. Επιστρέφεται στη διεπαφή ο κωδικός-αναγνωριστικό της συνεδρίας και ένα μήνυμα επιβεβαίωσης που πιστοποιεί ότι έγινε κανονικά η είσοδος στο σύστημα. Το αντικείμενο συνεδρίας είναι ενεργό για όσο χρόνο ο χρήστης πλοηγείται στην εφαρμογή. Διαγράφεται με την έξοδο του χρήστη ή μετά από μία ώρα αδράνειας της συνεδρίας.

Στο σχήμα 4.13 φαίνεται το διάγραμμα ακολουθίας δανεισμού βιβλίου στη σχολική βιβλιοθήκη.

Δραστηριότητα:: Ποιους ηθοποιούς αναγνωρίζετε στο σύστημα και ποια αντικείμενα. Να εξηγήσετε το σενάριο που απεικονίζεται στο σχήμα 4.12



Σχήμα 4.13: *Διάγραμμα ακολουθίας δανεισμού*

Δραστηριότητα: Να δημιουργήσετε ένα διάγραμμα ακολουθίας για την επιστροφή ενός βιβλίου στη σχολική βιβλιοθήκη.

4.4 Σχεδιασμός διεπαφής χρήστη

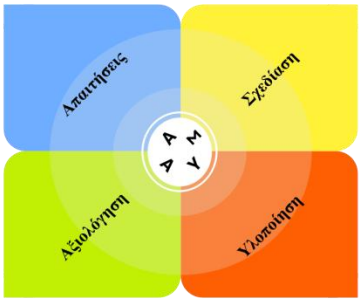
Η διεπαφή χρήστη είναι το τμήμα του πληροφοριακού συστήματος που έρχεται σε επαφή και αλληλεπιδρά με τον χρήστη. Η σωστή σχεδίαση της διεπαφής χρήστη είναι πολύ σημαντική για το πληροφοριακό σύστημα καθώς θα πρέπει να αναδεικνύει όλες τις δυνατότητες που προσφέρει το σύστημα και παράλληλα να ανταποκρίνεται στις ικανότητες, στις ιδιαιτερότητες και στις εμπειρίες του χρήστη.

4.4.1 Πρωτότυπα

Μια μέθοδος για την ανάπτυξη της διεπαφής χρήστη είναι η δημιουργία πρωτοτύπων. Τα πρωτότυπα είναι απλοποιημένα μοντέλα του συστήματος που δημιουργούνται με σκοπό την έγκαιρη εκτίμηση κάποιων χαρακτηριστικών που θα υπάρχουν στο τελικό προϊόν.

Τα πρωτότυπα χρησιμοποιούνται σε πολλούς τομείς. Στην αυτοκινητοβιομηχανία είναι μινιατούρες των αυτοκινήτων, στον κατασκευαστικό τομέα είναι οι μικρογραφίες κτιρίων, πάρκων κ.α, στη ζωγραφική είναι τα δοκιμαστικά σκίτσα. Στα πληροφοριακά συστήματα είναι μια αναπαράσταση ενός μέρους του συστήματος.

Για τη δημιουργία των πρωτοτύπων ακολουθείται ένας μικρός κύκλος ζωής εντός της φάσης του σχεδιασμού. Αρχικά εξετάζονται οι απαιτήσεις που έχουν οριστεί στην προηγούμενη φάση τους κύκλου ζωής του συστήματος. Στη συνέχεια ακολουθεί η σχεδίαση του πρωτοτύπου όπου επιλέγονται τα εργαλεία που θα χρησιμοποιηθούν, το είδος πρωτοτύπου που θα αναπτυχθεί, το στυλ και τα χαρακτηριστικά του. Την σχεδίαση ακολουθεί η υλοποίηση όσων προσδιορίστηκαν στον σχεδιασμό με τη χρήση των κατάλληλων εργαλείων. Στο τελικό στάδιο γίνεται η αξιολόγηση του πρωτοτύπου όπου έχει σχέση με τον έλεγχο των χαρακτηριστικών τα οποία προάγει. Η αξιολόγηση ενδεχομένως να αναδείξει προβλήματα που θα οδηγήσουν στον πρώτο στάδιο του κύκλου ζωής του πρωτοτύπου και στην εξέλιξή του από την αρχή.



Σχήμα 4.14: *Κύκλος ζωής πρωτοτύπου*

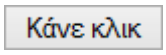
Η χρήση πρωτοτύπων έχει πολλά πλεονεκτήματα. Οι σχεδιαστές έχουν τη δυνατότητα να αναπαραστήσουν τις ιδέες τους με ένα τρόπο που να ευνοείται η επεξεργασία τους και η δημιουργία εναλλακτικών λύσεων. Έτσι διαμορφώνονται οι απαραίτητες συνθήκες για τον έλεγχο των εναλλακτικών λύσεων πριν τη δέσμευση σε κάποια από αυτές και την εξέλιξη του πρωτοτύπου. Εξάλλου η χρήση των πρωτοτύπων δημιουργεί ένα πλαίσιο επικοινωνίας ανάμεσα στους σχεδιαστές και στους ενδιαφερομένους. Οι ενδιαφερόμενοι σε αυτή την περίπτωση μπορεί να είναι οι πελάτες, οι χρήστες του συστήματος και οι υπεύθυνοι αξιολογητές του έργου. Παρέχεται λοιπόν στους ενδιαφερόμενους η δυνατότητα να έχουν έγκαιρα μία αντίληψη σχετικά με τα χαρακτηριστικά και τη φύση του τελικού προϊόντος. Επιπρόσθετα μπορούν να συμμετέχουν στην αξιολόγηση των νέων ιδεών σχεδίασης ή ακόμα και να υποδείξουν λύσεις για την βελτίωσή τους. Ανάλογα με το κόστος και το είδος των πρωτοτύπων και τις συνθήκες συνεργασίας θα μπορούσαν να επιλέξουν από μια σειρά εναλλακτικών πρωτοτύπων και σχεδίων που θα εκφράζουν διαφορετικές προσεγγίσεις και στυλ.

Απόρροια όλων των παραπάνω είναι η ελαχιστοποίηση των προβλημάτων που μπορεί να εμφανιστούν μετά την υλοποίηση του προϊόντος. Τυχών προβλήματα και θέματα που μπορεί να υπάρχουν είναι πολύ πιθανό να εντοπιστούν κατά τη διάρκεια χρήσης και αξιολόγησης των πρωτοτύπων γεγονός που δίνει ευελιξία στην ομάδα σχεδιασμού να επεξεργαστεί και να αναμορφώσει τις λύσεις και να ελαχιστοποιήσει το κόστος.

4.4.2 Αντικείμενα διεπαφών

Η διεπαφή του χρήστη είναι μια οπτική σύνθεση από γραφικά αντικείμενα τα οποία εμφανίζονται στο οπτικό μέσο και χρησιμοποιούνται για να ικανοποιήσουν την αμφίδρομη σχέση χρήστη και μηχανής. Τα αντικείμενα αυτά χρησιμοποιούνται για τη δημιουργία συμβάντων και για την έξοδο ή είσοδο πληροφοριών.

Κουμπιά εντολών – ActionButtons:



Τα κουμπιά εντολών παρουσιάζουν τις βασικές ενέργειες μιας διεπαφής και δίνουν τη δυνατότητα στους χρήστες να τις εκτελέσουν.

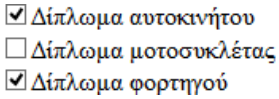
```
<button type="button" onclick="alert('Γειασας!')">Κάνεκλικ</button>
```

Πλαίσια Κειμένου – TextBoxes:

Τα πλαίσια κειμένου παρέχουν τον βασικό τρόπο για να εισάγουν δεδομένα οι χρήστες πληκτρολογώντας μέσα στο πλαίσιο. Χρησιμοποιούνται και για προβολή δεδομένων δίνοντας παράλληλα τη δυνατότητα επεξεργασίας στον χρήστη.

```
Όνομα: <input type="text" name="firstname">
```

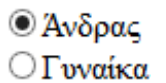
Κουμπιά Επιλογών – CheckBoxes:



Τα κουμπιά επιλογών χρησιμοποιούνται για την εισαγωγή δεδομένων τύπου διακόπτη για την εισαγωγή χαρακτηριστικών ή για την μη εισαγωγή αυτών.

```
<input type="checkbox" name="cdl" value="cdl">Δίπλωμα αυτοκινήτου<br>
<input type="checkbox" name="mdl" value="mdl">Δίπλωμα μοτοσυκλέτας<br>
<input type="checkbox" name="mdv" value=" mdv">Δίπλωμα φορτηγού
```

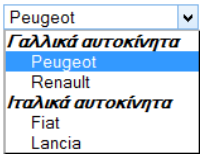
Ράδιο-Κουμπιά Επιλογών – RadioButtons:



Τα ράδιο-κουμπιά επιλογών χρησιμοποιούνται με τον ίδιο τρόπο που χρησιμοποιούνται τα κουμπιά επιλογών με τη διαφορά ότι θεωρούνται μια ομάδα και κάθε φορά επιλέγεται μόνο μία επιλογή από την ομάδα.

```
<input type="radio" name="s" value="male" checked>Άνδρας<br>

<input type="radio" name="s" value="female">Γυναίκα
```

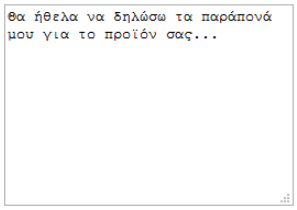
Λίστα Επιλογών – ListBoxes:



Η λίστα επιλογών δίνει τη δυνατότητα στους χρήστες να επιλέγουν δεδομένα από μία λίστα επιλογών. Προαιρετικά ο προγραμματιστής έχει την δυνατότητα να ομαδοποιεί τις επιλογές που προσφέρει στον χρήστη.

```
<select>
<optgrouplabel="Γαλλικά αυτοκίνητα">
<option value="p">Peugeot</option>
<option value="r">Renault</option>
</optgroup>
<optgrouplabel="Ιταλικά αυτοκίνητα">
<option value="f">Fiat</option>
<option value="l">Lancia</option>
</optgroup>
</select>
```

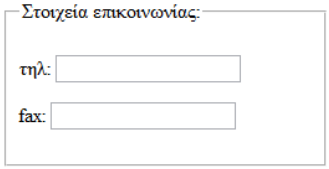
Περιοχή Κειμένου – TextArea:



Η περιοχή κειμένου χρησιμοποιείται, όπως και τα πλαίσια κειμένου, για την είσοδο και έξοδο δεδομένων με τη διαφορά ότι επιτρέπει πολλαπλές γραμμές κειμένου να εισαχθούν ή να παρουσιαστούν.

```
<textarea name="message" rows="10" cols="30">

Θα ήθελα να δηλώσω τα παράπονά μου για το προϊόν σας...</textarea>
```

Πλαίσια – Fieldsets:



Τα πλαίσια δεν προσφέρουν κάποια λειτουργικότητα. Παρόλα αυτά είναι σημαντικά γιατί επιτρέπουν την ομαδοποίηση διαφόρων αντικειμένων των διεπαφών και έτσι εξυπηρετούν στον καλύτερο σχεδιασμό μιας διεπαφής. Κάθε πλαίσιο έχει ένα όνομα που το προσδιορίζει και ονομάζεται «Legend».

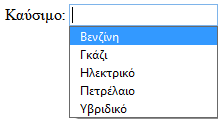
```
<fieldset>
<legend>Στοιχείαεπικοινωνίας:</legend><br>
τηλ: <input type="text"><p>
  fax: <input type="text"><br>
</fieldset>
```

Μπάρες ολίσθησης – Slider:

Οι μπάρες ολίσθησης επιτρέπουν την εισαγωγή αριθμητικών δεδομένων από ένα μεγάλο πεδίο τιμών.

Βαθμός: <input type="range" id="a" name="a" value="50">
Σημείωση:Για να εμφανίζεται η τιμή δίπλα στην μπάρα θα πρέπει να χρησιμοποιηθεί η εντολή <output> και η ιδιότητα oninput της φόρμας <form>: <form oninput="x.value=parseInt(a.value)"> Βαθμός: <input type="range" id="a" name="a" value="50"> <output name="x" for="a"></output></form>

Λίστες δεδομένων – DataList:



Οι λίστες δεδομένων επιτρέπουν την εισαγωγή δεδομένων. Καθώς ο χρήστης εισάγει δεδομένα εμφανίζεται μια λίστα επιλογών από την οποία ο χρήστης μπορεί να κάνει μια επιλογή προαιρετικά.

```
Καύσιμο: <input list="ka" name="ka">
<datalist id="ka">
<option value="Βενζίνη">
<option value="Γκάζι">
<option value="Ηλεκτρικό">
<option value="Πετρέλαιο">
<option value="Υβριδικό">
</datalist>
```

4.4.3 Βασικές αρχές και λάθη

Ο αισθητικός σχεδιασμός είναι βασικός πυλώνας μιας επιτυχημένης διεπαφής χρήστη. Η καλή αισθητική μιας διεπαφής κεντρίζει το ενδιαφέρον του χρήστη, του προκαλεί όμορφα συναισθήματα και τον προδιαθέτει θετικά. Σχετίζεται με ένα σύνολο παραμέτρων που επηρεάζουν το οπτικό αποτέλεσμα της διεπαφής. Οι πιο σημαντικές από αυτές τις παραμέτρους είναι τα χρώματα, η διάταξη των αντικειμένων και η κατάλληλη μορφοποίηση του κειμένου.

Δραστηριότητα: Επισκεφθείτε τις παρακάτω ιστοσελίδες:

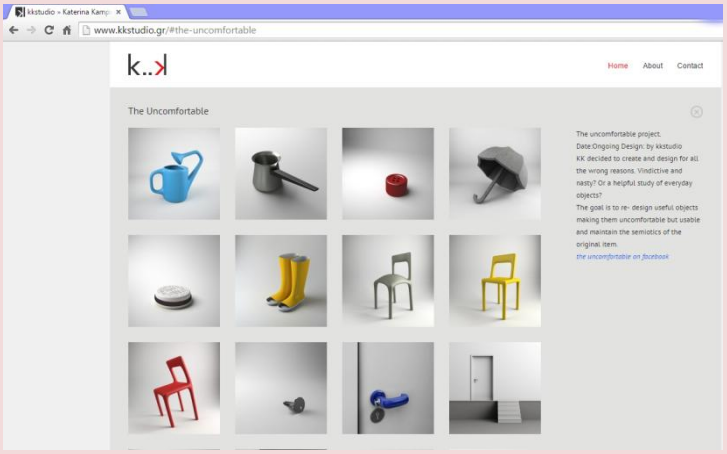
- http://www.nytimes.com/
- http://www.google.gr/
- http://www.oracle.com/

Πως κρίνετε την χρήση χρωμάτων, την διάταξη των αντικειμένων και την μορφοποίηση του κειμένου;

Βρείτε και προτείνετε ιστοσελίδες όπου σας ικανοποιεί ο αισθητικός σχεδιασμός. Εξηγήστε τον λόγο που τις προτιμήσατε.

Ο αισθητικός σχεδιασμός είναι καθοριστικός για την πρώτη εντύπωση που δίνει μια εφαρμογή στον χρήστη. Ωστόσο μια διεπαφή θα πρέπει να είναι χρηστική για να θεωρηθεί πετυχημένη. Αυτό αποτυπώνεται πλήρως σε διάφορα παραδείγματα που βλέπουμε στην καθημερινή ζωή.

Δραστηριότητα: Μεταβείτε στην ιστοσελίδα «http://www.kkstudio.gr/» της αρχιτέκτων Κατερίνας Καμπράνη και επισκεφθείτε την συλλογή της «The Uncomfortable». Ποια αντικείμενα σας αρέσουν αισθητικά; Βρίσκετε τα αντικείμενα χρηστικά; Ποιο μήνυμα πιστεύετε ότι θέλει να περάσει η σχεδιάστρια με αυτή την συλλογή της;



Εικόνα 4.15: *Στιγμιότυπο οθόνης από την ιστοσελίδα της ΚατερίναςΚαμπράνη (τελευταία επίσκεψη 10/7/2015)*

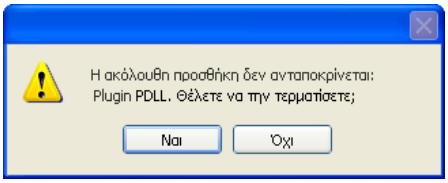
Οι σχεδιαστές διεπαφών πολλές φορές δεν ακολουθούν συγκεκριμένες αρχές αλλά ενεργούν βιωματικά σύμφωνα με τις εμπειρίες τους. Ωστόσο πολλοί ερευνητές, όπως για παράδειγμα οι Jacob Nielsen, Bruce Tognazzini και Ben Shneiderman, έχουν διακρίνει στον σχεδιασμό διεπαφών κάποιες βασικές ιδιότητες και έχουν προτείνει κάποιους κανόνες για την αξιολόγησή τους. Η κατανόηση αυτών των κανόνων και η τήρησή τους κατά τη διάρκεια του σχεδιασμού είναι πολύ πιθανό να οδηγήσουν σε μια επιτυχημένη σχεδίαση.

Οι κανόνες του Nielsen

Ο Jacob Nielsen διατύπωσε δεκατέσσερις ευριστικούς κανόνες για την αξιολόγηση μιας διεπαφής. Πολλοί από αυτούς τους κανόνες έχουν εφαρμογή σε όλα τα είδη των διεπαφών.

- **Αντιστοιχία μεταξύ του συστήματος και του πραγματικού κόσμου**

  Το σύστημα θα πρέπει να ταιριάζει στο νοητικό επίπεδο και την ιδιοσυγκρασία του χρήστη και να ανταποκρίνεται στις προσδοκίες του. Δεν θα πρέπει να χρησιμοποιεί ορολογία και ιδέες που δεν γνωρίζει ο χρήστης:

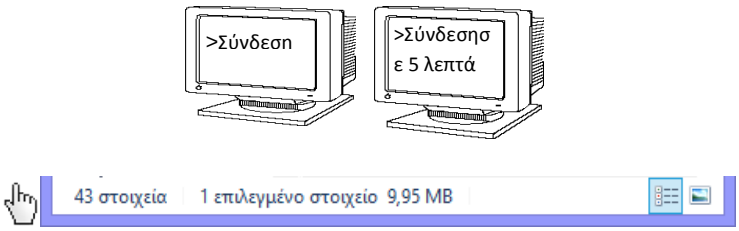


Εικόνα 4.16.: *Ένα μήνυμα που δεν καταλαβαίνει ο μέσος χρήστης*

Αντίθετα όταν η εφαρμογή ανήκει σε κάποιο συγκεκριμένο πεδίο π.χ. ιατρική δεν θα πρέπει να παραβλέπει την χρήση όρων από αυτή την επιστήμη.

- **Ορατή κατάσταση του συστήματος**

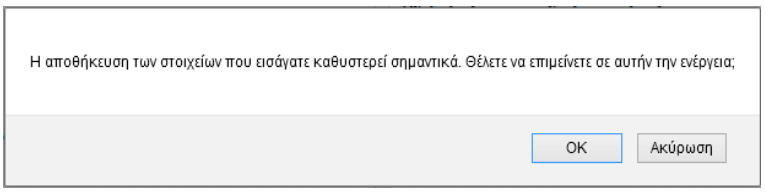
  Το σύστημα θα πρέπει να ενημερώνει ανά πάσα στιγμή τους χρήστες για το τι συμβαίνει, με την κατάλληλη ανατροφοδότηση μέσα σε συγκεκριμένο χρονικό διάστημα. Αυτό επιτυγχάνεται με διάφορες τεχνικές όπως η αλλαγή του κέρσορα πάνω σε συγκεκριμένες επιλογές, η ενημέρωση μέσα από τη γραμμή ελέγχου και η επισήμανση αντικειμένων που επιλέγονται.



Εικόνα 4.17.: *Παραδείγματα όπου η διεπαφή δίνει ανατροφοδότηση για την κατάσταση του συστήματος.*

- **Έλεγχος και ελευθερία του χρήστη**

  Συχνά δημιουργούνται προβληματικές καταστάσεις στο σύστημα λόγω των χρηστών που κάνουν κάποια επιλογή εκ παραδρομής ή λόγω άλλων αστάθμητων παραγόντων π.χ. αδυναμία σύνεσης στο Διαδίκτυο. Σε τέτοιες καταστάσεις είναι απαραίτητοι για τους χρήστες τρόποι για να αφήσουν εύκολα κα γρήγορα την ανεπιθύμητη κατάσταση.



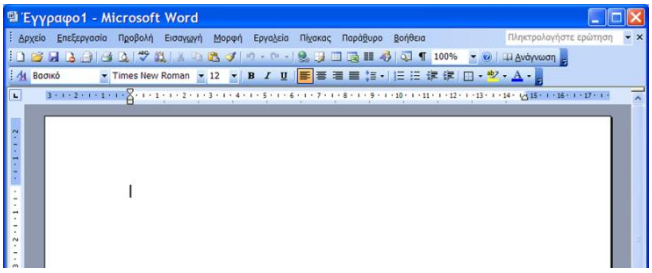
Εικόνα 4.18 : *Διάλογος που δίνει στον χρήστη την ελευθερία να παραιτηθεί από μια χρονοβόρα ενέργεια.*

- **Συνέπεια και συμμόρφωση με πρότυπα**

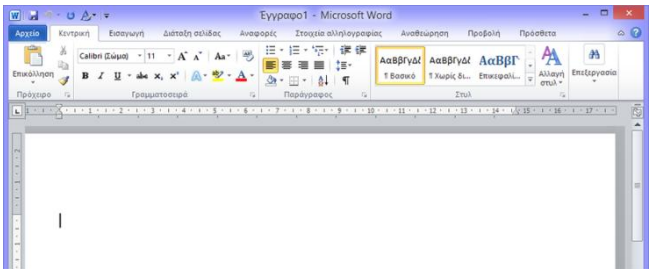
  Οι χρήστες δεν θα πρέπει να αναρωτιούνται αν διαφορετικές επιλογές, πληροφορίες ή καταστάσεις σημαίνουν το ίδιο πράγμα.

Εικόνα 4.19. :*Διαφορετικές οθόνες που δείχνουν έλλειψη συνέπειας*

Στην εικόνα 4.19 φαίνονται τρεις διαφορετικές οθόνες όπου τα κουμπιά «ΟΚ»και «Ακύρωση» τοποθετούνται σε διαφορετικά μέρη της διεπαφής. Η τοποθέτηση των κουμπιών σε διαφορετικές θέσεις και μάλιστα στο ίδιο πρόγραμμα είναι δείγμα έλλειψης συνέπειας και μπερδεύει τους χρήστες.



Εικόνα 4.20: *Στιγμιότυπο οθόνης του MicrosoftWord 2003*

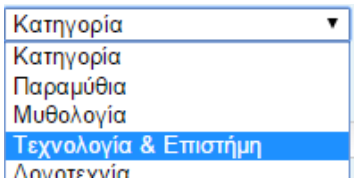


Εικόνα 4.21: *Στιγμιότυπο οθόνης του MicrosoftWord 2007*

Στις εικόνες 4.20 και 4.21 φαίνονται οι οθόνες του MicrosoftWord 2003 και 2007. Η Microsoft παραβίασε τον κανόνα της συνέπειας αφού προχώρησε σε θεαματικές αλλαγές στην διεπαφή από την μία έκδοση του προγράμματος στην άλλη. Στην έκδοση του 2007 το μενού δεν υπάρχει και οι μπάρες που περιέχουν τα εργαλεία έχουν αλλάξει δραστικά.

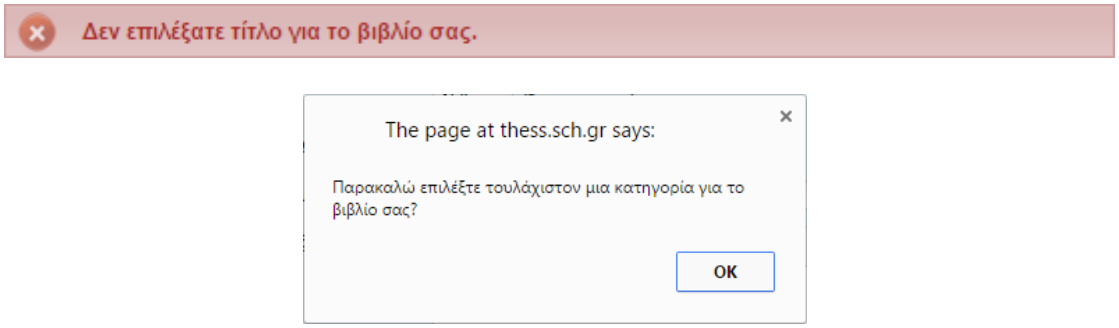
- **Πρόληψη σφαλμάτων**

Το σύστημα θα πρέπει να είναι έτσι σχεδιασμένο ώστε να αποτρέπει τον χρήστη από το να κάνει λάθη. Έτσι για παράδειγμα στην εισαγωγή δεδομένων είναι προτιμότερο για τον χρήστη να επιλέγει από μία λίστα δεδομένων παρά να τα πληκτρολογεί. Τα αντικείμενα των διεπαφών που προσφέρουν έτοιμες επιλογές στον χρήστη όπως η λίστα επιλογών της εικόνας 4.22 είναι προτιμότερα από τα αντίστοιχα αντικείμενα που τον υποχρεώνουν να πληκτρολογεί πληροφορίες όπως τα πλαίσια κειμένου.



Εικόνα 4.22.:*Το αντικείμενο διεπαφής λίστα επιλογών*

Επίσης είναι προτιμότερο ο χρήστης να ελέγχεται όσο το δυνατόν πιο γρήγορα για τυχόν παραλήψεις του. Όπου υπάρχει η δυνατότητα τα σφάλματα θα πρέπει να αναγνωρίζονται εγκαίρως από το σύστημα, να πληροφορούν τον χρήστη (εικόνα 4.23) και να του δίνουν την ευκαιρία να τα διορθώσει πριν προχωρήσει σε κάποια καταχώρηση.



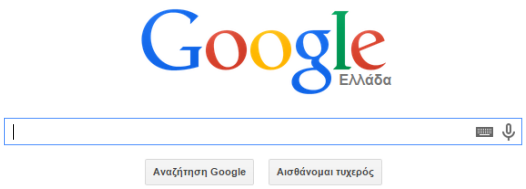
Εικόνα 4.23: *Μηνύματα που ενημερώνουν τον χρήστη για παραλήψεις*

- **Αναγνώριση αντί της ενθύμησης**

  Οι επιλογές και όλα τα στοιχεία και θα πρέπει να είναι ορατές ώστε ο χρήστης να μην είναι αναγκασμένος να θυμάται πληροφορίες από ένα τμήμα του διαλόγου σε ένα άλλο.

- **Αισθητική και μινιμαλιστική σχεδίαση:**

  Η σχεδίαση θα πρέπει να ακολουθεί τους κανόνες «*Η απλότητα είναι χρυσός*» και «*Οτιδήποτε τοποθετείται στην διεπαφή του χρήστη θα πρέπει να έχει ένα καλό λόγο για να είναι εκεί*». Όλες οι πληροφορίες που περιέχονται στις διεπαφές θα πρέπει να είναι χρήσιμες για τον χρήστη.



Εικόνα 4.24 : *Η οθόνη του Google*

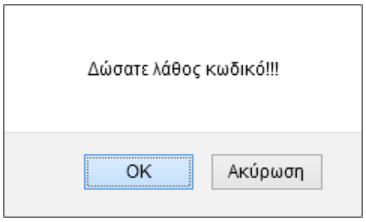
Ένα παράδειγμα εξαιρετικής απλότητας είναι η μηχανή αναζήτησης της Google εικόνα 4.24. Το περιβάλλον του είναι όσο πιο απλό γίνεται. Περιττά αντικείμενα δεν υπάρχουν και χρησιμοποιεί μεγάλη επιφάνεια σε λευκό. Σαν αποτέλεσμα ο χρήστης το θεωρεί πολύ εύκολο στη χρήση, ο σχεδιασμός του φαίνεται κάτι οικείο σαν να το ήξερε πάντα και φορτώνει πολύ γρήγορα.

Εικόνα 4.25: *Δύο διαφορετικές γραμμές εργαλείων*

Οι γραμμές εργαλείων δείχνουν τη διαφορά ανάμεσα σε έναν φορτωμένο και ένα μινιμαλιστικό σχεδιασμό. Η πρώτη γραμμή εργαλείων είναι γεμάτη από πολλά φανταχτερά χρώματα που αποσπούν την προσοχή και καθιστούν δύσκολη την ανάγνωσή της. Η δεύτερη γραμμή εργαλείων, από το πρόγραμμα Microsoft Office 2003, χρησιμοποιεί μόνο συγκεκριμένα χρώματα σε κατάλληλους συνδυασμούς. Τα εικονίδια έχουν ομαδοποιηθεί και έχουν κενό ανάμεσά τους έτσι για να διακρίνονται πολύ εύκολα και ακολουθούν τα πρότυπα που είναι γνωστά στον χρήστη.

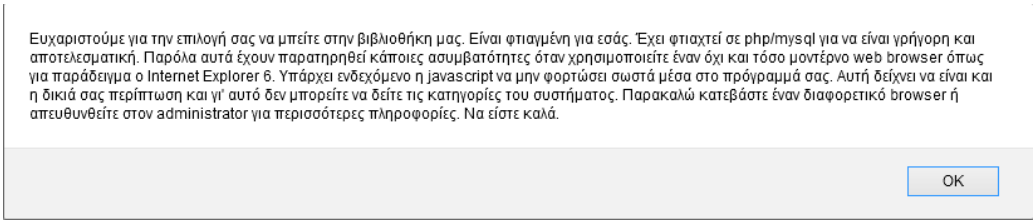
- **Αναφορά σφαλμάτων**

  Το σύστημα θα πρέπει να βοηθάει τους χρήστες να αναγνωρίσουν τα σφάλματά τους, να προβαίνουν σε διάγνωση της αιτίας τους και να ανακάμπτουν από αυτά.



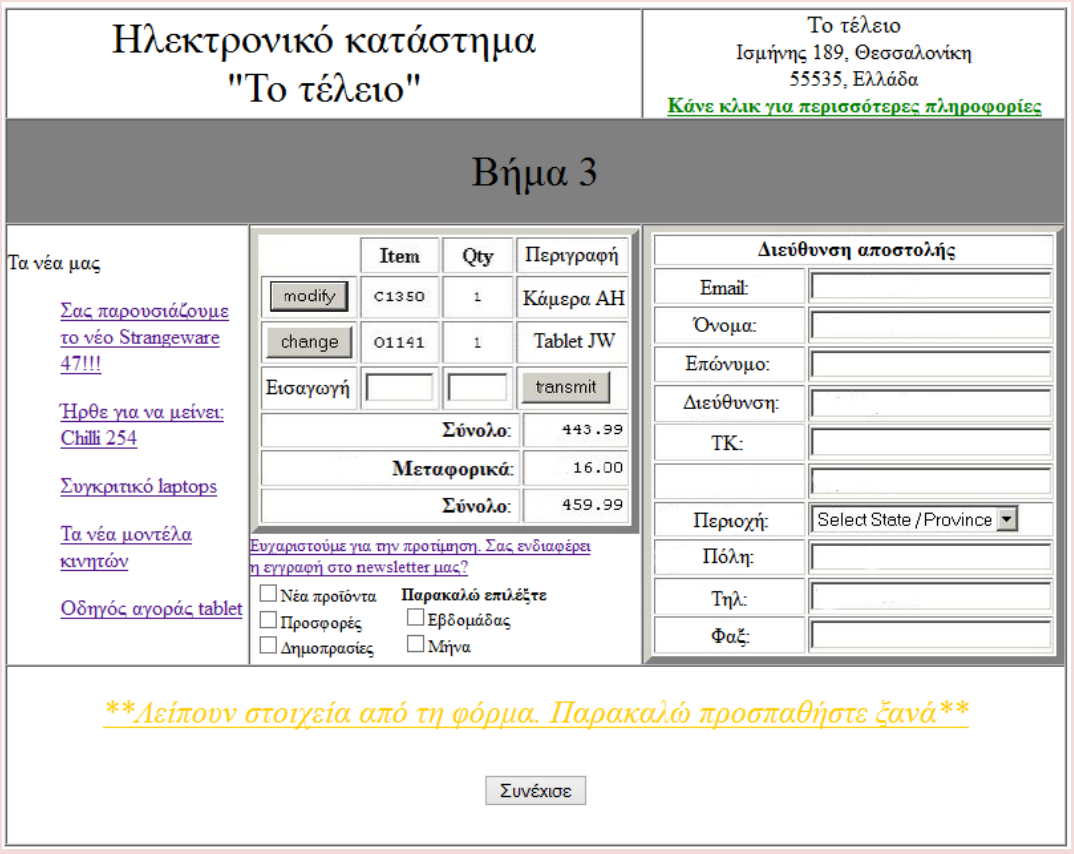
Εικόνα 4.26: *Μήνυμα σφάλματος*

Τα μηνύματα σφαλμάτων θα πρέπει να εκφράζονται σε γλώσσα κατανοητή από τον χρήστη, να εκθέτουν με ακρίβεια το πρόβλημα και όπου είναι δυνατόν να προτείνουν κάποια λύση.



Εικόνα 4.27: *Μια αποτυχημένη αναφορά σφάλματος*

Δραστηριότητα: Στην εικόνα που ακολουθεί φαίνεται η διεπαφή από το ηλεκτρονικό κατάστημα «Το τέλειο». Η συγκεκριμένη οθόνη παραβιάζει σχεδόν όλου τους κανόνες του Nielsen. Να εντοπίσετε που ακριβώς παραβιάζονται αυτοί οι κανόνες. Να γίνουν προτάσεις για την βελτίωση της χρηστικότητας της διεπαφής.



Εικόνα 4.28: *Ηλεκτρονικό κατάστημα «το τέλειο»*

Παράδειγμα:

Κανόνας που παραβιάζεται	**Σημείο παραβίασης**	**Πρόταση**
Αναγνώριση αντί της ενθύμησης	*Στην κεντρική φόρμα της οθόνης με τα αντικείμενα προς αγορά δίνεται η δυνατότητα στον χρήστη να προσθέσει ένα αντικείμενο βάζοντας τον κωδικό στην στήλη «Item» και την ποσότητα στην στήλη «Qty». Σε αυτή την περίπτωση ο χρήστης θα πρέπει να θυμάται τον κωδικό του αντικειμένου που πρέπει να συμπληρώσει στο καλάθι του.*	*Να διαγραφεί η συγκεκριμένη λειτουργία και να δίνεται η δυνατότητα στον χρήστη να επιστρέφει στη λίστα με τα προϊόντα για να προσθέσει και άλλα αντικείμενα στο καλάθι του.*

Δραστηριότητα: Αναζητήστε στο Διαδίκτυο τους κανόνες του Bruce Tognazzini και Ben Shneiderman. Βρείτε ομοιότητες και διαφορές με τους κανόνες του Nielsen.

Ερωτήσεις – Δραστηριότητες - Θέματα προς συζήτηση

1. Δημιουργία διαγραμμάτων δομής για να τεκμηριωθεί ο σχεδιασμός ενός συστήματος.

2. Χρησιμοποίηση του κατάλληλου γραφικού συμβολισμού για την σχεδίαση αντικειμένων συμπεριλαμβανομένων των χαρακτηριστικών και των μεθόδων τους. (Ενδεικτικά παραδείγματα: κατάλογο βιβλιοθήκης, πολυμηχάνημα, σύστημα κρατήσεων).

3. Ομαδοσυνεργατική εργασία με σκοπό την ανάπτυξη μιας πρωτότυπης διεπαφής χρήστη για ένα έργο υιοθετώντας βασικές αρχές σχεδιασμού. Η διεπαφή θα υποστηρίζει τη λειτουργικότητα ενός συστήματος χωρίς όμως να είναι κατά ανάγκη λειτουργικό. Προτείνεται ετεροαξιολόγηση μεταξύ των ομάδων.

4. Να γίνει διάγραμμα δομής μιας υποθετικής σχολικής εικονικής εταιρίας πώλησης αντικειμένων των μαθητών. Να σχεδιασθεί τμήμα παραγγελιών και τμήμα πωλήσεων αντικειμένων.

5. Χρησιμοποιώντας όλα τα αντικείμενα που σας δίνονται στο 4.3 και τους κανόνες που μάθατε στο 4.4 να δημιουργήσετε μια διεπαφή για το ηλεκτρονικό σύστημα εισαγωγής βιβλίων σε μια σχολική βιβλιοθήκη.

Βιβλιογραφία

Στα Ελληνικά

Βεσκούκης, Β. (2000). *Τεχνολογία Λογισμικού.* Πάτρα: Ι. Ε.Α.Π,

Γιακουμάκης Μ. & Διαμαντίδης Ν. (2009). *Τεχνολογία λογισμικού*. Αθήνα: Σταμούλη Α.Ε.

Στα Αγγλικά

Garlan D. & Shaw M. (1994). *An Introduction to Software Architecture*. Ανακτήθηκε από http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf

Nielsen J. (1993). *Usability Engineering*. 1st/edition, London: Academic press.

O'Brien J. A. & Marakas G. (2008). *Introduction to Information Systems*. 14th/edition, New York: McGraw-Hill.

Pfleeger L.S. (2012). *Τεχνολογία λογισμικού Θεωρία και πράξη.* Μτφρ. Φρυσήρας Κ., Επιμ Σταμέλος Γ, Αθήνα: Κλειδάριθμος.

Sommerville I. (2009). *Βασικές αρχές τεχνολογίας λογισμικού. Μτφρ*. Τσιλογιάννης Δ. Αθήνα: Κλειδάριθμος.

Σχεδιασμός και υλοποίηση διαδικτυακών εφαρμογών

Περιεχόμενα

5.1 Εισαγωγή
5.2 Σχεδιασμός και Ενσωμάτωση Δικτύου
- 5.2.1 Δίκτυα Υπολογιστών
- 5.2.2 Τοπολογίες Εταιρικών Δικτύων (Internet, Intranet, Extranet)
- 5.2.3 Ενσωμάτωση δικτύου
- 5.2.4 Περιγραφή δικτύου
- 5.2.5 Πρωτόκολλα επικοινωνίας
- 5.2.6 Χωρητικότητα δικτύου

5.3 Το περιβάλλον ανάπτυξης και η αρχιτεκτονική της εφαρμογής
- 5.3.1 Αρχιτεκτονική διαδικτυακού λογισμικού
- 5.3.2 Τεχνολογίες διαδικτυακού λογισμικού
- 5.3.3 Τεχνολογίες απεικόνισης
  - Εισαγωγή στην HTML και CSS
  - Εργαλεία HTML (notepad++)
  - Βασικές ετικέτες στην HTML
  - Λίστες και πίνακες στην HTML
  - Φόρμες στην HTML
- 5.3.4 Τεχνολογίες προγραμματισμού πελάτη (client side)
  - Εισαγωγή στην Javascript
  - Μεταβλητές και πίνακες στην Javascript
  - Δομές επιλογής-επανάληψης στην Javascript
  - Αλληλεπίδραση Javascript με στοιχεία HTML (DOM)
  - Διαχείριση γεγονότων στην Javascript
  - Παράδειγμα ελέγχου ορθής συμπλήρωσης φόρμα με Javascript

5.4 Προγραμματισμός εξυπηρετητή σε PHP
- 5.4.1 Μεταβλητές
- 5.4.2 Δομή Επιλογής

5.4.3 Δομή Επανάληψης
5.4.4 Πίνακες
5.4.5 Συναρτήσεις
5.4.6 Πρόσβαση και διαχείριση Βάσεων Δεδομένων MySQL από το διαδίκτυο χρησιμοποιώντας PHP
5.4.7 Εισαγωγή Δεδομένων φόρμας σε Βάση Δεδομένων MySQL από το διαδίκτυο χρησιμοποιώντας PHP
5.4.8 MVC Πλαίσια για την δημιουργία διαδικτυακών PHP εφαρμογών (PHP Web Applications)

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι:

- Να εφαρμόζουν την διαδικασία σχεδιασμού εφαρμογών στο διαδίκτυο σύμφωνα με τις προτεινόμενες απαιτήσεις.
- Να περιγράφουν τα χαρακτηριστικά των περιβαλλόντων ανάπτυξης καθώς και τις αντίστοιχες αρχιτεκτονικές εφαρμογών.
- Να αναγνωρίζουν την διασύνδεση των τεχνολογιών για την ανάπτυξη διαδικτυακών εφαρμογών.
- Να εφαρμόζουν τις τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών για την υλοποίηση διαδικτυακών πληροφοριακών συστημάτων.
- Να αξιολογούν την δουλειά τους (προσωπική - ομαδική) με συγκεκριμένα κριτήρια αποδοτικότητας

5.1 Εισαγωγή

Όπως έχουμε δει σε προηγούμενες ενότητες, μπορούμε να φανταστούμε τον σχεδιασμό ενός συστήματος σαν ένα σύνολο από σχέδια που χρησιμοποιούνται για την κατασκευή ενός σπιτιού. Τα σχέδια οργανώνονται με βάση τα μέρη από τα οποία αποτελείται ένα σπίτι και περιγράφουν τα δωμάτια, τους τοίχους, τα παράθυρα, τις πόρτες, τις καλωδιώσεις, τις ηλεκτρολογικές εγκαταστάσεις καθώς και όλες τις υπόλοιπες λεπτομέρειες. Παρόμοια οργάνωση ακολουθούμε και στον σχεδιασμό ενός συστήματος παρόλο που τα σημεία που περιγράφουμε διαφέρουν από αυτά ενός σπιτιού. Σχεδιάζουμε δηλαδή και καθορίζουμε τα διάφορα συστατικά του συστήματος. Το αποτέλεσμα της σχεδίασης αποτελεί ένα προσχέδιο για την μετέπειτα υλοποίηση του.

Η διαδικασία του σχεδιασμού αποτελείται από δύο επίπεδα. Ξεκινάει από ένα γενικό σχεδιασμό, τον λεγόμενο αρχιτεκτονικό σχεδιασμό (architectural design) και καταλήγει σε ένα πιο ειδικό σχεδιασμό στον λεπτομερή σχεδιασμό (detail design). Κατά την διάρκεια του αρχιτεκτονικού σχεδιασμού περιγράφουμε τα σημεία από τα οποία αποτελείται το σύστημα. Προσδιορίζουμε τη συνολική δομή και μορφή της λύσης σε ένα πιο αφηρημένο επίπεδο. Κατά τον λεπτομερή σχεδιασμό εστιάζουμε στις λεπτομέρειες, δηλαδή στο περιβάλλον ανάπτυξης και στις διάφορες τεχνολογίες υλικού και λογισμικού που θα χρησιμοποιηθούν.

Ο σχεδιασμός ενός πληροφοριακού συστήματος περιλαμβάνει μεταξύ άλλων τον σχεδιασμό και την ενσωμάτωση του δικτύου καθώς και τον σχεδιασμό της αρχιτεκτονικής της εφαρμογής και του λογισμικού. Δύο είναι τα βασικά ερωτήματα που πρέπει να απαντηθούν με την ολοκλήρωση των δύο παραπάνω σχεδιαστικών δραστηριοτήτων.

Το πρώτο ερώτημα αφορά την σχεδίαση και την ενσωμάτωση της δικτυακής δραστηριότητας: Έχουν καθοριστεί λεπτομερώς πώς τα διάφορα μέρη του συστήματος θα επικοινωνούν μεταξύ τους; Οι χρήστες ενός πληροφοριακού συστήματος εκτελούν μία σειρά από ενέργειες από διαφορετικές τοποθεσίες. Οι τοποθεσίες αυτές πρέπει να είναι συνδεδεμένες μεταξύ τους με τέτοιο τρόπο ώστε να ικανοποιούν με τον καλύτερο δυνατό τρόπο της απαιτήσεις της επιχείρησης ή του οργανισμού. Ο τομέας της επικοινωνίας της επιχείρησης ή του οργανισμού μέσω ενός δικτύου υπολογιστών αποτελεί αντικείμενο μελέτης της σχεδίασης του δικτύου. Κατά την σχεδίαση του δικτύου καθορίζονται με λεπτομέρεια η μορφή του δικτύου, οι δυνατότητες του, το υλικό που θα χρησιμοποιηθεί καθώς και σημαντικά τεχνικά ζητήματα όπως η αξιοπιστία, η ασφάλεια, η απόδοση και ο συγχρονισμός.

Το δεύτερο βασικό ερώτημα που πρέπει να απαντηθεί με την ολοκλήρωση της σχεδίασης της αρχιτεκτονικής της εφαρμογής και του λογισμικού είναι το εξής: Έχουν καθοριστεί οι χρησιμοποιούμενες τεχνολογίες λογισμικού που θα υποστηρίξουν τις δραστηριότητες του συστήματος που διενεργούνται από τους χρήστες και τους υπολογιστές; Για παράδειγμα, κατά την διαδικασία της ανάλυσης

αποφασίζεται αν οι χρήστες θα είναι σε θέση να έχουν πρόσβαση στο νέο σύστημα μόνο στους χώρους εργασίας τους ή θα έχουν επίσης την δυνατότητα να εργάζονται από το σπίτι τους μέσω μιας σύνδεσης στο Internet. Αν είναι επίσης απαραίτητο να επιτραπεί σε απομακρυσμένες ασύρματες συσκευές να συνδεθούν με το σύστημα. Ποια είναι τα διάφορα είδη δραστηριοτήτων καθώς και ο όγκος των δραστηριοτήτων αυτών που το νέο σύστημα θα πρέπει να είναι σε θέση να χειριστεί. Η υποστήριξη των παραπάνω εργασιών των χρηστών από τεχνολογικής πλευράς αποτελεί αντικείμενο μελέτης της σχεδίασης της αρχιτεκτονικής της εφαρμογής και του λογισμικού. Για παράδειγμα, κατά την σχεδίαση της αρχιτεκτονικής της εφαρμογής θα πρέπει να αποφασιστεί αν θα χρησιμοποιηθούν υπηρεσίες διαδικτύου και στην περίπτωση που η απάντηση είναι θετική ποια θα είναι η χρησιμοποιούμενη υποδομή καθώς και τα αντίστοιχα εργαλεία ανάπτυξης.

Στη συνέχεια θα αναφερθούμε διεξοδικά στα διάφορα μέρη που αποτελούν τον σχεδιασμό του δικτύου καθώς και της αρχιτεκτονικής της εφαρμογής. Η παρουσίαση των θεματικών ενοτήτων του κεφαλαίου συμπληρώνεται στο τέλος με εκτενή αναφορά στα εργαλεία ανάπτυξης διαδικτυακών εφαρμογών.

5.2 Σχεδιασμός και Ενσωμάτωση Δικτύου

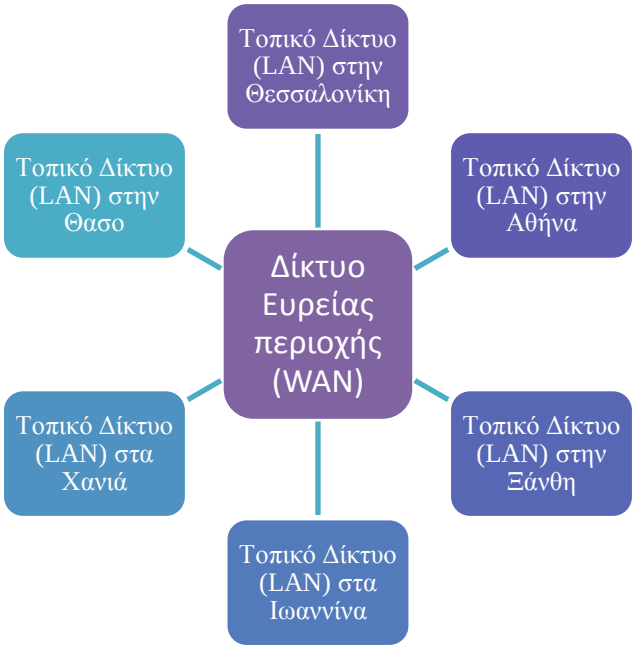
Τα **δίκτυα υπολογιστών** χρησιμοποιούνται για τις διάφορες δραστηριότητες των επιχειρήσεων και των οργανισμών. Υποστηρίζουν δυνατότητα επικοινωνίας μεταξύ των υπολογιστικών συστημάτων και των χρηστών. Η δυνατότητα αυτή μπορεί να αφορά παροχή διαφόρων υπηρεσιών (όπως για παράδειγμα τηλεφωνική υπηρεσία, video conferencing, e-mail) καθώς και κατανομή ψηφιακών πόρων (όπως για παράδειγμα πρόσβαση σε ηλεκτρονικά έγγραφα, σε εφαρμογές και βάσεις δεδομένων). Για τον λόγο αυτόν, τα σύγχρονα έργα ανάπτυξης πληροφοριακών συστημάτων συμπεριλαμβάνουν στον σχεδιασμό τους και αυτόν του δικτύου.

Ο σχεδιασμός του δικτύου είναι ένα σημαντικό ζήτημα που πρέπει να αντιμετωπιστεί νωρίς κατά τον σχεδιασμό ενός οποιουδήποτε πολυεπίπεδου (multitiered) συστήματος. Τα βασικά θέματα του σχεδιασμού του δικτύου ενός πληροφοριακού συστήματος αφορούν τα παρακάτω:

- Ενσωμάτωση των αναγκών του δικτύου του νέου συστήματος στην υπάρχουσα διαδικτυακή υποδομή
- Περιγραφή των δραστηριοτήτων επεξεργασίας και σύνδεσης στο δίκτυο από κάθε θέση του συστήματος
- Περιγραφή των πρωτόκολλων επικοινωνίας και του ενδιάμεσου λογισμικού (middleware) που συνδέει τα διάφορα στρώματα
- Διασφάλιση της επαρκής χωρητικότητας του δικτύου

5.2.1 Δίκτυα Υπολογιστών

Στο παρακάτω σχήμα 5.1 παρουσιάζεται ένα πιθανό δίκτυο υπολογιστών μιας επιχείρησης. Κάθε τοπικό δίκτυο LAN εξυπηρετεί μία γεωγραφική τοποθεσία και όλα τα τοπικά δίκτυα συνδέονται μέσω ενός δικτύου ευρείας περιοχής WAN. Οι χρήστες και οι υπολογιστές μιας συγκεκριμένης τοποθεσίας επικοινωνούν μεταξύ τους μέσω του τοπικού τους δικτύου LAN. Η επικοινωνία μεταξύ χρηστών που βρίσκονται σε διαφορετικές γεωγραφικές τοποθεσίες διεξάγεται μέσω των τοπικών τους δικτύων αντίστοιχα καθώς και του δικτύου ευρείας περιοχής που ενώνει τα τοπικά δίκτυα της εταιρίας ή του οργανισμού μεταξύ τους. Ο δρομολογητής συνδέει κάθε τοπικό δίκτυο με το δίκτυο ευρείας περιοχής. Πακέτα πληροφοριών ενός χρήστη ή ενός υπολογιστή ενός τοπικού δικτύου Α που προορίζονται για χρήστες ή υπολογιστές ενός άλλου τοπικού δικτύου Β μεταφέρονται μέσω των δρομολογητών στο δίκτυο ευρείας περιοχής και από εκεί στον αντίστοιχο χρήστη ή υπολογιστή του τοπικού δικτύου Β. Διάφορες τεχνολογίες χρησιμοποιούνται για την υλοποίηση των τοπικών και ευρείας περιοχής δικτύων, όπως τεχνολογίες Ethernet και μισθωμένες γραμμές ψηφιακής τεχνολογίας αντίστοιχα. Αυτές ποικίλουν σύμφωνα με το κόστος, την χωρητικότητα και την αξιοπιστία.



Σχήμα 5.1*: Δίκτυο υπολογιστών μιας επιχείρησης ή ενός οργανισμού*

Ένα δίκτυο υπολογιστών με τον κατάλληλο εξοπλισμό και επαρκή δυνατότητα μεταφοράς δεδομένων μπορεί να υποστηρίξει ταυτόχρονα πολλές υπηρεσίες. Υπάρχουν πολλοί τρόποι διανομής πόρων ενός πληροφοριακού συστήματος σε ένα δίκτυο υπολογιστών. Χρήστες, εφαρμογές και βάσεις δεδομένων μπορούν να είναι αξιοποιούν τις δυνατότητες ενός υπολογιστικού συστήματος ενός τοπικού δικτύου, διαφορετικών υπολογιστικών συστημάτων του ίδιου τοπικού δικτύου ή διαφορετικών υπολογιστικών συστημάτων διαφόρων τοπικών δικτύων. Τμήματα εφαρμογών και βάσεων δεδομένων κατανέμονται σε διάφορα υπολογιστικά συστήματα σε όλος το εύρος του δικτύου υπολογιστών μιας επιχείρησης ή ενός οργανισμού.

5.2.2 Τοπολογίες Εταιρικών Δικτύων (Internet, Intranet, Extranet)

Το **Διαδίκτυο (Internet)** είναι ένα παγκόσμιο σύστημα δικτύων υπολογιστών που συνδέονται μεταξύ τους χρησιμοποιώντας ένα κοινό χαμηλού επίπεδου πρότυπο δικτύωσης που ονομάζεται TCP/IP. Ο Παγκόσμιος Πληροφοριακός Ιστός (World Wide Web ή Web) είναι ένα σύνολο από πόρους (προγράμματα, αρχεία και υπηρεσίες) που μπορούν να προσπελαστούν μέσω του Διαδικτύου μέσω συγκεκριμένων πρωτοκόλλων. Το Διαδίκτυο είναι η υποδομή πάνω στην οποία βασίζεται ο Παγκόσμιος Ιστός. Με άλλα λόγια, οι πόροι του Παγκόσμιου Ιστού παρέχονται στους χρήστες μέσω του Διαδικτύου.

Ένα **εσωτερικό εταιρικό δίκτυο** (**intranet**) είναι ένα ιδιωτικό δίκτυο υπολογιστών ή ένα κλειστό εταιρικό δίκτυο που είναι προσπελάσιμο μόνο από ένα περιορισμένο σύνολο χρηστών. Οι χρήστες αυτοί είναι συνήθως εξουσιοδοτημένα μέλη της ίδιας εταιρείας ή της ίδιας ομάδας εργασίας και έχουν δικαιώματα πρόσβασης σε συγκεκριμένους μόνο πόρους. Τα μη δημοσιευμένα ονόματα πόρων, τα firewalls και οι κωδικοί πρόσβασης ατόμων ή ομάδων προστατεύουν τα intranet από μη-εξουσιοδοτημένους εξωτερικούς εισβολείς.

Το **διευρυμένο εταιρικό δίκτυο** (**extranet)** είναι ένα διευρυμένο εσωτερικό εταιρικό δίκτυο (intranet) το οποίο έχει επεκταθεί ώστε να περιλαμβάνει τους εξουσιοδοτημένους εξωτερικούς συνεργάτες μιας επιχείρησης ή ενός οργανισμού όπως για παράδειγμα είναι οι προμηθευτές, οι μεγάλοι πελάτες και οι στρατηγικοί εταίροι. Το διευρυμένο εταιρικό δίκτυο (extranet) επιτρέπει σε εξουσιοδοτημένες ομάδες συνεργαζόμενων εταιρειών να ανταλλάσσουν πληροφορίες και να συντονίζουν τις δραστηριότητές τους σχηματίζοντας κατά αυτόν τον τρόπο ένα εικονικό οργανισμό (**virtual organization)**. Μια ευρέως χρησιμοποιούμενη μέθοδος εφαρμογής ενός διευρυμένου εταιρικού δικτύου (extranet) είναι αυτή του εικονικού ιδιωτικού δικτύου (**virtual private network ή VPN**), το οποίο είναι ένα ιδιωτικό δίκτυο που είναι ασφαλές και προσβάσιμο μόνο στα μέλη του οργανισμού (ή του εικονικού οργανισμού).

5.2.3 Ενσωμάτωση δικτύου

Οι σύγχρονες επιχειρήσεις και οργανισμοί βασίζονται σε δίκτυα υπολογιστών για να υποστηρίξουν τις πολλές και διαφορετικές εφαρμογές τους. Στις περιπτώσεις που το υφιστάμενο δίκτυο μπορεί να φιλοξενήσει το νέο σύστημα, η ενσωμάτωση του νέου πληροφοριακού συστήματος πραγματοποιείται με τέτοιο τρόπο ώστε να μην επηρεάζονται οι υπάρχουσες εφαρμογές. Συνήθως απαιτούνται ελάχιστες αλλαγές όπως η προσθήκη συνδέσεων για νέους εξυπηρετητές, η τροποποίηση της δρομολόγησης και η διαμόρφωση του τοίχου προστασίας (firewall) για να μπορέσει να καταστεί δυνατή η επικοινωνία μεταξύ των νέων επιπέδων εφαρμογών. Ο προγραμματισμός επιπρόσθετων σημαντικών αλλαγών, όπως αναβάθμιση της χωρητικότητας, εισαγωγή νέων πρωτοκόλλων επικοινωνίας ή τροποποίηση πρωτοκόλλων ασφάλειας είναι πολύ πιο περίπλοκος και αποτελεί ευθύνη του

διαχειριστή του δικτύου, διότι είναι ο μόνος που γνωρίζει το υφιστάμενο δίκτυο και τον τρόπο που οι διάφορες διαδικτυακές εφαρμογές εκτελούνται και συνεργάζονται μεταξύ τους. Ο ρόλος του αναλυτή είναι να παρέχει στον διαχειριστή του δικτύου αρκετές πληροφορίες και χρόνο έτσι ώστε να καταστεί δυνατή η ανάπτυξη, η δοκιμή και η ολοκλήρωση του νέου συστήματος.

5.2.4 Περιγραφή δικτύου

Υπάρχουν αρκετοί τρόποι περιγραφής της διαδικτυακής υποδομής μιας εφαρμογής. Συνήθως ένα διάγραμμα δικτύου περιγράφει το πώς τα διάφορα επίπεδα εφαρμογών κατανέμονται στις διάφορες τοποθεσίες και υπολογιστικά συστήματα. Πιο συγκεκριμένα, αποτυπώνονται στο διάγραμμα, που «τρέχουν» τα διάφορα προγράμματα εφαρμογών, που βρίσκονται οι εξυπηρετητές και οι σταθμοί εργασίας και πως είναι οργανωμένοι οι διαδικτυακοί πόροι. Οι απαιτήσεις της εφαρμογής και η πολιτική του οργανισμού ή της εταιρείας είναι αυτές που καθορίζουν την θέση των εξυπηρετητών, τις διόδους επικοινωνίας και τα ζητήματα ασφάλειας.

5.2.5 Πρωτόκολλα επικοινωνίας

Με βάση το διάγραμμα του δικτύου προσδιορίζονται οι απαιτήσεις στα πρωτόκολλα επικοινωνίας, δηλαδή η μορφή και η σειρά των μηνυμάτων που ανταλλάσσονται μεταξύ δύο ή περισσοτέρων κόμβων που επικοινωνούν μεταξύ τους καθώς και οι διάφορες ενέργειες που διενεργούνται κατά τη μετάδοση ή τη λήψη ενός μηνύματος.

Για παράδειγμα, στην περίπτωση που έχουμε ένα ιδιωτικό δίκτυο δικτύου ευρείας περιοχής WAN ποια πρωτόκολλα επικοινωνίας πρέπει να υποστηρίζονται για να μπορούν οι διάφοροι εξουσιοδοτημένοι χρήστες να συνδεθούν και να κάνουν ερωτήματα; Και στην περίπτωση κατάρρευσης του ιδιωτικού αυτού WAN θα μπορούν τα παραπάνω πρωτόκολλα επικοινωνίας να υποστηρίξουν εναλλακτικές λύσεις, όπως αυτής της δρομολόγησης των μηνυμάτων μέσω κρυπτογραφημένων συνδέσεων; Όλοι οι πελάτες θα πρέπει να είναι σε θέση να στείλουν αιτήματα HTTP και να λαμβάνουν ενεργό περιεχόμενο ως HTML φόρμες και ενσωματωμένα σενάρια. Οι διακομιστές εφαρμογών θα πρέπει να είναι σε θέση να επικοινωνούν με υπηρεσίες πιστωτικού έλεγχου μέσω του Διαδικτύου. Οι τοίχοι προστασίας (Firewalls) και οι δρομολογητές πρέπει να ρυθμιστούν για να υποστηρίξουν όλες τις αλληλεπιδράσεις μεταξύ των σταθμών εργασίας, των υπολογιστών των πελατών, των εξυπηρετητών Web/εφαρμογών καθώς και των άλλων υπηρεσιών. Ένα τοπικό δίκτυο κέντρο δεδομένων πρέπει να υποστηρίζει τουλάχιστον ένα πρωτόκολλο για τη μετάδοση ερωτημάτων βάσεων δεδομένων και απαντήσεων μεταξύ των εξυπηρετητών κεντρικών υπολογιστών (mainframe) και Web/εφαρμογών.

5.2.6 Χωρητικότητα δικτύου

Κατά την διάρκεια της ανάλυσης καταγράφονται πληροφορίες σχετικά με τις δραστηριότητες και το είδος των δραστηριοτήτων ανά τοποθεσία. Δηλαδή, ποιες

οντότητες δεδομένων προσπελαύνονται (για παράδειγμα πελάτης, παραγγελία), στο πλαίσιο ποιών δραστηριοτήτων (για παράδειγμα δημιουργία νέας παραγγελίας), ποιες ενέργειες γίνονται πάνω σε αυτά (για παράδειγμα δημιουργία, ενημέρωση, διάβασμα, διαγραφή) και που συμβαίνουν όλα τα παραπάνω (δηλαδή σε ποιόν εξυπηρετητή συγκεκριμένα). Οι πληροφορίες αυτές μπορούν να χρησιμεύσουν για μία πρώτη εκτίμηση των απαιτήσεων για ικανότητα επικοινωνίας. Για παράδειγμα, για την δημιουργία μίας νέας παραγγελίας, ο όγκος των δεδομένων που θα μεταδοθούν είναι της τάξης των 500 bytes και ο μέσος χρόνος προσπέλασης των δεδομένων θα είναι της τάξης των 2 προσπελάσεων ανά λεπτό και σε ώρες αιχμής θα είναι 10 προσπελάσεις ανά λεπτό.

Με τον σχεδιασμό και την ολοκλήρωση των περισσότερων τμημάτων του πληροφοριακού συστήματος, όπως αυτής των βάσεων δεδομένων είναι δυνατό να έχουμε μία καλύτερη προσέγγιση των απαιτήσεων σε ικανότητα επικοινωνίας ή να μπορούμε να μετρήσουμε με περισσότερη ακρίβεια τον πραγματικό χρόνο μετάδοσης δεδομένων.

Το επόμενο βήμα στον σχεδιασμό του πληροφοριακού συστήματος αφορά την αρχιτεκτονική της εφαρμογής καθώς και το αντίστοιχο λογισμικό εφαρμογών. Οι αποφάσεις που λαμβάνονται κατά τον σχεδιασμό της αρχιτεκτονικής των εφαρμογών επηρεάζουν αλλά και επηρεάζονται με την σειρά τους από την υφιστάμενη διαδικτυακή εφαρμογή. Στην παρακάτω ενότητα θα παρουσιαστούν οι διάφορες επιλογές που είναι δυνατό να συναντήσουμε στην αρχιτεκτονική μιας εφαρμογής καθώς και στο αντίστοιχο λογισμικό.

5.3 Το περιβάλλον ανάπτυξης και η αρχιτεκτονική της εφαρμογής

Ο **παγκόσμιος ιστός** (**World Wide Web**) εξαπλώνεται διαρκώς σε ολοένα και περισσότερους τομείς της ανθρώπινης δραστηριότητας. Αποτελεί τομέα των Τεχνολογιών Πληροφορικής και Επικοινωνιών που εξελίσσεται ραγδαία. Η εξέλιξη του παγκόσμιου ιστού οφείλεται στην βελτίωση των υποδομών δικτύου, στην μείωση του κόστους πρόσβασης στο διαδίκτυο για τον μέσο χρήστη καθώς στην ευκολία πρόσβασης στις υπηρεσίες που προσφέρει από οποιοδήποτε μέρος του κόσμου. Αν και μάθαμε σε προηγούμενες τάξεις ποιες είναι αυτές οι υπηρεσίες- δυνατότητες που προσφέρει ο παγκόσμιος ιστός, δεν μπορούμε να αποκλείσουμε νέες καινοτόμες υπηρεσίες που δημιουργούνται σε καθημερινή βάση. Γι αυτό η μορφή του παγκόσμιου ιστού έχει αλλάξει ριζικά από την εποχή που δημιουργήθηκε από τον **Tim Berners-Lee το 1989**. Η μελλοντική του μορφή δεν είναι καθόλου εύκολα προβλέψιμη.

Η ταχύτατη ανάπτυξη του παγκόσμιου ιστού οφείλεται και στο γεγονός ότι πολλοί φορείς (κράτη, εταιρείες, κοινότητες προγραμματιστών, ιδιώτες) επενδύουν χρόνο και χρήμα σε αυτόν. Το γεγονός αυτό όμως είχε ως αποτέλεσμα την δημιουργία πολλών εργαλείων, προτύπων και γενικότερα τεχνολογιών για την ανάπτυξη διαδικτυακών εφαρμογών. Τάση η οποία θα συνεχιστεί και στο μέλλον. Γι αυτό το λόγο ο

μελλοντικός προγραμματιστής διαδικτυακών εφαρμογών θα πρέπει να έχει την ικανότητα γρήγορης παρακολούθησης και μάθησης νέων γνώσεων-τεχνολογιών.

Σκοπός αυτής της ενότητας είναι να κατανοήσουμε και να μάθουμε τις βασικές τεχνολογίες ανάπτυξης –υλοποίησης διαδικτυακών εφαρμογών με τέτοιο τρόπο ώστε να μπορούμε στο μέλλον να προσαρμοζόμαστε γρήγορα και εύκολα μόνοι μας στις τεχνολογικές εξελίξεις του χώρου και γιατί όχι να διαμορφώσουμε εμείς την μελλοντική εξέλιξη του παγκόσμιου ιστού.

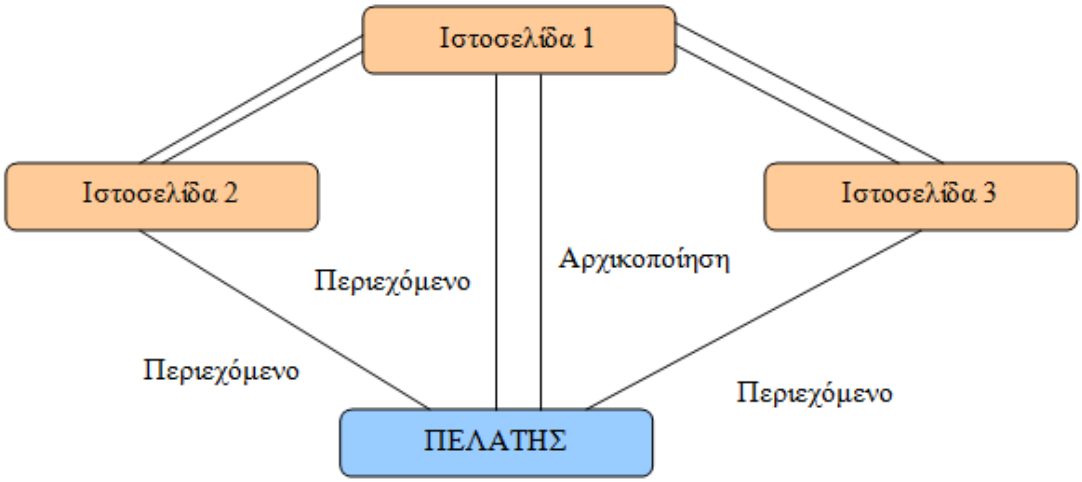
5.3.1 Αρχιτεκτονική διαδικτυακού λογισμικού

Όπως όλα τα ανθρώπινα πολύπλοκα επινοήματα (κτίρια, ηλεκτρονικές συσκευές, αυτοκίνητα κ.α.) ή καλύτερα εφευρέσεις που αποτελούνται από πολλά διαφορετικά μέρη που συνεργάζονται μεταξύ τους έτσι και το διαδικτυακό λογισμικό χαρακτηρίζεται από την αρχιτεκτονική του. Όπως ήδη έχουμε μάθει από προηγούμενα κεφάλαια με τον όρο αρχιτεκτονική αναφερόμαστε στα συστατικά που αποτελούν το διαδικτυακό λογισμικό καθώς και στην δομή με την οποία αλληλεπιδρούν. Πριν δοθεί η γενική αρχιτεκτονική διαδικτυακού λογισμικού κρίνεται σκόπιμο να καταλάβουμε τι είναι διαδικτυακό λογισμικό – εφαρμογή και πως αυτό διαφέρει από μια απλή ιστοσελίδα.

Ορισμός: Διαδικτυακή εφαρμογή-λογισμικό είναι μια εφαρμογή πελάτη/εξυπηρετητή (client/server) που χρησιμοποιεί ένα φυλλομετρητή (web browser) για το πρόγραμμα πελάτη και «καταναλώνει» μια αλληλεπιδραστική υπηρεσία με την σύνδεση του σε εξυπηρετητές μέσω του διαδικτύου. Μια διαδικτυακή εφαρμογή προβάλλει κατάλληλο δυναμικό περιεχόμενο βασισμένο στις απαιτήσεις του χρήστη, στις καταγεγραμμένες συνήθειες του και σε θέματα ασφάλειας.

Από τον παραπάνω ορισμό είναι ξεκάθαρο πως μια διαδικτυακή εφαρμογή δεν έχει τον στατικό χαρακτήρα προβολής περιεχομένου της απλής ιστοσελίδας αλλά δίνει την δυνατότητα αλληλεπίδρασης και παραγωγής περιεχομένου στον χρήστη. Η αρχιτεκτονική δε που ακολουθεί μια διαδικτυακή εφαρμογή είναι αυτή του πελάτη/εξυπηρετητή. Στο σχήμα που ακολουθεί προβάλλεται η γενική αρχιτεκτονική που ακολουθούν οι διαδικτυακές εφαρμογές.

Σύμφωνα με το παρακάτω σχήμα 5.2 ο φυλλομετρητής του χρήστη επικοινωνεί με το εξυπηρετητή ιστοσελίδας για την προβολή και δημιουργία περιεχομένου. Επίσης ο εξυπηρετητής επικοινωνεί με εξυπηρετητές άλλων σελίδων για την παροχή κατάλληλου περιεχομένου στον χρήστη (πελάτη). Ο πελάτης (χρήστης) μπορεί να αντλεί περιεχόμενο και από άλλους εξυπηρετητές χωρίς να το αιτηθεί σε αυτούς.



Σχήμα 5.2: *Γενική Αρχιτεκτονική Διαδικτυακών εφαρμογών*

5.3.2 Τεχνολογίες διαδικτυακού λογισμικού

Σύμφωνα με την αρχιτεκτονική που δόθηκε στην προηγούμενη υποενότητα οι αντίστοιχες τεχνολογίες υλοποίησης διαδικτυακών εφαρμογών πρέπει να υποστηρίζουν τα παρακάτω δομικά στοιχεία της αρχιτεκτονικής:

- Πελάτης (web browser)
- Εξυπηρετητής (web server)
- Επικοινωνία εξυπηρετητών

Για τον πελάτη οι τεχνολογίες υλοποίησης διακρίνονται σε:

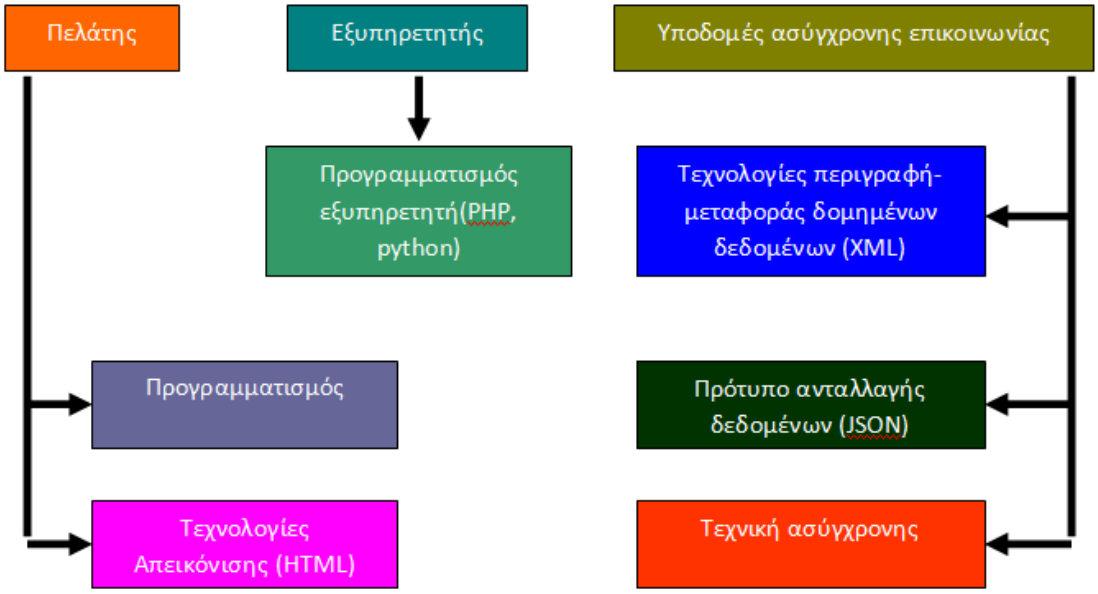
- Τεχνολογίες απεικόνισης (HTML, CSS). Τεχνολογίες με τις οποίες το περιεχόμενο προβάλλεται ομοιόμορφα σε διάφορα είδη φυλλομετρητών αλλά και υπολογιστών.
- Προγραμματισμός πελάτη (javascript). Τεχνολογία με την οποία γράφεται κώδικας που εκτελείται από τον φυλλομετρητή του χρήστη. Χρησιμοποιείται για κινούμενα γραφικά, για έλεγχο ορθής εισαγωγής στοιχείων κ.α.

Για τον εξυπηρετητή χρησιμοποιούνται τεχνολογίες προγραμματισμού του (PHP, python, ASP, NODE.js, JSP κ.α.). Ο κώδικας αυτών τεχνολογιών εκτελείται στον εξυπηρετητή και τα αποτελέσματά του μόνο προβάλλονται στον πελάτη. Οι τεχνολογίες αυτές χρησιμοποιούνται για την δημιουργία και προβολή δυναμικού περιεχομένου.

Οι υποδομές ασύγχρονης επικοινωνίας μεταξύ εξυπηρετητών ή πελατών διακρίνονται σε:

- Τεχνολογίες περιγραφής και μεταφοράς δομημένων δεδομένων (XML)
- Πρότυπο ανταλλαγής δεδομένων (JSON)
- Τεχνική ασύγχρονης επικοινωνίας (AJAX)

Όλες οι παραπάνω τεχνολογίες συνοψίζονται στο παρακάτω σχήμα 5.3.



***Σχήμα 5.3**: Τεχνολογίες διαδικτυακού λογισμικού*

5.3.3. Τεχνολογίες απεικόνισης

Οι τεχνολογίες απεικόνισης αποτελούν το μέσο με το οποίο η πληροφορία που μεταδίδεται στο διαδίκτυο απεικονίζεται με ενιαίο τρόπο σε οποιοδήποτε υπολογιστικό σύστημα (προσωπικός υπολογιστής PC, tablets, έξυπνα κινητά smartphones) που διαθέτει λογισμικό φυλλομετρητή (browser).

Στην ενότητα αυτή θα μάθουμε να αξιοποιούμε την βασική τεχνολογία απεικόνισης HTML (HyperText Markup Language) για την ανάπτυξη διαδικτυακών εφαρμογών. Επίσης θα μάθουμε να χρησιμοποιούμε την τεχνολογία μορφοποίησης περιεχομένου CSS (Cascade Style Sheet).

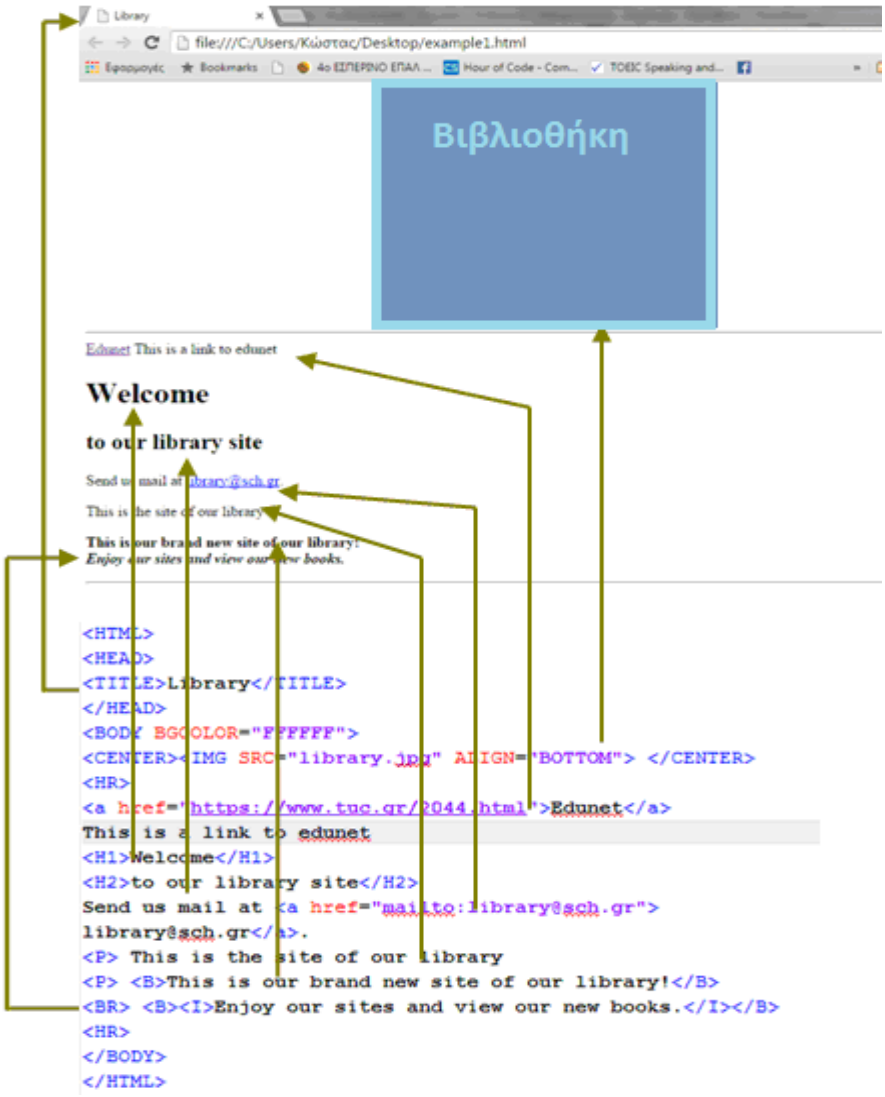
Εισαγωγή στην HTML και CSS

Στην δεκαετία του 1980, αν και είχαν ήδη δημιουργηθεί οι υποδομές δικτύωσης, προέκυψε το πρόβλημα της συμβατότητας αναπαράστασης της πληροφορίας. Ενώ οι επιστήμονες της εποχής διέθεταν υπολογιστικά συστήματα δικτυωμένα μεταξύ τους η ανταλλαγή πληροφοριών ήταν μια δύσκολη διαδικασία καθώς διέθεταν διαφορετικούς υπολογιστές με διαφορετικά λειτουργικά συστήματα και με διαφορετικά λογισμικά επεξεργασίας και προβολής κειμένου. Η ανάγκη ανταλλαγής

επιστημονικών δεδομένων οδήγησε τον Tim Berners-Lee επιστήμονα του CERN στη ανακάλυψη της γλώσσας απεικόνισης HTML. Σε αυτό το σημείο πρέπει να επισημανθεί ότι η ανάγκη ή καλύτερα η ανακάλυψη ενός κενού (στην καθημερινή ζωή, στο εμπόριο, στις επιστήμες κ.α.) αποτελεί ένα από τα βασικά κίνητρα για την ανακάλυψη και προώθηση της καινοτομίας.

Συζήτηση στην τάξη: Με βάση την καθημερινότητα σας, τις γνώσεις σας και τα ενδιαφέροντα σας σκεφτείτε και συζητήστε με τους συμμαθητές σας ιδέες που θα μπορούσαν να αποτελέσουν τεχνολογικές ή/και επιχειρηματικές καινοτομίες. Αφού ακούσετε όλες τις ιδέες των συμμαθητών σας διαλέξτε, χωρίς προσωπικούς ενδοιασμούς τις καλύτερες. Υιοθετήστε τις ιδέες που επιλέχθηκαν και σε ομάδες προτείνετε τρόπους βελτίωσής τους. Είστε έτοιμοι να καινοτομήσετε;

Για την αντιμετώπιση του προβλήματος της συμβατότητας αναπαράστασης της πληροφορίας ο Tim Berners-Lee πρότεινε την δημιουργία μιας γλώσσας περιγραφής της, την HTML. Οποιοδήποτε υπολογιστικό σύστημα μπορεί να αναπαραστήσει αξιόπιστα και ομοιόμορφα την πληροφορία αρκεί να διαθέτει κατάλληλο λογισμικό που διαβάζει και κατανοεί τις "οδηγίες" της HTML και ανάλογα με αυτές εμφανίζει την πληροφορία στην οθόνη. Το λογισμικό αυτό είναι ο γνωστός σε όλους μας ως φυλλομετρητής (browser). Η HTML δεν αποτελεί μια γλώσσα προγραμματισμού όπου οι εντολές που γράφουμε εκτελούνται από τον επεξεργαστή αλλά είναι μια γλώσσα περιγραφής ιστοσελίδας και οι "οδηγίες" της ή όπως θα δούμε στην συνέχεια οι ετικέτες, απευθύνονται στον φυλλομετρητή για να εμφανίσει κατάλληλα το περιεχόμενο. Όλα αυτά απεικονίζονται στο παρακάτω σχήμα 5.4.



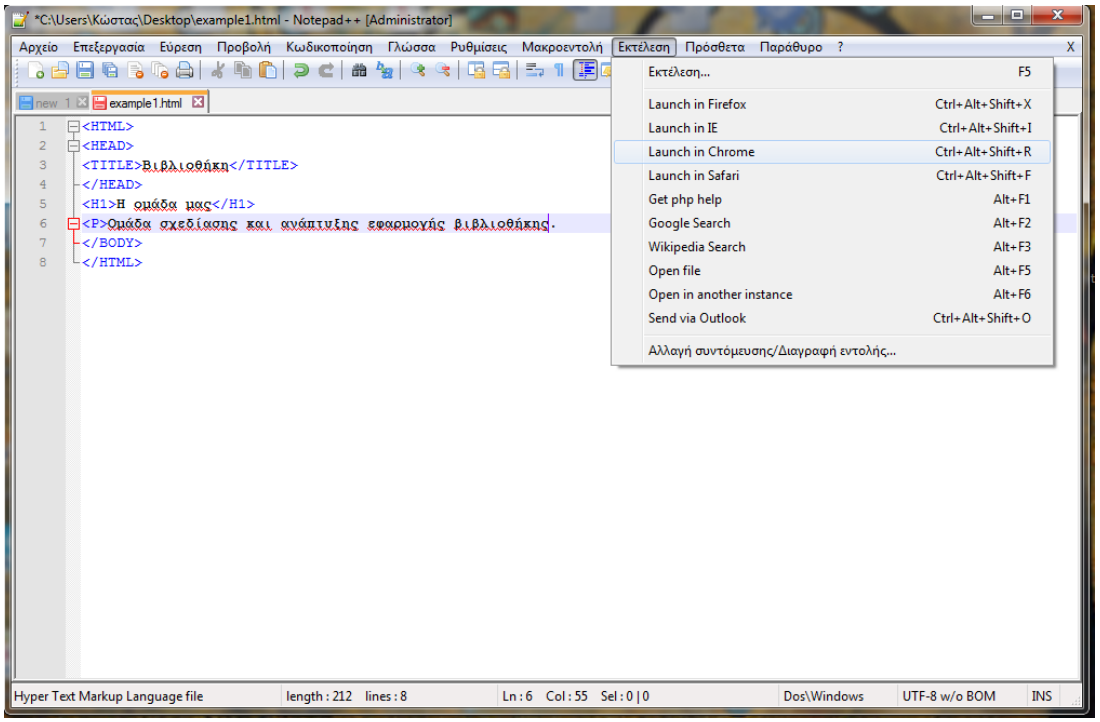
***Σχήμα 5.4**: Οδηγίες-ετικέτες HTML στον φυλλομετρητή*

Ενώ, όπως θα δούμε παρακάτω, είναι δυνατή η μορφοποίηση περιεχομένου με την HTML, προέκυψε η ανάγκη να διαχωριστεί το περιεχόμενο από την μορφοποίηση του καθώς οι ιστοσελίδες μεγάλωναν σε όγκο και η συντήρηση-αναβάθμιση τους γινόταν ολοένα και δυσκολότερη. Για αυτό τον λόγο δημιουργήθηκε η τεχνολογία CSS με την οποία μπορεί να γίνει χωριστά η μορφοποίηση των ετικετών της HTML.

Εργαλεία HTML (notepad++)

Ένα από τα πλεονεκτήματα της HTML είναι ότι χρειάζεται έναν απλό κειμενογράφο (notepad) για την συγγραφή κώδικα και έναν φυλλομετρητή για τον έλεγχο. Βέβαια έχουν δημιουργηθεί πολλά ολοκληρωμένα περιβάλλοντα ανάπτυξης ιστοσελίδων για επαγγελματική χρήση. Στη συνέχεια θα χρησιμοποιήσουμε το λογισμικό ανοικτού κώδικα notepad++ για την συγγραφή κώδικα σε HTML, CSS, Javascript και PHP. Το λογισμικό αυτό προτείνεται διότι είναι "ελαφρύ" στην εκτέλεση του, διευκολύνει την συγγραφή κώδικα με την αυτόματη χρήση χρωμάτων στις δεσμευμένες λέξεις-εντολές της κάθε τεχνολογίας, έχει την δυνατότητα διαχείρισης πολλών εγγράφων και

επιτρέπει την άμεση εκτέλεση του φυλλομετρητή για έλεγχο. Το notepad++ είναι διαθέσιμο για αποθήκευση και εγκατάσταση στην ηλεκτρονική διεύθυνση https://notepad-plus-plus.org/. Το περιβάλλον του notepad++ απεικονίζεται στο παρακάτω σχήμα 5.5.



***Σχήμα 5.5**: Το λογισμικό notepad++*

Όπως φαίνεται στο σχήμα 5.5 το notepad++ διαθέτει το κλασσικό μενού επιλογών (αρχείο, επεξεργασία, εύρεση, προβολή). Προσοχή πρέπει να δοθεί κατά την αποθήκευση ενός αρχείου όσον αφορά την επέκταση του. Όλα τα αρχεία HTML αποθηκεύονται με την επέκταση .html για να είναι αναγνώσιμα από τον φυλλομετρητή. Ο κώδικας εισάγεται στην επιφάνεια της κάθε ανοικτής καρτέλας. Χρήσιμη λειτουργία αποτελεί η "εκτέλεση" με την οποία μπορούμε να δούμε την ιστοσελίδα που μόλις δημιουργήσαμε στον επιθυμητό μας φυλλομετρητή.

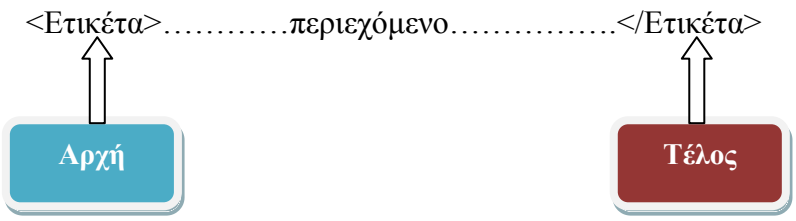
Δραστηριότητα: Ανοίξτε το notepad++ και δημιουργήστε ένα νέο αρχείο με επέκταση .html (myfisrtpage.html). Στην συνέχεια εισάγετε τον κώδικα που βλέπετε στο παραπάνω σχήμα 5.5. Εκτελέστε τον κώδικα στον διαθέσιμο φυλλομετρητή. Επισκεφτείτε την ηλεκτρονική διεύθυνση

http://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic

εισάγετε τον ίδιο κώδικα στο κατάλληλο πλαίσιο και δείτε το αποτέλεσμα στο διπλανό πλαίσιο. Αναζητήστε στο διαδίκτυο και άλλους online HTML editors. Συγκρίνετε τους offline με τους online HTML editors. Είναι δικαιολογημένη η τάση ανάπτυξης ακόμα και των κλασσικών απλών εφαρμογών στο διαδίκτυο;

Βασικές ετικέτες στην HTML

Οι περισσότερες ετικέτες στην HTML ακολουθούν την παρακάτω γενική δομή



***Σχήμα 5.6**: Γενική δομή ετικέτας HTML*

Η αρχή κάθε ετικέτας επισημαίνεται με το όνομα της ετικέτας ανάμεσα στα δύο <> σύμβολα ενώ το τέλος επισημαίνεται με το σύμβολο / και το όνομα της ετικέτας ανάμεσα στα δύο σύμβολα <> όπως φαίνεται στο παραπάνω σχήμα 5.6. Ανάμεσα στη αρχή και το τέλος παρεμβάλλεται το περιεχόμενο που απεικονίζεται με αυτήν την ετικέτα. Το τέλος της ετικέτας δεν είναι απαραίτητο να βρίσκεται στην ίδια γραμμή με την αρχή και μπορεί να γραφεί σε επόμενη γραμμή. Επιπροσθέτως, μια ετικέτα μπορεί να εμπεριέχει ως περιεχόμενο και άλλες ετικέτες.

Η βασική δομή που ακολουθεί οποιοδήποτε αρχείο HTML φαίνεται στο παρακάτω σχήμα 5.7.

```
<HTML>
  <HEAD>
    <TITLE>  ΤΙΤΛΟΣ ΣΕΛΙΔΑΣ </TITLE>
  </HEAD>
  <BODY>
        ΚΥΡΙΟ ΠΕΡΙΕΧΟΜΕΝΟ
  </BODY>
</HTML>
```

***Σχήμα 5.7**: Βασική δομή αρχείου HTML*

Η ετικέτα <HTML> προσδιορίζει την αρχή και το τέλος ενός αρχείου HTML. Ένα HTML αρχείο αποτελείται από δύο μέρη. Την επικεφαλίδα που προσδιορίζεται από την ετικέτα <HEAD> και το κυρίως σώμα που προσδιορίζεται από την ετικέτα <BODY>. Στην επικεφαλίδα μπορεί να μπει η ετικέτα <TITLE>, ο τίτλος της σελίδας όπως διαφαίνεται στην καρτέλα της ιστοσελίδας στον φυλλομετρητή, ετικέτες σχετικές με μορφοποίηση καθώς και ετικέτες-λέξεις κλειδιά που θα δούμε αναλυτικά σε επόμενη υποενότητα. Στο κύριο μέρος μπαίνει όλο το περιεχόμενο της ιστοσελίδας με οποιαδήποτε μορφή.

Για την παρουσίαση περιεχομένου σε παράγραφο χρησιμοποιείται η ετικέτα <p>. Το τέλος σε αυτήν την ετικέτα δεν είναι απαραίτητο. Για την δημιουργία κειμένου κεφαλίδας χρησιμοποιούνται οι ετικέτες <H1>, <H2>, <H3>, <H4>, <H5> και <H6>. Για την εισαγωγή εικόνας χρησιμοποιείται η ετικέτα <img>, όπως φαίνεται στην παρακάτω δήλωση.

<img src="mypicture.jpg" style="width:300px;height:220px;">

Η παράμετρος src της ετικέτας <img> αποτελεί το όνομα του αρχείου της εικόνας που θα προβληθεί. Το αρχείο εικόνας πρέπει να βρίσκεται στο ίδιο φάκελο με το αρχείο HTML διαφορετικά στο src πρέπει να γραφεί όλη η διαδρομή θέσης του αρχείου εικόνας. Οι τύποι αρχείων εικόνας που υποστηρίζονται από την HTML είναι οι: JPEG, GIF και PNG. Στην παράμετρο style ορίζεται ένα παράθυρο σε pixels με τις ιδιότητες width (πλάτος) και height (ύψος) μέσα στο οποίο θα προσαρμοστεί η εικόνα.

Για την δημιουργία συνδέσμων (links - λέξεις ή εικόνες με τις οποίες ο χρήστης μπορεί να μεταβεί σε άλλο κόμβο πληροφορίας) χρησιμοποιείται η ετικέτα <a> , όπως φαίνεται στην παρακάτω δήλωση.

Σχολικό Δίκτυο

Σύμφωνα με το παραπάνω παράδειγμα οι λέξεις «Σχολικό δίκτυο» αποτελούν σύνδεσμο. Κάθε φορά που ο χρήστης κάνει click σε αυτές τις λέξεις ο φυλλομετρητής φορτώνει το περιεχόμενο της σελίδας που βρίσκεται στην παράμετρο href της ετικέτας <a>. Στο συγκεκριμένο παράδειγμα θα φορτώσει την σελίδα **www.sch.gr.** Η λειτουργία όλων αυτών των ετικετών φαίνονται στην πράξη στο παρακάτω σχήμα 5.8.

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>
   <img src="library.jpg" style="width:300px;height:220px;">
   <H1>Η βιβλιοθήκη μας</H1>
   <H2>Η βιβλιοθήκη μας</H2>
   <H3>Η βιβλιοθήκη μας</H3>
   <H4>Η βιβλιοθήκη μας</H4>
   <H5>Η βιβλιοθήκη μας</H5>
   <H6>Η βιβλιοθήκη μας</H6>
   <P>Καλώς ήρθατε στην ιστοσελίδα του σχολείου μας
   <P>Εδώ μπορείτε να ενημερωθείτε για τα βιβλία που μπορείτε να δανειστείτε
  </BODY>

</HTML>
```

***Σχήμα 5.8**: Παράδειγμα βασικών HTML ετικετών*

Δραστηριότητα: Εισάγετε τον κώδικα του παραπάνω παραδείγματος στο notepad++ και κάντε τους απαραίτητους ελέγχους. Εμφανίζεται η εικόνα; Διορθώστε το

πρόβλημα. Δημιουργήστε μια νέα σελίδα όπου ως τίτλο θα έχει το μικρό σας όνομα, θα εμφανίζεται η φωτογραφία σας (selfie) καθώς και η φράση «Μερικά λόγια για μένα» με κεφαλίδα της αρεσκείας σας και τέλος θα περιέχει μια σύντομη περιγραφή του εαυτού σας.

Λίστες και πίνακες στην HTML

Στην προηγούμενη ενότητα μάθαμε τις βασικές ετικέτες της HTML με τις οποίες μπορούμε να αποδώσουμε πληροφορία με την μορφή κειμένου και/ή εικόνας. Πολλές φορές όμως δεν αρκεί μόνο η παρουσίαση της πληροφορίας αλλά πρέπει να αποδίδεται το πώς αυτή είναι δομημένη. Στην υποενότητα αυτή θα μάθουμε τις βασικές ετικέτες δόμησης πληροφορίας, τις λίστες και τους πίνακες.

Οι λίστες που είναι διαθέσιμες στην HTML είναι οι **αριθμημένες (ordered)** και οι **μη αριθμημένες (unordered)**. Οι αριθμημένες λίστες εμφανίζουν δεδομένα με μια αύξουσα σειρά αρίθμησης ενώ οι μη αριθμημένες λίστες δεν εμφανίζουν σειρά αρίθμησης. Η ετικέτα για την δημιουργία αριθμημένης λίστας είναι η <OL> και η ετικέτα που χρησιμοποιείται για την δημιουργία αντικειμένου μέσα στην λίστα είναι η <LI>. Για τη δημιουργία μη αριθμημένης λίστας χρησιμοποιείται η ετικέτα <UL> ενώ για την δημιουργία αντικειμένων μέσα στην λίστα χρησιμοποιείται πάλι η ετικέτα <LI>. Όλα αυτά φαίνονται στο παρακάτω σχήμα 5.9.

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>

   <H1>Κατηγορίες Βιβλίων</H1>
   <UL>
        <LI>Πληροφορικής</LI>
        <LI>Μαθηματικά</LI>
        <LI>Ιστορία</LI>
        <LI>Μυθιστορήματα</LI>
    </UL>

   <H1>Μαθητές που έχουν διαβάσει τα περισσότερα βιβλία</H1>
   <OL>
        <LI>Κώστας Χ.</LI>
        <LI>Μαρία Κ.</LI>
        <LI>Ιουλιάνα Α.</LI>
        <LI>Κατερίνα Ε.</LI>
    </OL>


  </BODY>

</HTML>
```

***Σχήμα 5.9**: Λίστες στην HTML*

Δραστηριότητα: Εισάγετε τον κώδικα του παραπάνω παραδείγματος στο notepad++ και κάντε τους απαραίτητους ελέγχους. Ποια είναι η διαφορά των δύο ειδών λιστών;.

Η πιο κλασσική δομή δεδομένων είναι αυτή του πίνακα. Σε έναν πίνακα τα δεδομένα οργανώνονται σε γραμμές και στήλες όπως φαίνεται στο παρακάτω σχήμα 5.10.

Α.Μ.	Ονοματεπώνυμο	Ιδιότητα	Τηλέφωνο	e-mail
1123	Κώστας Παυλάκης	Καθ	6977777777	kpaul@vsch.gr
2321	Γιάννης Παπαδόπουλος	Μαθ	6988888888	gpap@vsch.gr
4742	Χρυσούλα Σταυρακίδου	Καθ	6955555555	chstaur@vsch.gr
6734	Νικολέτα Καραβαγγέλη	Μαθ	6944444444	nkarabag@vsch.gr

***Σχήμα 5.10**: Δομή δεδομένων – Πίνακας*

Η ετικέτα <TABLE> χρησιμοποιείται για την δημιουργία πίνακα στην HTML. Η ετικέτα <TR> χρησιμοποιείται για την δημιουργία γραμμών μέσα στον πίνακα ενώ η ετικέτα <TD> χρησιμοποιείται για την δημιουργία κελιών μέσα στην γραμμή (αν ο αριθμός κελιών είναι ο ίδιος σε κάθε γραμμή τότε αυτός ο αριθμός είναι ο αριθμός των στηλών). Η χρήση αυτών των ετικετών φαίνεται στο παρακάτω σχήμα 5.11.

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>

   <H1>Κατάλογος Βιβλίων</H1>
   <TABLE border="1px">
       <TR>
         <TD>ISBN</TD>
         <TD>TITLE</TD>
         <TD>YEAR PUBLISHED</TD>
       </TR>
       <TR>
         <TD>0778983932</TD>
         <TD>Introduction to JAVA</TD>
         <TD>2009</TD>
       </TR>
       <TR>
         <TD>0028892822</TD>
         <TD>Arduino projects</TD>
         <TD>2011</TD>
       </TR>
   </TABLE>
  </BODY>
</HTML>
```

***Σχήμα 5.11**: Πίνακας στην HTML*

Δραστηριότητα: Εισάγετε τον κώδικα του παραπάνω παραδείγματος στο notepad++ και κάντε τους απαραίτητους ελέγχους. Αναζητήστε στο διαδίκτυο το ρόλο της ιδιότητας border του πίνακα. Προσθέστε μια νέα στήλη με τίτλο θεματολογία. Για κάθε βιβλίο θα υπάρχει συνοπτική θεματολογία με την μορφή μη αριθμημένης λίστας.

Φόρμες στην HTML

Στις προηγούμενες ενότητες μάθαμε ετικέτες με τις οποίες απεικονίζουμε πληροφορία. Παρόλα αυτά τα σύγχρονα online πληροφοριακά συστήματα δίνουν την δυνατότητα αλληλεπίδρασης στον χρήστη, δηλαδή την δυνατότητα εισαγωγής δεδομένων από τον χρήστη στο πληροφοριακό σύστημα. Η ετικέτα <FORM> της HTML επιτρέπει την εισαγωγή γραφικών στοιχείων με τα οποία ο χρήστης μπορεί να εισάγει πληροφορία. Σε αυτό το σημείο πρέπει να διευκρινιστεί ότι η HTML δεν αρκεί για την αποστολή και αποθήκευση δεδομένων χρήστη αλλά χρειάζεται και προγραμματισμός στον εξυπηρετητή (PHP), γνώσεις που θα περιγραφούν σε επόμενη ενότητα. Η χρήση της ετικέτας <FORM> περιγράφεται στο σχήμα 5.12.

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>

   <H1>Εγγραφή νέου χρήστη</H1>

    <form action="action_page.php">
        First name:<br>
        <input type="text" name="firstname" value="">
        <br>
        Last name:<br>
        <input type="text" name="lastname" value="">
        <br>
        e-mail:<br>
        <input type="text" name="email" value="">
        <br>
        phone:<br>
        <input type="text" name="phone" value="">
        <br>
        password:<br>
        <input type="text" name="password" value="">
        <br>
        retype password:<br>
        <input type="text" name="repassword" value="">
        <br><br>
        <input type="submit" value="Submit">
    </form>
  </BODY>
</HTML>
```

***Σχήμα 5.12**: Η ετικέτα <FORM>*

Βασική ιδιότητα της ετικέτας <FORM> είναι η action, με την οποία δηλώνεται το αρχείο PHP του οποίου ο κώδικας θα εκτελεστεί στον εξυπηρετητή όταν πατηθεί το κουμπί της φόρμας. Η ετικέτα <input> χρησιμοποιείται για την εισαγωγή γραφικών στοιχείων στη φόρμα. Η ιδιότητα type υποδηλώνει τον τύπο του γραφικού στοιχείου, δηλαδή αν είναι πλαίσιο κειμένου (text), κουμπί ενέργειας (submit) και άλλα. Η ιδιότητα name αποτελεί το αναγνωριστικό όνομα του γραφικού στοιχείου για την περεταίρω επεξεργασία των δεδομένων του από την PHP ενώ η ιδιότητα value περιέχει τα δεδομένα του γραφικού στοιχείου.

Δραστηριότητα: Εισάγετε τον κώδικα του παραπάνω παραδείγματος στο notepad++ και κάντε τους απαραίτητους ελέγχους. Αναζητήστε στο διαδίκτυο το γραφικό στοιχείο radiobutton και χρησιμοποιώντας το δώστε την επιλογή στο χρήστη να επιλέγει το φύλο.

5.3.4 Τεχνολογίες προγραμματισμού πελάτη (client side)

Στην προηγούμενη ενότητα μάθαμε για την HTML, τεχνολογία με την οποία μπορούμε να απεικονίσουμε την πληροφορία στατικά. Δεν μπορούμε όμως να απεικονίσουμε την δυναμική της πληροφορίας. Για αυτό το λόγο χρησιμοποιούνται οι τεχνολογίες προγραμματισμού πελάτη (client side) με τις οποίες μπορούμε να εισάγουμε κώδικα που εκτελείται δυναμικά και αλληλεπιδρά με τα στοιχεία της HTML. Ο κώδικας αυτός εκτελείται στον φυλλομετρητή του χρήστη (client). Διάφορες τεχνολογίες έχουν δημιουργηθεί για αυτό το σκοπό όπως είναι η vbscript και η Javascript . Στις επόμενες ενότητες θα μάθουμε τις βασικές έννοιες της Javascript έτσι ώστε να μπορούμε να την χρησιμοποιούμε αλλά και να αποκτήσουμε τις βάσεις για περεταίρω εκμάθηση αυτής της τεχνολογίας.

Εισαγωγή στην Javascript

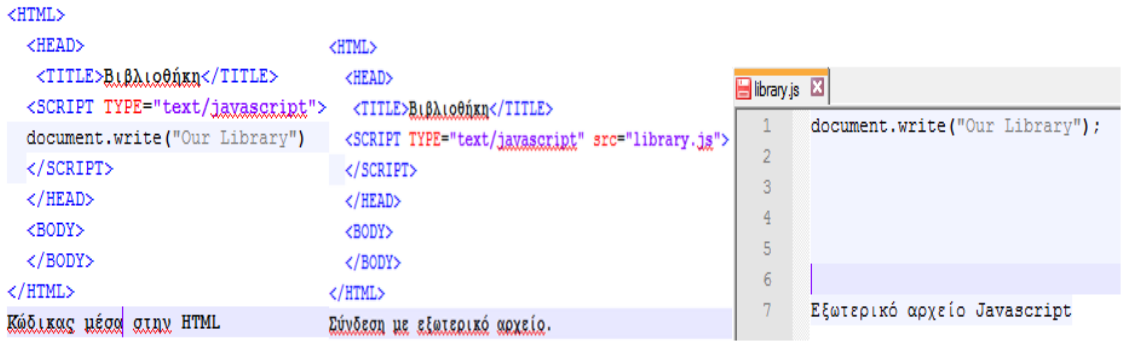
Η Javascript αποτελεί μια από τις πιο διαδεδομένες γλώσσες συγγραφής σεναρίων (scripting language) για προγραμματισμό στην πλευρά του πελάτη (client side). Έχει πολλές δυνατότητες και χρησιμοποιείται για τον εμπλουτισμό ενός online πληροφοριακού συστήματος με αυξημένη λειτουργικότητα και αλληλεπίδραση. Το σημαντικότερο πλεονέκτημά της είναι ότι εκτελείται στο πελάτη (στον φυλλομετρητή) καθιστώντας την εφαρμογή ολοένα και πιο γρήγορη. Επίσης η Javascript χρησιμοποιείται από πολλούς προγραμματιστές για αυτό και υπάρχουν άφθονοι πόροι στο διαδίκτυο τόσο για την εκμάθησή της όσο και για την γρήγορη επίλυση προβλημάτων (έτοιμες και πολλές βιβλιοθήκες είναι ελεύθερες προς χρήση).

Η Javascript ενσωματώνεται σε ένα αρχείο HTML με την ετικέτα <SCRIPT> της HTML. Ενδιάμεσα σε αυτή την ετικέτα μπορεί να γραφεί κώδικας σε Javascript. Επίσης κώδικας σε αυτή τη γλώσσα μπορεί να γραφεί σε εξωτερικό αρχείο με κατάληξη .js (π.χ. library.js). Η σύνδεση με το αρχείο αυτό γίνεται μέσω της ιδιότητας src της ετικέτας <SCRIPT>. Η ετικέτα <SCRIPT> μπορεί να ενσωματωθεί

σε οποιοδήποτε μέρος μιας σελίδας HTML (BODY ή HEAD). Η πρώτη εντολή της Javascript (εμφάνιση μηνύματος "our library" στην οθόνη) που θα μάθουμε είναι η:

document.write("Our Library");

Όλα τα παραπάνω συνοψίζονται στο σχήμα 5.13.



***Σχήμα 5.13**: Εισαγωγή στην Javascript*

Μπορούμε να χρησιμοποιήσουμε το Notepad++ για την δημιουργία εξωτερικών αρχείων τύπου .js.

Δραστηριότητα: Εισάγετε τον κώδικα των παραπάνω δύο παραδειγμάτων στο notepad++ και κάντε τους απαραίτητους ελέγχους.

Μεταβλητές και πίνακες στην Javascript

Όπως όλες οι γλώσσες προγραμματισμού έτσι και η Javascript υποστηρίζει τον μηχανισμό προσωρινής αποθήκευσης δεδομένων στην μνήμη RAM, δηλαδή τον μηχανισμό των μεταβλητών. Οι τύποι δεδομένων που υποστηρίζονται από αυτήν είναι:

- Αριθμοί (ακέραιοι – δεκαδικοί)
- Αλφαριθμητικά (strings)
- Λογικές (true–false)
- null (κενή μεταβλητή)
- undefined (μη προσδιορισμένη)

Οι μεταβλητές στην Javascript δημιουργούνται με την δεσμευμένη λέξη var, το όνομα της μεταβλητής, το σύμβολο εκχώρησης = και την τιμή που θέλουμε να εκχωρήσουμε. Ο τύπος δεδομένων της μεταβλητής ορίζεται αυτόματα από τον τύπο δεδομένων της εκχωρούμενης τιμής. Όλα αυτά συνοψίζονται στα παρακάτω παραδείγματα δημιουργίας μεταβλητών.

- var number = 8;
- var pi = 3.14;

- var book_title = “Internet programming ”;
- var book_year = “2009”;
- var booked = FALSE;
- var temp = null;
- var feature = undefined;

Οι βασικοί τελεστές που χρησιμοποιούνται για τη επεξεργασία μεταβλητών είναι οι: +, -. *, / και η χρήση τους φαίνεται στα παρακάτω παραδείγματα (τα σύμβολα // προσθέτουν σχόλια στον κώδικα).

- number = number + 8; // αυξάνεται η τιμή της number κατά 8
- number += 2; // αυξάνεται η τιμή της number κατά 2
- book_title += book_year //Η μεταβλητή book_title θα αποκτήσει την τιμή “Internet programming 2009”
- number = number - 2 // Η τιμή του number μειώνεται κατά 2
- number = number * 3 // Η τιμή του number πολλαπλασιάζεται με το 3
- number = number / 4 // Η τιμή του number διαιρείται με το 4

Η Javascript επιτρέπει την δημιουργία πινάκων, δηλαδή την προσωρινή αποθήκευση στην μνήμη RAM πολλαπλών δεδομένων με το ίδιο όνομα. Μπορούμε να ξεχωρίσουμε τα δεδομένα με την χρήση δείκτη. Στην Javascript δεν είναι απαραίτητο όλα τα στοιχεία ενός πίνακα να είναι του ίδιου τύπου. Συγκεκριμένα ένας πίνακας στην Javascript δημιουργείται με τους τρεις εναλλακτικούς τρόπους όπως φαίνεται παρακάτω:

Α) var Books = new Array();

Β) var Books = new Array(“Learning JAVA”, “Data structures in C++”, “Arduino projects”);

Γ) var Books = [“Learning JAVA”, “Data structures in C++”, “Arduino projects”];

Με τον Α) τρόπο δημιουργείται ο κενός πίνακας Books. Οι τρόποι Β) και Γ) είναι ισοδύναμοι και δημιουργούν τον πίνακα Books τριών θέσεων όπου σε κάθε θέση έχει τον τίτλο του κατά σειρά βιβλίου. Η αρίθμηση των θέσεων του πίνακα ξεκινάει από το 0. Έτσι το πρώτο στοιχείο είναι το Books[0], το δεύτερο το Books[1] και το τρίτο το Books[2]. Χρησιμοποιώντας την εντολή:

document.write(Books[1]);

θα τυπωθούν στην ιστοσελίδα τα περιεχόμενα της δεύτερης θέσης του πίνακα Books, δηλαδή το αλφαριθμητικό “Data structures in C++”. Ενώ με την εντολή εκχώρησης:

Books[2] = “Raspberry PI in action”;

τα περιεχόμενα της τρίτης θέσης του πίνακα Books θα αλλάξουν σε "Raspberry PI in action". Μια βασική ιδιότητα των πινάκων είναι η length (πλήθος στοιχείων πίνακα). Έτσι το Books.length μας δίνει 3 όσο και το πλήθος των στοιχείων. Αν θέλουμε να διαγράψουμε τα δυο τελευταία στοιχεία του πίνακα θα γράφαμε:

Books.length = 1;

Η Javascript διαθέτει αρκετές μεθόδους για την επεξεργασία πινάκων (λειτουργίες πάνω στους πίνακες). Για παράδειγμα η μέθοδος reverse() αντιστρέφει την σειρά των στοιχείων. Έτσι η εντολή:

Books.reverse();

Βάζει στο πρώτο στοιχείο την τιμή "Arduino projects" ενώ στο τρίτο βάζει την τιμή "Learning JAVA". Μια άλλη μέθοδος είναι η push() που εισάγει στο τέλος του πίνακα ένα νέο στοιχείο.

Books.push("Internet of Things");

Συζήτηση στην τάξη: Ποια θα είναι η μορφή του πίνακα Books μετά την εκτέλεση της παραπάνω εντολής; Υπάρχουν άλλες μέθοδοι – ιδιότητες για έναν πίνακα; Αναζητήστε στο διαδίκτυο.

Δομές επιλογής-επανάληψης στην Javascript.

Όπως σε όλες τις γλώσσες προγραμματισμού έτσι και στην Javascript υπάρχουν οι δομές επιλογής και επανάληψης. Πριν προχωρήσουμε στην περιγραφή αυτών των δομών θα περιγραφούν οι τελεστές σύγκρισης και οι λογικοί τελεστές της γλώσσας. Οι τελεστές σύγκρισης μαζί με την λειτουργία τους δίνονται παρακάτω:

- < , μικρότερο
- > , μεγαλύτερο
- <= , μικρότερο ή ίσο
- >= , μεγαλύτερο ή ίσο
- = = , ίσο (προσοχή να μην συγχέεται με τον τελεστή εκχώρησης =)
- **!**= , όχι ίσο
- = = = , αυστηρά ίσο (ίσο σε τιμή και σε τύπο δεδομένων)
- **!**= = , αυστηρά μη ίσο (μη ίσο σε τιμή και σε τύπο δεδομένων)

Ακολουθούν οι λογικοί τελεστές σύγκρισης με τους οποίους μπορεί να δημιουργηθούν σύνθετες λογικές συνθήκες:

- **&&** , λογικός τελεστής ΚΑΙ
- | | , λογικός τελεστής Η'
- **!** , λογικός τελεστής ΟΧΙ (αντιστρέφει μια λογική πρόταση)

Στην συνέχεια θα περιγραφεί η βασική δομή επιλογής **if,** σε δύο διαφορετικές εκδοχές με την μορφή παραδειγμάτων. Εξυπακούεται πως μπορούν χρησιμοποιηθούν και πιο σύνθετες συνθήκες.

```
var limit = 10;
var num_of_books = 5;
if (num_of_books < limit){
document.write("Μπορείτε να δανειστείτε")
}
```

***Σχήμα 5.14**: Η δομή επιλογής if*

Σε αυτή την εκδοχή η δομή if ελέγχει εάν ισχύει η συνθήκη μέσα στις παρενθέσεις (). Σε περίπτωση που ισχύει εκτελείται ο κώδικας ανάμεσα στις αγκύλες {}. Στην αντίθετη περίπτωση δεν εκτελείται ο κώδικας ανάμεσα στις {} και η ροή του προγράμματος συνεχίζει μετά την δομή if. Στο παράδειγμα αυτό ισχύει η συνθήκη. Κατά συνέπεια θα εκτελεστεί ο «ελεγχόμενος» κώδικας και θα τυπωθεί στην ιστοσελίδα το μήνυμα "Μπορείτε να δανειστείτε".

```
var limit = 10;
var num_of_books = 10;
if (num_of_books < limit){
document.write("Μπορείτε να δανειστείτε");
}else {
  document.write("Δεν μπορείτε να δανειστείτε");
  }
```

***Σχήμα 5.15**: Η δομή επιλογής if-else*

Σε αυτή την εκδοχή η δομή if ελέγχει εάν ισχύει η συνθήκη μέσα στις παρενθέσεις. Σε περίπτωση που ισχύει εκτελείται μόνο ο κώδικας ανάμεσα στις πρώτες αγκύλες {}. Στην αντίθετη περίπτωση εκτελείται μόνο ο κώδικας ανάμεσα στις δεύτερες αγκύλες else {}. Η ροή του προγράμματος συνεχίζει μετά την δομή if. Στο παράδειγμα αυτό δεν ισχύει η συνθήκη οπότε θα εκτελεστεί ο «ελεγχόμενος» κώδικας του else και θα τυπωθεί στην ιστοσελίδα το μήνυμα "Δεν μπορείτε να δανειστείτε".

Η Javascript διαθέτει δύο δομές επανάληψης την **while** και την **for**. Η **while** χρησιμοποιείται όταν, κατά την σχεδίαση-συγγραφή του προγράμματος, δεν είναι γνωστός ο αριθμός των επαναλήψεων ενώ η **for** χρησιμοποιείται όταν αυτός ο αριθμός είναι γνωστός. Ακολουθούν οι επεξηγήσεις των δομών αυτών μέσω παραδειγμάτων. Εξυπακούεται πως μπορούν χρησιμοποιηθούν και πιο σύνθετες συνθήκες στην περίπτωση της **while.**

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  <SCRIPT TYPE="text/javascript">
    var Books = new Array("Absolute JAVA", "Learning PHP", "Hardware design");
    var i = 0;

    while (i < Books.length){
    document.write(Books[i]);
    document.write("<br />");
    i++;
    }
  </SCRIPT>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```

***Σχήμα 5.16**:Η δομή επανάληψης while*

Στην δομή επανάληψης while ο κώδικας που βρίσκεται ανάμεσα στις αγκύλες {} επαναλαμβάνεται εφόσον ισχύει η συνθήκη στις παρενθέσεις (). **Προσοχή!** Στον επαναλαμβανόμενο κώδικα πρέπει να υπάρχουν εντολές που κάποια στιγμή θα αλλάζουν τις τιμές μεταβλητής/των έτσι ώστε κάποια στιγμή να μην ισχύει η συνθήκη και η ροή του προγράμματος να μεταφερθεί κάτω από την δομή while. Σε αυτό το παράδειγμα ελέγχεται αν η μεταβλητή i είναι μικρότερη από το μήκος του πίνακα, εάν είναι, τυπώνεται το στοιχείο του πίνακα Books που βρίσκεται στην θέση i. Σε κάθε επανάληψη η i αυξάνεται κατά 1(i++;). Με αυτόν τον τρόπο τυπώνονται όλα τα στοιχεία του πίνακα Books (εφόσον η αρχική τιμή του i είναι 0, αρχικοποίηση πριν την δομή while).

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  <SCRIPT TYPE="text/javascript">
    var Books = new Array("Absolute JAVA", "Learning PHP", "Hardware design");

    for (var i = 0; i < Books.length; i++){
    document.write(Books[i]);
    document.write("<br />");
    }
  </SCRIPT>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```

***Σχήμα 5.17**: Η δομή επανάληψης for*

Στην δομή επανάληψης for χρησιμοποιείται μια μεταβλητή που έχει τον ρόλο του μετρητή των επαναλήψεων. Το τμήμα ελέγχου της δομής επανάληψης, που βρίσκεται ανάμεσα στις παρενθέσεις, αποτελείται από τρία τμήματα α) Αρχικοποίηση της μεταβλητής-μετρητή (δίνεται η αρχική τιμή της μεταβλητής) var i = 0; β) Συνθήκη

ελέγχου μεταβλητής-μετρητή (λογική συνθήκη, i < Books.length, που περιέχει οπωσδήποτε την μεταβλητή-μετρητής, όσο ισχύει η συνθήκη επαναλαμβάνεται η εκτέλεση των εντολών που βρίσκονται ανάμεσα στις αγκύλες {}) γ) ρυθμός μεταβολής μεταβλητής-μετρητή σε κάθε επανάληψη i++ (αύξηση κατά 1, στο τμήμα αυτό δηλώνεται το πόσο θα αυξάνεται η μεταβλητή-μετρητής σε κάθε επανάληψη). Το παράδειγμα στο σχήμα 5.17 (for) έχει ακριβώς το ίδιο αποτέλεσμα με το παράδειγμα στο σχήμα 5.16 (while).

Δραστηριότητα: Εισάγετε τον κώδικα των παραπάνω παραδειγμάτων στο notepad++ και κάντε τους απαραίτητους ελέγχους. Δημιουργήστε πρόγραμμα σε Javascript που θα εκχωρεί σε έναν πίνακα 10 τίτλους βιβλίων ενώ σε ένα άλλο πίνακα θα αποθηκεύει το αν το κάθε βιβλίο είναι κρατημένο ή όχι (true/false). Κατόπιν το πρόγραμμα θα τυπώνει τα βιβλία που είναι ελεύθερα και το σύνολο αυτών.

Αλληλεπίδραση Javascript με στοιχεία HTML (DOM)

Η Javascript αλληλεπιδρά με ετικέτες της HTML μέσω του μοντέλου DOM (Document Object Model). Το DOM αποτελεί μια ιδεατή δομή της ιστοσελίδας έτσι ώστε η Javascript να έχει πρόσβαση στα στοιχεία της. Η βασική εντολή της DOM (με τις παραλλαγές της) είναι η:

document.getElementById(“someId”).value;

document.getElementById(“someId”).name;

document.getElementById(“someId”).innerHTML;

Η εντολή document.getElementById(“someId”) επιστρέφει το στοιχείο της HTML που έχει την ιδιότητα id ίση με “someId”. Γι αυτό το λόγο στα στοιχεία της HTML που θέλουμε να αλληλεπιδρούν με την Javascript ορίζουμε την ιδιότητα id. Προσθέτοντας την ιδιότητα .value η εντολή επιστρέφει την τιμή της ιδιότητας value του στοιχείου HTML. Ομοίως συμβαίνει προσθέτοντας την ιδιότητα .name. Ενώ προσθέτοντας την ιδιότητα innerHTML η εντολή επιστρέφει το περιεχόμενο του στοιχείου. Αυτά απεικονίζονται στο σχήμα 5.18.

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>

  </HEAD>
  <BODY>
  <H1 id="et1"> Our library site</h1>
  <H2 id="et2">Find amazing books</H2>
  <p id="par1"> find the books you want</p>

  <SCRIPT TYPE="text/javascript">
    var s1 = document.getElementById("et1").innerHTML;
    document.write(s1);
    document.write("<br />");
    var s2 = document.getElementById("et2").innerHTML;
    document.write(s2);
    document.write("<br />");
    var s3 = document.getElementById("par1").innerHTML;
    document.write(s3);
    document.write("<br />");
    document.getElementById("par1").innerHTML = "Ran out of books"
  </SCRIPT>

  </BODY>
</HTML>
```

***Σχήμα 5.18**: Η εντολή document.getElementById*

Στο παράδειγμα του σχήματος 5.18 η Javascript, μέσω της παραπάνω εντολής του DOM, διαβάζει τα περιεχόμενα των ετικετών <H1>, <H2> και <p> και τα ξανατυπώνει στην ιστοσελίδα. Η τελευταία εντολή του script κάνει ακριβώς το αντίθετο. Εκχωρεί μέσω του DOM μια νέα τιμή "Ran out of books" και το περιεχόμενο της ετικέτας <p> αλλάζει δυναμικά.

Μια άλλη εντολή του DOM που χρησιμοποιείται για διασύνδεση στοιχείων φόρμας <FORM> είναι η:

document.form[0].element[0].value/name/innerHTML;

Το form[0] αναφέρεται στην πρώτη φόρμα της ιστοσελίδας, το form[1] στην δεύτερη, το form[2] στην τρίτη και γενικότερα το form[n] στην n+1 φόρμα για την περίπτωση που έχουμε παραπάνω από μια φόρμες σε μια ιστοσελίδα. Συνήθως έχουμε μια φόρμα οπότε το form[0] αρκεί. Το element[0] αναφέρεται στο πρώτο στοιχείο της φόρμας (π.χ. εάν αυτό είναι πλαίσιο κειμένου, τότε θα αναφέρεται σε αυτό), το element[1] στο δεύτερο στοιχείο της φόρμας, το element[2] στο τρίτο στοιχείο της φόρμας και γενικότερα το element[n] στο n+1 στοιχείο της φόρμας. Οι ιδιότητες value/name/innerHTML έχουν τις ίδιες λειτουργίες όπως περιγράφηκαν στο παράδειγμα της getElementById.

Δραστηριότητα: Εισάγετε τον κώδικα των παραπάνω παραδειγμάτων στο notepad++ και κάντε τους απαραίτητους ελέγχους. Δημιουργήστε ένα νέο αρχείο HTML και εισάγετε μια φόρμα εισαγωγής στοιχείων βιβλίων. Χρησιμοποιώντας το DOM τυπώστε το περιεχόμενο των στοιχείων της φόρμας και αλλάξτε το με κώδικα Javascript.

Διαχείριση γεγονότων στην Javascript

Στις προηγούμενες ενότητες μάθαμε να δημιουργούμε προγράμματα σε Javascript και να τα εκτελούμε φορτώνοντας την ιστοσελίδα στον φυλλομετρητή. Η εκτέλεση αυτή γινόταν με την φόρτωση της ιστοσελίδας. Πολλές φορές όμως είναι απαραίτητο ενέργειες-κώδικας να εκτελούνται όταν υπάρχει διάδραση με τον χρήστη. Με άλλα λόγια είναι απαραίτητο να εκτελείται κώδικας σε Javascript όταν συμβαίνουν κάποια γεγονότα στην ιστοσελίδα. Τέτοια γεγονότα μπορεί να είναι το πάτημα ενός κουμπιού, το πάτημα σε ένα πλαίσιο κειμένου , η απομάκρυνση από ένα πλαίσιο κειμένου κ.α. Μερικά από τα ποιο συνηθισμένα γεγονότα είναι:

- onclick, συμβαίνει όταν πατηθεί κουμπί
- onfocus, συμβαίνει όταν πατηθεί πλαίσιο κειμένου
- onblur, συμβαίνει όταν γίνει απομάκρυνση από ένα στοιχείο ελέγχου φόρμας
- onkeyup, συμβαίνει όταν αφήνουμε ένα πλήκτρο

Πριν προχωρήσουμε στην επεξήγηση το πώς η Javascript διαχειρίζεται αυτά τα γεγονότα, θα περιγράψουμε την λειτουργία της συνάρτησης στο Σχήμα 5.19. Η συνάρτηση αποτελεί μηχανισμό δημιουργίας υποπρογράμματος, έννοια που έχουμε συναντήσει και σε άλλα μαθήματα.

```
function welcome(){
window.alert("Welcome to our library");
}

welcome();
```

***Σχήμα 5.19**: Η συνάρτηση welcome()*

Για την δημιουργία μιας συνάρτησης στην Javascript γράφουμε την λέξη function. Ακολουθεί το όνομα που δίνουμε στην συνάρτηση και οι παρενθέσεις (). Εφόσον χρειάζονται παράμετροι, τις εισάγουμε μέσα στις παρενθέσεις(). Ακολουθεί το σώμα της συνάρτησης που βρίσκεται ανάμεσα στις αγκύλες{}. Για τυχόν επιστρεφόμενη τιμή από την συνάρτηση χρησιμοποιείται η λέξη return και η τιμή που θέλουμε η συνάρτηση να επιστρέψει. Για την κλήση της συνάρτησης γράφουμε το όνομα της συνάρτησης, τις παρενθέσεις () και τυχόν τιμές παραμέτρων μέσα σε αυτές.

Στο παράδειγμα του σχήματος 5.19 έχουμε την συνάρτηση welcome() που δεν έχει παραμέτρους και δεν επιστρέφει τιμή. Η μοναδική εντολή που υπάρχει στο σώμα της συνάρτησης είναι η:

window.alert("Welcome to our library");

Με αυτήν την εντολή δημιουργείτε ένα παράθυρο ειδοποίησης που εμφανίζει το μήνυμα "Welcome to our library". Η κλήση της συνάρτησης γίνεται με την εντολή **welcome();**

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>
  <H1 id="et1"> Our library site</h1>

  <FORM>
  ENTER YOUR NAME: <INPUT type="text" onfocus="welcome();" value="">
  <INPUT type="button" onclick="welcome();" value="OK">
  </FORM>

  <SCRIPT TYPE="text/javascript">
  function welcome(){
  window.alert("Welcome to our library");
  }
  </SCRIPT>

  </BODY>
</HTML>
```

***Σχήμα 5.20**: Κλήση συνάρτησης από γεγονότα*

Στο σχήμα 5.20 περιγράφεται μια ιστοσελίδα που περιέχει μια φόρμα. Η φόρμα αποτελείται από ένα πλαίσιο εισαγωγής κειμένου και ένα κουμπί. Στο πλαίσιο εισαγωγής κειμένου είναι εμφανής η ιδιότητας διαχείρισης γεγονότος onfocus η οποία παίρνει τιμή ίδια με το όνομα της συνάρτησης welcome(). Αυτό σημαίνει ότι όταν συμβεί το γεγονός πατήματος μέσα στο πλαίσιο κειμένου θα εκτελεστεί ο κώδικας της συνάρτησης welcome(). Ομοίως στο κουμπί έχουμε την ιδιότητα διαχείρισης γεγονότος onclick η οποία παίρνει τιμή ίδια με το όνομα της συνάρτησης welcome(). Αυτό σημαίνει ότι όταν συμβεί το γεγονός πατήματος του κουμπιού θα εκτελεστεί ο κώδικας της συνάρτησης welcome(). Στο παράδειγμα αυτό περιγράφηκε ο ποιο απλός τρόπος διαχείρισης γεγονότος ενσωματώνοντας τα γεγονότα ως ιδιότητες στις ετικέτες της HTML.

Δραστηριότητα: Εισάγετε τον κώδικα των παραπάνω παραδειγμάτων στο notepad++ και κάντε τους απαραίτητους ελέγχους.

Παράδειγμα ελέγχου ορθής συμπλήρωσης φόρμα με Javascript

Σε αυτήν την ενότητα θα δούμε πως μπορούμε να χρησιμοποιήσουμε όλα αυτά που μάθαμε στην Javascript για να ελέγξουμε αν ο χρήστης έχει συμπληρώσει σωστά μια φόρμα εισαγωγής πληροφοριών βιβλίου στο σύστημα. Γίνεται η παραδοχή ότι ο έλεγχος αφορά την μη εισαγωγή κενής πληροφορίας σε κάθε πεδίο. Στο πρώτο βήμα σχεδιάζεται η φόρμα εισαγωγής στοιχείων στην HTML όπως φαίνεται στο σχήμα 5.21. Κάθε στοιχείο της φόρμας (πλαίσια κειμένου) έχει εισαχθεί σε ένα πίνακα. Επίσης έχει προστεθεί και μια τρίτη στήλη στον πίνακα (κενή προς το παρόν) για την προβολή μηνυμάτων-οδηγιών στον χρήστη ανά πεδίο εισαγωγής. Αξιοσημείωτο είναι ότι σε όλα τα πλαίσια κειμένου αλλά και στα κενά πεδία της τρίτης στήλης του πίνακα έχει προστεθεί η ιδιότητα id (για να μπορούν να είναι εύκολα και διαχειρίσιμα από το DOM). Όσον αφορά τα πλαίσια κειμένου έχει προστεθεί και η ιδιότητα-γεγονός onkeyup (όταν αφήνουμε ένα πλήκτρο) στην οποία εκχωρείται η συνάρτηση checking();

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>
  <H1 id="et1"> Our library site</h1>

  <FORM>
    <TABLE border="1">
     <tr><td>Book Title</td><td><INPUT type="text" id="book_t" onkeyup="checking();" value=""></td><td id="comments_t"></td></tr>
     <tr><td>Book ISBN</td><td><INPUT type="text" id="book_i" onkeyup="checking();" value=""></td><td id="comments_i"></td></tr>
     <tr><td>Book Author</td><td><INPUT type="text" id="book_a" onkeyup="checking();" value=""></td><td id="comments_a"></td></tr>
    </TABLE>
     <INPUT type="button" id="button1" value="OK">
  </FORM>
  </BODY>
</HTML>
```

***Σχήμα 5.21**: Φόρμα εισαγωγής βιβλίων*

Το επόμενο βήμα είναι να γραφεί κώδικας σε Javascript (για το γεγονός όταν αφήνουμε πλήκτρο) που θα ελέγχει κάθε πλαίσιο κειμένου. Αν το πλαίσιο κειμένου είναι κενό θα εμφανίζεται στο διπλανό κελί του πινάκα μήνυμα, που θα προτρέπει το χρήστη να εισάγει κατάλληλη τιμή. Διαφορετικά θα εμφανίζεται το μήνυμα "OK". Ο κώδικας αυτός εμφανίζεται στο σχήμα 5.22.

```
<SCRIPT TYPE="text/javascript">
function checking(){
var elements = new Array();
var displays = new Array();
var messages = new Array();

elements[0] = document.getElementById("book_t").value;
elements[1] = document.getElementById("book_i").value;
elements[2] = document.getElementById("book_a").value;

displays[0] = "comments_t";
displays[1] = "comments_i";
displays[2] = "comments_a";

messages[0] = "Please insert book title";
messages[1] = "Please insert book ISBN";
messages[2] = "Please insert book Author";

  for (var i = 0; i < elements.length; i++){
    if (elements[i]==""){
       document.getElementById(displays[i]).innerHTML = messages[i];
      }else{
       document.getElementById(displays[i]).innerHTML = "OK";
       }
  }

}
</SCRIPT>
```

***Σχήμα 5.22**: Κώδικας ελέγχου ορθότητας δεδομένων φόρμας*

Με το DOM η ιδιότητα value του κάθε πλαισίου κειμένου εισάγεται σε ένα πίνακα (elements). Στον πίνακα displays εισάγονται τα id των κελιών του πίνακα όπου θα εμφανίζονται τα μηνύματα ενώ στον πίνακα messages αποθηκεύονται τα μηνύματα. Κατόπιν για κάθε πλαίσιο κειμένου (for) γίνεται έλεγχος αν ο χρήστης δεν έχει εισάγει δεδομένα (elements[i] == “”). Αν δεν έχει εισάγει, μέσω DOM

documenet.getElementById(display[i]).innerHTML=messages[i];

εκχωρείται στο κατάλληλο κελί το κατάλληλο μήνυμα. Αν έχει εισάγει, με παρόμοιο τρόπο εκχωρείται η τιμή “OK”.

Δραστηριότητα: Εισάγετε τον κώδικα των παραπάνω παραδειγμάτων στο notepad++ και κάντε τους απαραίτητους ελέγχους. Αναζητήστε στο διαδίκτυο συναρτήσεις διαχείρισης αλφαριθμητικών έτσι ώστε να δημιουργήσετε μια νέα φόρμα εισαγωγής στοιχείων μαθητή πάνω στην οποία θα γίνεται έλεγχος ορθότητας δεδομένων δίνοντας έμφαση στην ορθότητα του εισαγόμενου e-mail.

5.4 Προγραμματισμός εξυπηρετητή σε PHP

Η **PHP** είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που μπορεί να κάνει όλα τα είδη των πραγμάτων, όπως να συλλέγει και να διαχειρίζεται τα δεδομένα μιας φόρμας που αποστέλλεται από ένα πρόγραμμα περιήγησης, να δημιουργεί προσαρμοσμένο διαδικτυακό (web) περιεχόμενο για να εξυπηρετήσει το πρόγραμμα περιήγησης, να επικοινωνεί με μια βάση δεδομένων, να περιορίζει τους χρήστες ώστε αυτοί να έχουν πρόσβαση σε συγκεκριμένες σελίδες στην ιστοσελίδα σας καθώς και να στέλνει και να λαμβάνει cookies (δηλαδή μικρά πακέτα δεδομένων που χρησιμοποιεί το πρόγραμμα περιήγησής σας για να θυμάται ορισμένα πράγματα, όπως για παράδειγμα αν είστε συνδεδεμένοι στο Πανελλήνιο Σχολικό Δίκτυο).

Η PHP είναι γλώσσα συγγραφής σεναρίων (scripting language) για δημιουργία δυναμικού διαδικτυακού περιεχομένου. Τα σενάρια (scripts) PHP εκτελούνται στον διακομιστή (server) και όχι στον πελάτη (client). Μέσα σε μία HTML σελίδα μπορείτε να ενσωματώσετε PHP κώδικα που θα εκτελείται κάθε φορά που θα επισκέπτεστε την συγκεκριμένη σελίδα. Γενικά, σε κώδικα HTML, CSS και JavaScript παρεμβάλλεται κώδικας PHP. Τα αρχεία PHP έχουν κατάληξη .php και αποθηκεύονται και εκτελούνται στον διακομιστή web. Ο PHP κώδικας μεταφράζεται στον διαδικτυακό διακομιστή (web server) και παράγεται περιεχόμενο σε μορφή κώδικα HTML ή άλλο περιεχόμενο που ο χρήστης μπορεί να δει.

Ρίξτε μια ματιά στον παρακάτω κώδικα. Σας θυμίζει κάτι; Πολλοί θα αναγνωρίσετε στον κώδικα αυτόν την γλώσσα HTML. Και πού ακριβώς βρίσκεται ο κώδικας PHP; Ο κώδικας PHP γράφεται μεταξύ των διαχωριστικών-οριοθετών (tags) **<?php** και **?>**. Μπορείτε να μαντέψετε τι ακριβώς κάνει ο κώδικας αυτός; Μήπως εμφανίζει ένα μήνυμα στην οθόνη; Και πιο είναι αυτό; Μαντέψατε σωστά. Το μήνυμα είναι το εξής «Θέλω να μάθω να προγραμματίζω σε PHP». Ποια συνάρτηση χρησιμοποιεί για να εμφανίσει το μήνυμα αυτό; Το βρήκατε. Είναι η συνάρτηση echo.

```
<html>
   <head>
  </head>
  <body>
      <p>
        <?php
          echo "Θέλω να μάθω να προγραμματίζω σε PHP";
        ?>
      </p>
  </body>
</html>
```

Θα μπορούσατε να ρωτήσετε το εξής: Το ίδιο πράγμα θα μπορούσα να το κάνω και σε JavaScript. Αυτό ισχύει. Η διαφορά έγκειται στο γεγονός ότι με την JavaScript μπορώ να κάνω λιγότερα πράγματα από ότι με την PHP. Ας εξηγήσουμε το γιατί.

Η JavaScript τρέχει γενικά στο πρόγραμμα περιήγησης ή αλλιώς στον πελάτη (client). Αυτό σημαίνει ότι γνωρίζει πραγματικά μόνο δύο πράγματα. Αυτά που συμβαίνουν στο πρόγραμμα περιήγησής σας καθώς και αυτά που του στέλνει η ιστοσελίδα με την οποία συνδέεστε. Η PHP από την άλλη πλευρά τρέχει στον ίδιο υπολογιστή (ή αλλιώς στον διακομιστή (server)) με τον ιστότοπο που επισκέπτεστε. Αυτό σημαίνει ότι έχει πρόσβαση σε όλες τις πληροφορίες και τα αρχεία που υπάρχουν σε αυτόν τον υπολογιστή. Η παραπάνω πρόσβαση δίνει στην PHP την δυνατότητα να δημιουργήσει προσαρμοσμένες σελίδες HTML για να τις στείλει στη συνέχεια στο πρόγραμμα περιήγησής σας, να χειριστεί τα cookies καθώς και να εκτελέσει διάφορες εργασίες χρησιμοποιώντας δεδομένα που βρίσκονται στον διακομιστή.

Δραστηριότητα: Τροποποιήστε τον παραπάνω κώδικα για να εμφανίσετε το όνομά σας.

Δραστηριότητα: Μαντέψτε τι θα εμφανιστεί με το παρακάτω τμήμα κώδικα:

```
<?php
        echo 5 * 7;
?>
```

Για να αρχίσετε να αναπτύσσετε διαδικτυακές εφαρμογές PHP που συνομιλούν με βάσεις δεδομένων χρειάζεστε έναν διακομιστή web server (για παράδειγμα XAMPP) ακόμα και στον δικό σας υπολογιστή, όπου μπορείτε να εγκαταστήσετε την PHP, την MySQL καθώς και έναν απλό επεξεργαστή κειμένου (για παράδειγμα notepad++). Ενδεικτικό παράδειγμα τοπικής εγκατάστασης xampp υπάρχει στο παράρτημα των σημειώσεων. Εναλλακτική λύση παρέχουν και οι on line πλατφόρμες ελέγχου κώδικα PHP σε περίπτωση που δεν θέλουμε να εγκαταστήσουμε λογισμικό.

5.4.1 Μεταβλητές

Μέχρι τώρα έχουμε εμφανίσει λέξεις και αποτελέσματα πράξεων. Για να δημιουργήσουμε πιο σύνθετα προγράμματα, χρειαζόμαστε έναν τρόπο για να «σώσουμε» αυτές τις τιμές. Μπορούμε να το κάνουμε αυτό χρησιμοποιώντας μεταβλητές. Μια μεταβλητή μπορεί να αποθηκεύσει μία λέξη ή έναν αριθμό. Για παράδειγμα, με την παρακάτω δήλωση

$myAge = 32

αποθηκεύουμε στην μεταβλητή με το όνομα $myAge τον αριθμό 32. Όλα τα ονόματα των μεταβλητών αρχίζουν υποχρεωτικά με το σύμβολο $. Μετά το σύμβολο $ μπορεί να ακολουθούν λατινικοί χαρακτήρες (A - z), ψηφία (0 - 9) ή ο χαρακτήρας «_» με οποία σειρά θέλουμε. Τα ονόματα των μεταβλητών είναι case-sensitive, δηλ. το **$myAge** και το **$myage** είναι δύο διαφορετικές μεταβλητές.

Δραστηριότητα: Στον παρακάτω κώδικα δημιουργήστε δύο μεταβλητές με το όνομα $myName και $myAge και εκχωρήστε σε αυτές το όνομά σας και την ηλικία σας

αντίστοιχα. Εμφανίστε την τιμή των παραπάνω μεταβλητών χρησιμοποιώντας την εντολή echo. Τοποθετήστε τις εντολές εκχώρησης τιμής μεταξύ των tags <?php και ?>

```
<html>
   <head>
   </head>
      <body>
            <p>
                  <?php

                  ?>
            </p>
      </body>
</html>
```

Οι τύποι δεδομένων που υποστηρίζονται από την PHP είναι οι εξής: String, Integer, Float (λέγεται και double), Boolean, Array, Object, NULL και Resource.

Το String είναι μία ακολουθία από χαρακτήρες, όπως "Hello world!" ανάμεσα όμως σε απλά ή διπλά εισαγωγικά.

Δραστηριότητα: Εκτελέσετε τον παρακάτω κώδικα για να μάθετε τι αυτοκίνητο οδηγεί ο James Bond. Μπορείτε να αλλάξετε το αυτοκίνητο ή/και τον οδηγό; Το σύμβολο // το χρησιμοποιούμε για να γράφουμε σχόλια. Τι καινούριο μάθατε από την εκτέλεση αυτού του κώδικα;

```
<?php
    $identity = 'James Bond';
    $car = 'BMW';
    // Η μεταβλητή $sentence μας λέει ποιος οδηγεί και τι αυτοκίνητο
    $sentence = "$identity drives a $car";
    echo $sentence;
?>
```

Δραστηριότητα: Πόσοι τρόποι υπάρχουν να πούμε το ίδιο πράγμα; Εκτελέσετε τον παρακάτω κώδικα για να μάθετε τι εννοούμε. Τι καινούριο μάθατε από την εκτέλεση αυτού του κώδικα;

```
<?php
    // αυτό...
```

```
    $a = 5;
    $a = $a + 10;
    // ... είναι ίδιο με αυτό
    $a = 5;
    $a += 10;
?>
```

Δραστηριότητα: Ας επανέλθουμε πάλι στα String. Πόσοι τρόποι υπάρχουν να πούμε το ίδιο πράγμα; Εκτελέσετε τον παρακάτω κώδικα για να μάθετε τι εννοούμε. Τι καινούριο μάθατε από την εκτέλεση αυτού του κώδικα; Ποιος είναι ο ρόλος της τελείας «.»;

```
<?php
    // αυτό...
    echo "Hello, world!";
    // ... είναι ίδιο με αυτό
   echo "Hello," . " " . "world" . "!";
?>
```

5.4.2 Δομή Επιλογής

Ας θεωρήσουμε ότι θέλουμε να γράψουμε ένα πρόγραμμα το οποίο να ρωτάει αν η ηλικία σου είναι μεγαλύτερη ή ίση του 18. Αν η απάντηση είναι ναι, θέλουμε να εμφανίζεται ένα μήνυμα που να λέει «Μπορείς να ψηφίσεις!». Για να το πετύχουμε αυτό θα χρησιμοποιήσουμε την δομή επιλογής **if**, όπως φαίνεται στο παρακάτω παράδειγμα:

```
<?php
        $age = 19;
        if( $age > = 18 ) {
                echo " Μπορείς να ψηφίσεις!";
        }
?>
```

Στο παραπάνω παράδειγμα, χρησιμοποιήσαμε την εντολή if για να κάνουμε κάτι, να εμφανίσουμε δηλαδή ένα μήνυμα, αν η απάντηση στην ερώτηση είναι ναι δηλαδή **Αληθής** όπως λέμε στην γλώσσα προγραμματισμού PHP. Τι γίνεται στην περίπτωση που θέλουμε να εμφανίσουμε ένα μήνυμα που να λέει «Είσαι ακόμα μικρός!», όταν η απάντηση στην ερώτηση είναι όχι; Χρησιμοποιούμε τότε την σύνθετη δομή επιλογής **if / else** όπως φαίνεται στο παρακάτω παράδειγμα:

```
<?php
        $age = 17;
        if( $age > = 18 ) {
                echo " Μπορείς να ψηφίσεις!";
        }
        else {
                echo " Είσαι ακόμα μικρός!";
        }
?>
```

5.4.3 Δομή Επανάληψης

Όταν δημιουργούμε προγράμματα, συχνά πρέπει να επαναλάβουμε τα ίδια πράγματα πολλές φορές, όπως στο παρακάτω παράδειγμα που πρέπει να εμφανίσουμε αριθμούς από το 1 μέχρι και το 100. Πόσες φορές πρέπει να καλέσουμε την εντολή echo; Ας σημειώσουμε ότι το σύμβολο «//» στην αρχή της φράσης «και τα λοιπά και τα λοιπά» χρησιμοποιείται για να εισάγουμε σχόλια στο πρόγραμμά μας. Τα σχόλια είναι μόνο για εμάς και μας βοηθούν να θυμόμαστε τι κάνει το πρόγραμμά μας.

```
<?php
   echo 1;
   echo 2;
   echo 3;
   // και τα λοιπά και τα λοιπά
?>
```

Υπάρχει τρόπος να γράψουμε λιγότερες εντολές; Ας μελετήσουμε το παρακάτω παράδειγμα στο οποίο χρησιμοποιείται η δομή επανάληψης «for»:

```
<?php
      // Εμφανίστε τους αριθμούς από το 1 μέχρι και το 100
      for ($i = 1; $i < 101; $i = $i + 1) {
        echo $i;
      }
?>
```

Η δομή επανάληψης «for» ξεκινάει με την δεσμευμένη λέξη «for». Είναι σαν να λέμε στην PHP ετοιμάσου να εκτελέσεις αυτό που θα σου πούμε πολλές φορές. Στη συνέχεια προσθέτουμε ένα ζευγάρι από παρενθέσεις «()». Μέσα στις παρενθέσεις «()» λέμε στην PHP τρία πράγματα. Τα τρία αυτά πράγματα χωρίζονται μεταξύ τους με το σύμβολο «;» και λένε αντίστοιχα τα εξής: Πώς αρχίζει η επανάληψη, πότε τελειώνει η επανάληψη και τι κάνουμε κάθε φορά πριν πάμε στην επόμενη επανάληψη. Στη συνέχεια των παρενθέσεων ακολουθούν τα σύμβολα «{}». Μέσα στα σύμβολα «{}» λέμε στην PHP ποιές εντολές να εκτελεί σε κάθε επανάληψη.

Έτσι, το παραπάνω τμήμα κώδικα λέει στην PHP τα εξής: Ξεκίνα τον κύκλο των επαναλήψεων θέτοντας στην μεταβλητή $i την τιμή 1, σταμάτα τον κύκλο των επαναλήψεων πριν φτάσει η μεταβλητή $i την τιμή 101, αύξανε κάθε φορά την τιμή της μεταβλητής $i κατά 1 πριν πάς στην επόμενη επανάληψη και σε κάθε επανάληψη εμφάνιζε την τρέχουσα τιμή της μεταβλητής $i.

Δραστηριότητα: Στο παρακάτω πρόγραμμα προσπαθώ να εμφανίσω τους πέντε πρώτους άρτιους αριθμούς. Δηλαδή τους 2, 4, 6, 8, και 10. Κάτι όμως δεν κάνω καλά. Μπορείτε να βρείτε τι φταίει;

```
<html>
 <head>
  <title>Δομή επανάληψης For </title>
 </head>
 <body>
  <p>
   <?php
    // Εμφάνισε τους πέντε πρώτους άρτιους αριθμούς
    for ($i = 2; $i < 10; $i = $i + 1){
     echo $i;
    }
   ?>
  </p>
 </body>
</html>
```

Όταν θέλουμε να επαναλάβουμε ένα σύνολο εντολών πολλές φορές χωρίς όμως να γνωρίζουμε εκ των προτέρων τον αριθμό των επαναλήψεων τότε χρησιμοποιούμε την εντολή **while**. Θα ρωτήσετε πότε σταματάμε τον κύκλο των επαναλήψεων στην περίπτωση αυτή; Ο κύκλος των επαναλήψεων τερματίζει όταν μία συγκεκριμένη συνθήκη παύει να ισχύει, δηλαδή όταν γίνει ψευδής (false). Μελετήστε το παρακάτω παράδειγμα:

```
<?php
$loopCond = true;
while ($loopCond){
        echo "<p> Είμαστε μέσα στην επανάληψη </p>";
        $loopCond = false;
```

```
}
echo "<p> Βγήκαμε από την επανάληψη </p>";
 ?>
```

Η δομή επανάληψης «while» συντάσσεται με τον ακόλουθο τρόπο:

```
while(condition) {
  // εντολές που εκτελούνται κάθε φορά όσο η συνθήκη condition είναι αληθής
}
```

Με τις επαναλήψεις, πρέπει να είμαστε προσεκτικοί όσον αφορά τον τερματισμό τους. Θεωρείται ότι ο παρακάτω κύκλος επαναλήψεων θα τερματίσει κάποια στιγμή;

```
while(2 > 1){
  // εντολές
}
```

Δραστηριότητα: Ποια εντολή πρέπει να προσθέσουμε για να τερματίσει ο κύκλος των επαναλήψεων; Θυμηθείτε ότι ο Ο κύκλος των επαναλήψεων τερματίζει όταν η συνθήκη που συνοδεύει το while παύει να ισχύει, δηλαδή όταν γίνει ψευδής (false).

```
<?php
      $loopCond = true;
            while ($loopCond == true ){
            echo "<p> Η επανάληψη τρέχει. </p>";
                  // Πρόσθεσε μία εντολή που θα σταματήσει την επανάληψη
             }
      echo ".. και τώρα τελείωσε";
?>
```

Έχετε παρατηρήσει ότι η επανάληψη while ελέγχει πρώτα την συνθήκη που την συνοδεύει και στην περίπτωση που είναι αληθής εκτελείται ο κώδικας που περιέχεται μεταξύ των αγκύλων {}. Μία εναλλακτική περίπτωση της while αποτελεί η **do/while** που κάνει ακριβώς το αντίθετο. Δηλαδή, πρώτα εκτελείται η επανάληψη και στη συνέχεια ελέγχεται η συνθήκη.

Δραστηριότητα: Μελετήστε το παρακάτω παράδειγμα και εξηγείστε γιατί η η εντολή «echo $i» θα εκτελεστεί μόνο μία φορά.

```
<?php
```

```
$i = 0;
do {
   echo $i;
} while ($i > 0);
?>
```

5.4.4. Πίνακες

Ένας πίνακας είναι μια λίστα από αντικείμενα, δηλαδή σαν μία λίστα αγορών. Σας επιτρέπει να αποθηκεύσετε περισσότερα από ένα αντικείμενα σε μία μόνο μεταβλητή.

Σκεφτείτε το εξής. Όταν γράφετε μία λίστα με τα ψώνια σας, θα μπορούσατε να χρησιμοποιήσετε ένα ξεχωριστό φύλλο χαρτί για κάθε αντικείμενο που θέλετε να αγοράσετε. Το κάθε φύλλο χαρτί είναι και μια ξεχωριστή μεταβλητή. Μπορείτε να φανταστείτε τον εαυτό σας με τόσα χαρτιά στο χέρι όσα και τα πράγματα που θέλετε να αγοράσετε; Γιατί να μην χρησιμοποιήσετε ένα φύλλο χαρτί για όλα αυτά που θέλετε να αγοράσετε; Αυτό το φύλλο χαρτί το οποίο περιέχει όλα αυτά που θέλετε να αγοράσετε ονομάζεται **πίνακας (array)**. Στο παρακάτω παράδειγμα, ο πίνακας με όνομα $array περιέχει τα στοιχεία «ντομάτες», «αυγά» και «ζάχαρη».

```
<html>
  <head>
    <title>Πίνακες!</title>
  </head>
  <body>
    <?php
      $array = array("ντομάτες", "αυγά", "ζάχαρη");
    ?>
  </body>
</html>
```

Δραστηριότητα: Ποιο στοιχείο του πίνακα θα εμφανιστεί στο παρακάτω παράδειγμα;

```
<?php
    $myArray = array("do", "re", "mi");
    echo $myArray[0]
?>
```

Δραστηριότητα: Εξηγείστε τι συμβαίνει στο παρακάτω τμήμα κώδικα.

```
<?php
      $languages = array("HTML/CSS", "JavaScript", "I don’t know", "Python",
"Ruby");
      $languages[2] = "PHP";
      echo $languages[2] ;
?>
```

5.4.5. Συναρτήσεις

Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα κομμάτια κώδικα που μπορείτε να χρησιμοποιήσετε μέσα στο πρόγραμμά σας. Η PHP έχει πολλές ενσωματωμένες συναρτήσεις. Μία από αυτές είναι η strlen() που επιστρέφει τον αριθμό των χαρακτήρων που υπάρχουν σε μία λέξη (ή πιο σωστά σε ένα αλφαριθμητικό). Στο παρακάτω παράδειγμα εμφανίζεται ο αριθμός 5, όσοι είναι και οι χαρακτήρες της λέξης «Μαρία».

```
<?php
   $length = strlen("Μαρία");
   print $length;
?>
```

Εκτός από τις ενσωματωμένες συναρτήσεις PHP που μας παρέχονται μπορούμε να δημιουργήσουμε και τις δικές μας, όπως φαίνεται στο παρακάτω παράδειγμα, όπου εμφανίζουμε το όνομά μας.

```
<?php
    function displayname() {
       echo "Το όνομά μου είναι Πέτρος";
    }
displayname();
?>
```

Οι συναρτήσεις δεν θα ήταν τόσο χρήσιμες αν δεν μπορούσαν να δεχτούν κάποια δεδομένα, ας τα ονομάσουμε δεδομένα εισόδου (input), τα οποία θα μπορούσαν στη συνέχεια να τα επεξεργαστούν. Ας ονομάσουμε τα δεδομένα εισόδου παράμετροι (parameters) ή ορίσματα (arguments). Οι παράμετροι είναι μεταβλητές, τις τιμές των οποίων χρησιμοποιεί η συνάρτηση για να κάνει υπολογισμούς.

Στο παρακάτω τμήμα κώδικα η συνάρτηση δέχεται μία παράμετρο, την οποία την πολλαπλασιάζει με τον εαυτό της και στη συνέχεια εμφανίζει το αποτέλεσμα της αριθμητικής πράξης.

```
<?php
        function squareValue($number) {
                echo $number * $number;
        }
        $n = 6;
        // Εμφανίζεται ο αριθμός 36
        squareValue($n);
?>
```

Δραστηριότητα: Μπορείτε να εξηγήσετε τι κάνει το παρακάτω πρόγραμμα;

```
<?php
    function aboutMe($name, $age) {
       echo "Hello! My name is " . $name . " and I am " .  $age  ." years old.";
    }
    aboutMe( "Maria", 17);
?>
```

Δραστηριότητα: Γράψτε τον κώδικα μίας συνάρτησης την οποία θα ονομάσετε greetings. Η συνάρτηση αυτή θα δέχεται ως παράμετρο το $name και θα εμφανίζει με την βοήθεια του echo ένα string το οποίο θα είναι ίσο με "Greetings, " . $name . "!". Στη συνέχεια καλέστε την συνάρτηση greetings() με το όνομά σας για να δοκιμάσετε τι έχετε κάνει.

Δραστηριότητα: Εξηγείστε τι συμβαίνει στο παρακάτω τμήμα κώδικα. Ποιος είναι ο ρόλος του return σε σχέση με το echo;

```
<?php
   function sum($a, $b) {
      $c = $a + $b;
      return $c;
   }
```

```
    echo sum(2,13) ;
?>
```

5.4.6. Πρόσβαση και διαχείριση Βάσεων Δεδομένων MySQL από το διαδίκτυο χρησιμοποιώντας PHP

Ο συνδυασμός της PHP με την βάση δεδομένων MySQL και με την υποστήριξη ενός διαδικτυακού εξυπηρετητή (web server) μας επιτρέπει να δημιουργήσουμε διαδικτυακές εφαρμογές προσαρμοσμένες στις ανάγκες των χρηστών. Ας δούμε την λειτουργία της αρχιτεκτονικής των διαδικτυακών βάσεων δεδομένων, μέσω ενός παραδείγματος αυτό της αναζήτησης βιβλίων.

- Το πρόγραμμα περιήγησης ενός χρήστη κάνει μία http:// αίτηση για μια συγκεκριμένη ιστοσελίδα. Ο χρήστης αναζητεί τα βιβλία ενός συγκεκριμένου συγγραφέα χρησιμοποιώντας μία HTML φόρμα. Συμπληρώνει μία φόρμα με τα στοιχεία που αναζητεί και κάνει κλικ στο κουμπί «Υποβολή». Στην περίπτωση αυτή καλείται για παράδειγμα ένα σενάριο που ονομάζεται results.php.

- Ο διαδικτυακός εξυπηρετητής (web server) λαμβάνει ένα αίτημα για το results.php. Βρίσκει το αρχείο αυτό και το δίνει στην μηχανή (engine) PHP για επεξεργασία.

- Η μηχανή (engine) PHP αρχίζει να αναλύει το σενάριο. Μέσα στο σενάριο περιέχονται εντολές σύνδεσης με την βάση δεδομένων και εκτέλεσης του ερωτήματος (αναζήτηση των βιβλίων). Η PHP ανοίγει μία σύνδεση με τον εξυπηρετητή MySQL και στέλνει το ερώτημα.

- Ο εξυπηρετητής MySQL λαμβάνει το ερώτημα, το επεξεργάζεται και στέλνει πίσω στην μηχανή PHP τα αποτελέσματα, δηλαδή την λίστα με τα βιβλία.

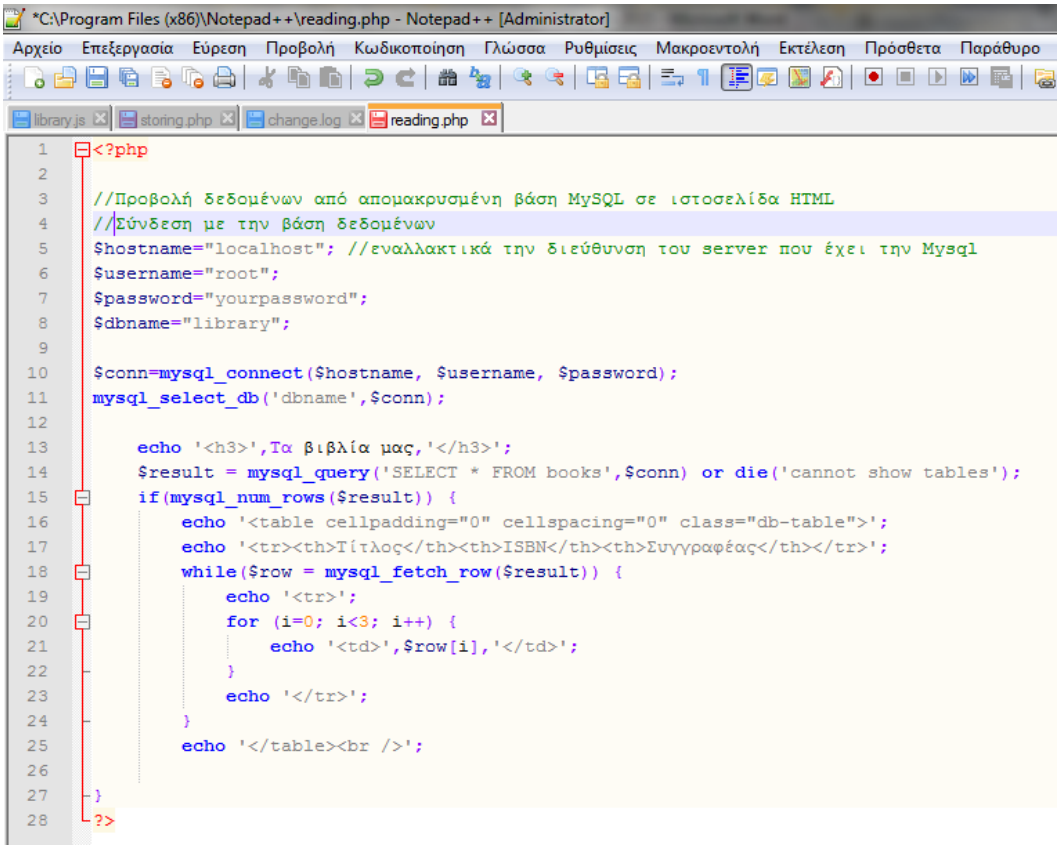
- Η μηχανή PHP μορφοποιεί κατάλληλα τα αποτελέσματα σε μορφή κώδικα HTML και τα επιστρέφει στον διαδικτυακό εξυπηρετητή.

- Ο διαδικτυακός εξυπηρετητής τα στέλνει στη συνέχεια στο πρόγραμμα περιήγησης, όπου ο χρήστης μπορεί να δει την λίστα με τα βιβλία που ζήτησε.

Παρακάτω εμφανίζονται τα βασικά τμήματα του περιεχόμενου του σεναρίου results.php που ανακτά τα αποτελέσματα αναζήτησης από την βάση δεδομένων MySql και τα μορφοποιεί για παρουσίαση.

5.4.7. Προβολή δεδομένων βάσης δεδομένων MySQLσε ιστοσελίδα HTML

Βασική λειτουργία της PHP αποτελεί η διασύνδεση της με βάση δεδομένων (MySQL) με σκοπό την προβολή τους σε ιστοσελίδα HTML. Στο παράδειγμα μας, στην απομακρυσμένη βάση δεδομένων library υπάρχει ο πίνακας books με τα πεδία btitle, isbn και bauthor. Σκοπός του παραδείγματος (σχήμα 5.23) είναι η προβολή των στοιχείων όλων των βιβλίων που είναι αποθηκευμένα στην βάση σε ιστοσελίδα HTML.



***Σχήμα 5.23**: Προβολή δεδομένων σε HTML από MySQL*

Αφού γίνει σύνδεση με τον απομακρυσμένο υπολογιστή (mysql_connect εντολές 5,6,7,10) επιλέγεται η βάση δεδομένων library (mysql_select_db εντολές 8, 11). Κατόπιν χρησιμοποιείται η εντολή echo προκειμένου να εισαχθεί στην ιστοσελίδα (μέσω PHP) η ετικέτα HTML<H3>. Στη συνέχεια υποβάλλεται το ερώτημα επιλογής όλων των δεδομένων του πίνακα books (mysql_query εντολή 14). Εφόσον το ερώτημα επιστρέψει δεδομένα (εντολή 15 – έλεγχος αν $result είναι διαφορετικό του null) δημιουργείται η κεφαλίδα του πίνακα προβολής (εντολή echo 16,17). Η επανάληψη while (εντολή 18) στον πίνακα $row φορτώνει τα δεδομένα κάθε γραμμής (σε κάθε επανάληψη) του πίνακα books. Το εμφωλευμένο for (εντολές 20, 21, 22) τυπώνει σε κελί πίνακα HTML το κάθε πεδίο της γραμμής ($row) του πίνακα books.

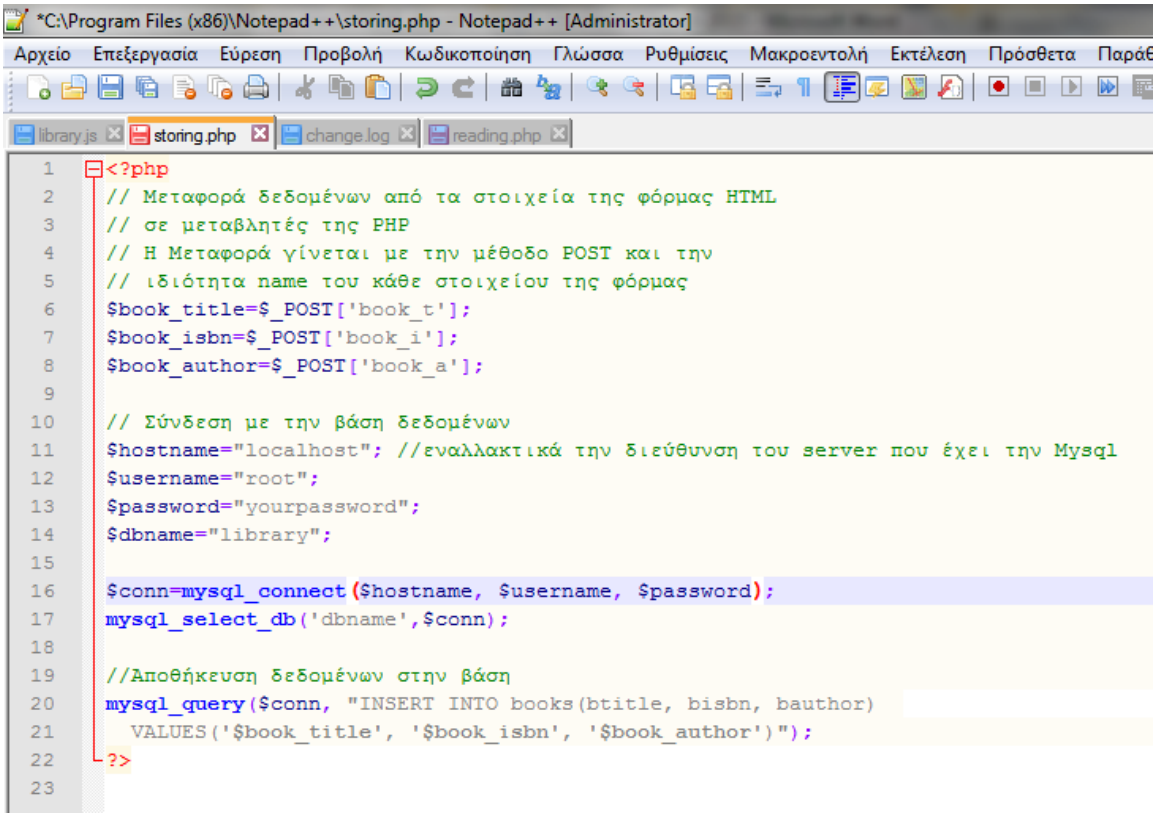
5.4.8. Εισαγωγή Δεδομένων φόρμας σε Βάση Δεδομένων MySQL από το διαδίκτυο χρησιμοποιώντας PHP

Η αντίστροφη λειτουργία με αυτήν που περιγράφηκε στην προηγούμενη ενότητα είναι αυτή της εισαγωγής δεδομένων από φόρμα HTML και αποθήκευση τους σε απομακρυσμένη βάση δεδομένων MySQL. Στην απομακρυσμένη βάση δεδομένων library υπάρχει ο πίνακας books με τα πεδία btitle, isbn και bauthor. Στο πρώτο βήμα δημιουργείται φόρμα HTML όπως φαίνεται στο σχήμα 5.24.

```
<HTML>
  <HEAD>
   <TITLE>Βιβλιοθήκη</TITLE>
  </HEAD>
  <BODY>
  <H1> Εισαγωγή βιβλίου</H1>

  <FORM action="storing.php" method="post">

     Book Title <INPUT type="text" name="book_t"  value=""><br />
     Book ISBN <INPUT type="text" name="book_i"  value=""><br />
     Book Author <INPUT type="text" name="book_a"  value=""><br />

     <INPUT type="submit" name="submit" value="submit">
  </FORM>
  </BODY>
</HTML>
```

***Σχήμα 5.24**: Φόρμα εισαγωγής βιβλίων*

Στην ιδιότητα action της φόρμας εκχωρείται το όνομα του αρχείου PHP (storing.php) του οποίου ο κώδικας θα εκτελεστεί όταν πατηθεί το πλήκτρο "submit". Με την ιδιότητα method δηλώνεται ο τρόπος μεταφοράς δεδομένων (POST) από την HTML στην PHP. Ο κώδικας PHP που αποθηκεύει τα δεδομένα σε μία απομακρυσμένη βάση δεδομένων περιγράφεται στο σχήμα 5.25.



***Σχήμα 5.25**: Κώδικας PHP αποθήκευσης δεδομένων*

Αρχικά αποθηκεύονται οι τιμές από την φόρμα HTML μέσω του POST σε μεταβλητές της PHP. Κατόπιν γίνεται η σύνδεση στην απομακρυσμένη βάση δεδομένων και υποβάλλεται το ερώτημα αποθήκευσης.

5.4.9. Σύνοδοι/Sessions στην PHP

Στα προηγούμενα παραδείγματα μάθαμε το πως αποθηκεύονται και το πώς ανακτώνται δεδομένα σε/από μια βάση δεδομένων. Λειτουργίες πολύ χρήσιμες για την υλοποίηση ενός πληροφοριακού συστήματος. Παρόλα αυτά, δεν επαρκούν (με τον τρόπο με τον οποίο περιγράφηκαν στις προηγούμενες ενότητες) καθώς αυτές οι λειτουργίες είναι διαθέσιμες για όλους τους χρήστες του διαδικτύου. Όταν όμως κάνουμε είσοδο (login) στον e-mail λογαριασμό μας και μετά, το περιεχόμενο είναι διαθέσιμο μόνο για εμάς μολονότι δουλεύοντας με το e-mail μας πλοηγούμαστε σε διαφορετικές σελίδες του ίδιου ιστότοπου. Με άλλα λόγια υπάρχει η ανάγκη μεταφοράς δεδομένων (ταυτότητα χρήστη) από σελίδα σε σελίδα.

Οι σύνοδοι είναι μεταβλητές με τις οποίες μπορούν να μεταφερθούν δεδομένα από σελίδα σε σελίδα. Σε αντίθεση με τα cookies (αποθήκευση δεδομένων χρήστη στον φυλλομετρητή) οι σύνοδοι (sessions) αποθηκεύονται στον εξυπηρετητή (server). Για την δημιουργία sessions χρησιμοποιείται η εντολή:

session_start();

Ενώ για την αποθήκευση δεδομένων χρησιμοποιείται ο πίνακας $_SESSION. Ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει όσες θέσεις θέλει σε αυτόν τον πίνακα. Ως δείκτες μπορεί να χρησιμοποιήσει όποια ονομασία επιθυμεί και φυσικά να εισάγει οποιαδήποτε τιμή όπως φαίνεται παρακάτω:

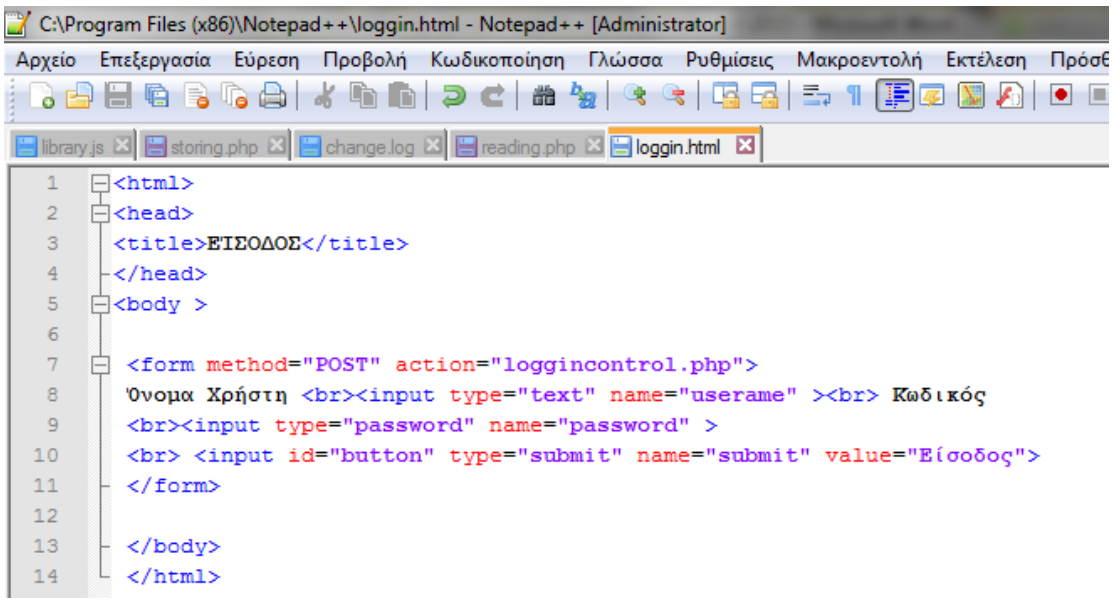
$_SESSION[“loggedIn”]=”true’;

$_SESSION[“username”]=”someuser”;

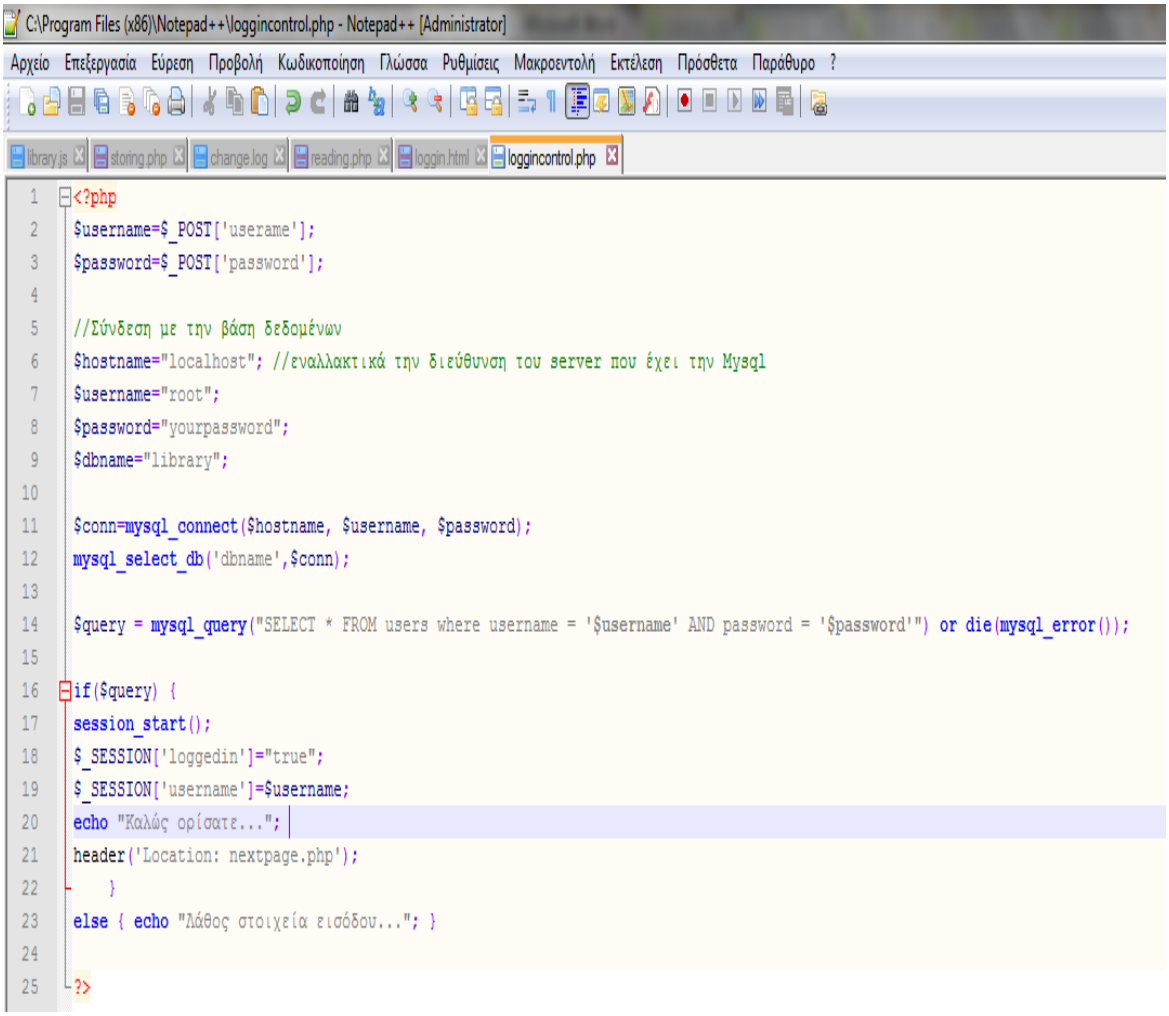
Οποιαδήποτε επόμενη σελίδα εκκινήσει το session (session_start()) μπορεί να διαβάσει οποιαδήποτε θέση του πίνακα $_SESSION με τον κατάλληλο δείκτη (π.χ. $_SESSION[“username”]) ή να αλλάξει την τιμή οποιασδήποτε θέσης με εκχώρηση. Για να κλείσει το session (π.χ. logout) πρέπει μια σελίδα να καλέσει την εντολή:

session_destroy();

Στο παράδειγμα που ακολουθεί δημιουργείται ιστοσελίδα στην οποία ο χρήστης εισάγει σε φόρμα το όνομα χρήστη και τον κωδικό του, γίνεται έλεγχος ορθής εισόδου με την βάση δεδομένων και σε περίπτωση επιτυχίας δημιουργείται session με το οποίο στην επόμενη σελίδα εμφανίζει το όνομα χρήστη. Αρχικά δημιουργούμε το αρχείο loggin.html (σχήμα 5.26) όπου εμφανίζεται μια φόρμα με τα στοιχεία εισόδου και όταν πατηθεί το κουμπί submit εκτελείται ο κώδικας στο αρχείο loggincontrol.php (σχήμα 5.27).

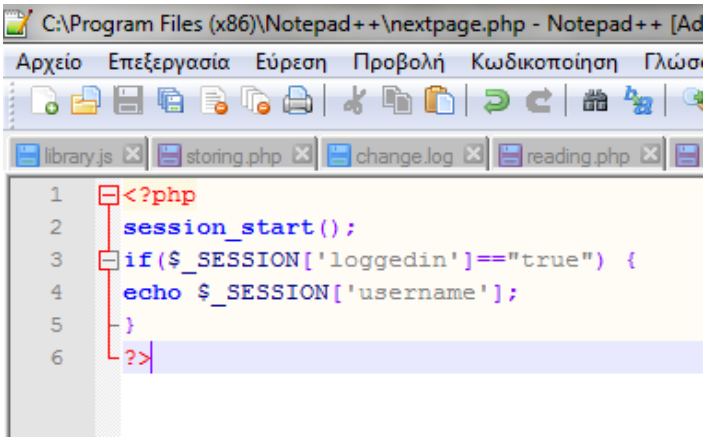


***Σχήμα 5.26**: Φόρμα στοιχείων εισόδου*



***Σχήμα 5.27**: Έλεγχος στοιχείων εισόδου*

Μέσω post μεταφέρονται τα δεδομένα εισόδου του χρήστη στην PHP. Αφού γίνει σύνδεση στη βάση, γίνεται ερώτημα στον πίνακα users για εγγραφή με όνομα χρήστη και κωδικό ίδιο με αυτό που εισήγαγε ο χρήστης. Σε περίπτωση επιτυχίας δημιουργείται το session με τις γνωστές μεταβλητές και μεταφέρεται η ροή στην σελίδα nextpage.php (σχήμα 5.28) όπου γίνεται έλεγχος στην $_SESSION[‘loggedin”] και αν είναι “true” τυπώνεται το όνομα χρήστη μέσω της αντίστοιχης session μεταβλητής.

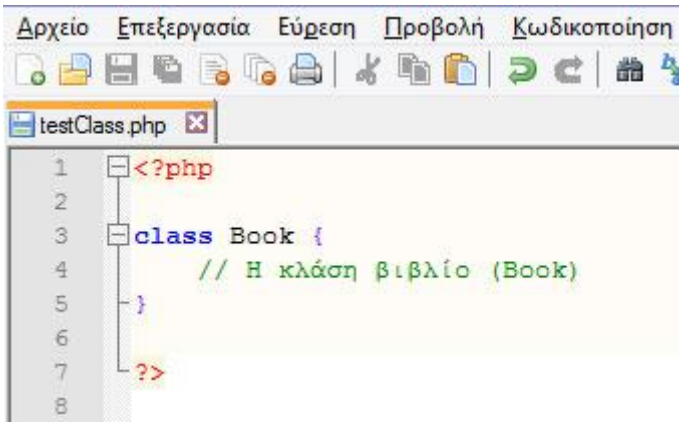


***Σχήμα 5.28**:nextpage.php*

5.4.10. Δημιουργία κλάσεων στην php

Στο 3ο κεφάλαιο παρουσιάστηκε η αντικειμενοστραφής προσέγγιση ανάλυσης απαιτήσεων και στο 4ο κεφάλαιο ο σχεδιασμός προσανατολισμένος στα αντικείμενα. Στη φάση της υλοποίησης δημιουργούνται οι κλάσεις στη γλώσσα προγραμματισμού που έχει επιλεχθεί για να αναπτυχθεί το σύστημα.

Η διαδικασία για να δημιουργηθεί μια κλάση στην γλώσσα προγραμματισμού php είναι αρκετά απλή. Αρχικά, συνήθως σε ένα νέο αρχείο (π.χ. το αρχείο testClass.php), δηλώνουμε την κλάση γράφοντας την δεσμευμένη λέξη **class**, το όνομα της κλάσης και τις αγκύλες **{}**:



Η κλάση αποτελεί το πρότυπο βάσει του οποίου δημιουργούνται τα αντικείμενα κατά τον χρόνο εκτέλεσης ενός προγράμματος. Στην php μπορούμε δημιουργήσουμε ένα αντικείμενο κατά τον χρόνο εκτέλεσης ενός προγράμματος και να το αποθηκεύσουμε σε μια μεταβλητή. Για να δημιουργήσουμε ένα αντικείμενο π.χ. τύπου «Book» χρησιμοποιούμε την δεσμευμένη λέξη **new** ως εξής:

```
$book = new Book;
```

Όπως έχει αναφερθεί στο κεφάλαιο 4.3 οι κλάσεις αποτελούνται από τις ιδιότητες και τις μεθόδους. Οι ιδιότητες δηλώνονται μέσα στην κλάση ως μεταβλητές:

```
class Book {
    var $title = "Η ιδιότητα τίτλος!!!";

}
```

Οι μέθοδοι δηλώνονται μέσα στην κλάση ως συναρτήσεις:

```
class Book {
    var $title = "Η ιδιότητα τίτλος!!!";

    function getTitle() {
        // Η μέθοδος πάρε ή δες Τίτλο!!!
    }
}
```

Οι ιδιότητες και οι μέθοδοι που περιέχονται σε κάθε αντικείμενο προσεγγίζονται από το αντικείμενο με τη χρήση του βέλους (->):

```
echo $bookObj->getTitle();
```

Για να αναφερθούμε στη μέθοδο ή στις ιδιότητες μιας κλάσης μέσα από την ίδια την κλάση χρησιμοποιούμε τη λέξη **$this**:

```
class Book {
    var $title = "Η ιδιότητα τίτλος!!!";

    function getTitle() {
        return $this->title;
    }

    function setTitle($t) {
        // Η μέθοδος θέσε Τίτλο!!!
        $this->title = $t;
    }
}
```

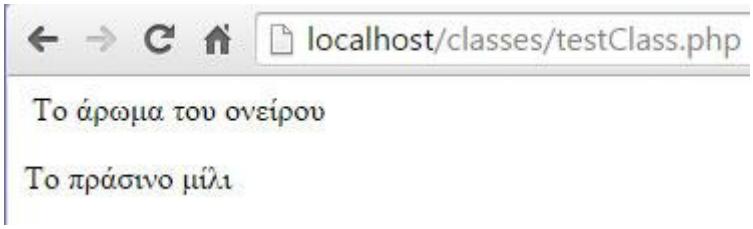
Εκτός απ' τις μεθόδους και τις μεταβλητές που έχουν αποτυπωθεί στη φάση του σχεδιασμού, όλες οι κλάσεις περιλαμβάνουν εξ' ορισμού κάποιες μεθόδους όπως είναι η μέθοδος constructor (κατασκευαστής). Η μέθοδος constructor καλείται κάθε φορά που δημιουργείται ένα νέο αντικείμενο και έχει το **όνομα της κλάσης** ή το όνομα **__construct**. Η μέθοδος constructor μπορεί να παίρνει σαν είσοδο ένα ή περισσότερα ορίσματα:

```php
class Book {
    var $title;

    function Book($t) {
        $this->title = $t;
    }

    function getTitle() {
        return $this->title;
    }

    function setTitle($t) {
        $this->title = $t;
    }
}
```

Για να δημιουργηθούν αντικείμενα της παραπάνω κλάσης δίνονται σαν είσοδο τα κατάλληλα δεδομένα που ορίζονται στη μέθοδο constructor:

```php
$book1Obj = new Book("Το άρωμα του ονείρου");
$book2Obj = new Book("Το πράσινο μίλι");

echo $book1Obj->getTitle() . "<p>";
echo $book2Obj->getTitle();
```

Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση του αρχείου testClass.php:



Δραστηριότητα: Να δημιουργήσετε την κλάση Book με τις κατάλληλες ιδιότητες και μεθόδους.

5.4.11. Αντικείμενα και βάσεις δεδομένων

Τα αντικείμενα μπορούν να συνδέονται με τη βάση δεδομένων, να ανακτούν δεδομένα, να αποθηκεύουν ή να διαγράφουν δεδομένα. Ένα παράδειγμα MySQL πίνακα για να αποθηκεύονται τα βιβλία στη βάση δεδομένων φαίνεται παρακάτω:

```
CREATE TABLE IF NOT EXISTS `book` (
 `aa` varchar(10) NOT NULL DEFAULT '0',
 `title` varchar(150) NOT NULL DEFAULT '',
 `writer` varchar(100) NOT NULL DEFAULT '',
```

```
  `publisher` varchar(100) NOT NULL DEFAULT '',
  `edition` varchar(100) NOT NULL DEFAULT '',
  `no_of_pages` smallint(4) NOT NULL DEFAULT '0',
  `category` varchar(10) NOT NULL DEFAULT '0',
  `cover` varchar(100) NOT NULL DEFAULT 'default.png',
  `review` mediumtext NOT NULL,
  PRIMARY KEY `aa` (`aa`),
  KEY `idx_cat` (`category`),
  KEY `idx_aa` (`aa`),
  KEY `idx_writer` (`writer`)
);
```

όπου '*aa*' είναι ο αύξων αριθμός που δίνει ο υπεύθυνος της βιβλιοθήκης στο βιβλίο, το '*category*' είναι η κατηγορία που ανήκει το βιβλίο π.χ. λογοτεχνικό, επιστημονικό κτλ, '*cover*' είναι το όνομα της εικόνας – εξώφυλλο του βιβλίου και '*review*' είναι μια μικρή περίληψη για το περιεχόμενο του βιβλίου.

Με τη δημιουργία ενός αντικειμένου Book θα πρέπει να γίνει εισαγωγή των κατάλληλων δεδομένων στον πίνακα. Έτσι θα πρέπει να δημιουργηθεί ένα πεδίο στον παραπάνω πίνακα με τη χρήση της εντολής **INSERT**:

```
INSERT INTO book (`aa`, `title`, `writer`, `publisher`, `edition`, `no_of_pages`, `category`, `cover`, `review`) VALUES ('1', 'Ματωμένα χώματα', 'Διδώ Σωτηρίου', 'Κέδρος', '2008', 344, 'Λογοτεχνία', 'matomena_xomata.jpg', '«Πόλεμοι και ξανά πόλεμοι! Τι στο καλό θα βγάλει η η εποχή μας...');
```

Η συγκεκριμένη εντολή **INSERT** θα πρέπει να καλείται από τη μέθοδο constructor ή από οποιαδήποτε άλλη μέθοδο δημιουργηθεί γι' αυτό τον σκοπό όπως π.χ. μια μέθοδος storeBook. Η μέθοδος αυτή θα πρέπει να παίρνει σαν είσοδο όλα τα δεδομένα που χρειάζονται για να δημιουργηθεί η καταχώρηση στη βάση:

```
function Book($aa, $t, $w, $p, $e, $nop, $cat, $co, $r) {
    $book_iquery_str = "INSERT INTO book (`aa`, `title`, `writer`, `publisher`, `edition`,`no_of_pages`, `category`, `cover`, `review`) VALUES ('$aa', '$t', '$w', '$p', '$e', $nop, '$cat', '$co', '$r');";
    $book_iquery = mysql_query($book_iquery_str)  or die (mysql_error());
}
```

Η εντολή **mysql_query** στέλνει ερωτήματα SQL στη βάση δεδομένων και επιστρέφει μια λίστα με καταχωρήσεις που επιστρέφονται από το ερώτημα. Υπενθυμίζεται ότι

πριν γίνει ένα ερώτημα προς τη βάση θα πρέπει να έχει γίνει σύνδεση με τον εξυπηρετητή MySQL και επιλογή της κατάλληλης βάσης δεδομένων.

Αντίστοιχα η εντολή setTitle θα πρέπει να καλεί ένα ερώτημα **UPDATE** προς τη βάση για να τροποποιηθεί ο τίτλος του βιβλίου που έχει αποθηκευτεί στη βάση:

```
function setTitle($t) {
    $uquery_str = "UPDATE `book` SET title='$t' WHERE aa='$this->aa';";
    $uquery = mysql_query($uquery_str) or die (mysql_error());
    $this->title = $t;
}
```

Επίσης είναι απαραίτητη μια μέθοδος που ανακτά τα δεδομένα από τη βάση δεδομένων στις ιδιότητες ενός αντικειμένου όπως π.χ. το retrieveBook. Η μέθοδος μπορεί να δημιουργηθεί με τη χρήση της εντολής **SELECT**:

```
function retrieveBook($aa) {
    $this->aa = $aa;
    $book_squery_str = "SELECT * FROM book WHERE aa='$aa';";
    $book_squery = mysql_query($book_squery_str)  or die (mysql_error());
    $row = mysql_fetch_array ($book_squery);
    $this->title = $row['title'];
    $this->writer = $row['writer'];
    $this->publisher = $row['publisher'];
    $this->edition = $row['edition'];
    $this->no_of_pages = $row['no_of_pages'];
    $this->categoty = $row['category'];
    $this->cover = $row['cover'];
    $this->review = $row['review'];
}
```

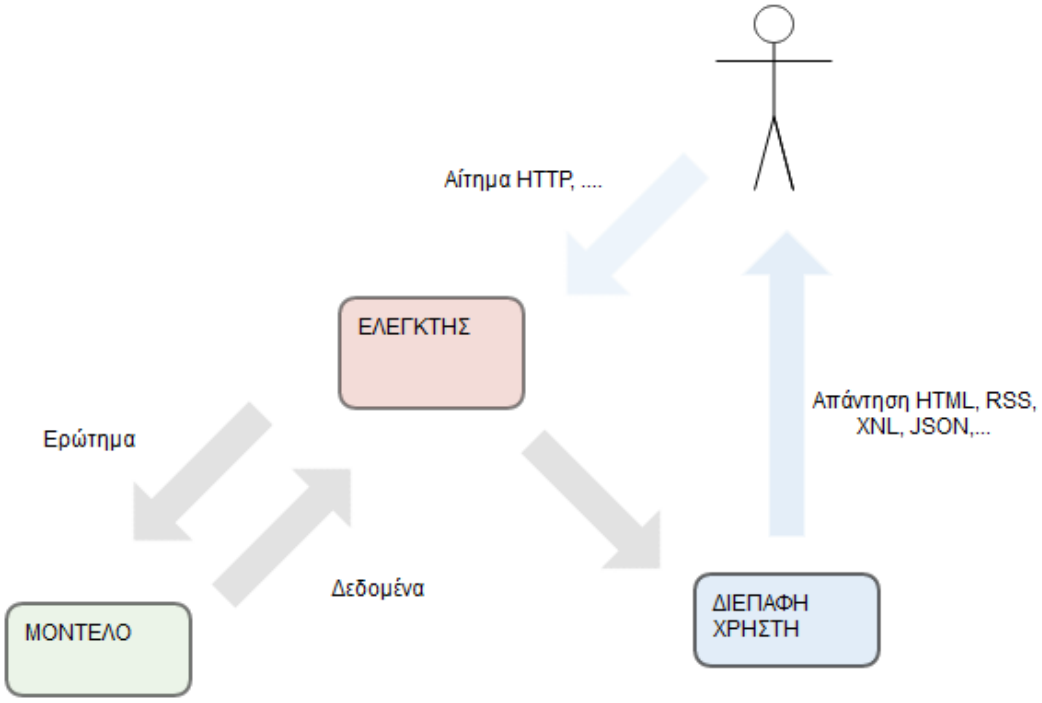
Η εντολή **mysql_fetch_array** επιστρέφει έναν πίνακα με τα αποτελέσματα του ερωτήματος SQL προς τη βάση.

Δραστηριότητα: Να εμπλουτίσετε την κλάση Book με τις κατάλληλες μεθόδους που θα διατυπώνουν ερωτήματα προς τον πίνακα Book της βάση δεδομένων.

5.4.12. MVC Πλαίσια για την δημιουργία διαδικτυακών PHP εφαρμογών (PHP Web Applications)

Η PHP είναι μια σταθερή πλατφόρμα, αλλά είναι ακόμα πιο ισχυρή όταν χρησιμοποιούμε ένα από τα πλαίσια MVC (Model View Controler), όπως το CakePHP, το Zend, το Symfony και το CodeIgniter που μας επιτρέπουν να συντομεύσουμε τον χρόνο ανάπτυξης και να οργανώσουμε καλύτερα τον κώδικά μας. Το πλαίσιο MVC αποτελεί το σκελετό γύρω από τον οποίο κτίζεται η εφαρμογή μας. Το πρότυπο MVC όπως φαίνεται στο παρακάτω σχήμα 5.9 χωρίζει μία εφαρμογή σε τρία συνεκτικά μέρη: στο μοντέλο (model), στη διεπαφή χρήστη (view) και στον ελεγκτή (controller).

Το μοντέλο (Model) είναι υπεύθυνο για την αποθήκευση και την ανάκτηση των δεδομένων. Αποτελεί εκείνο το μέρος της εφαρμογής που περιλαμβάνει την διαχείριση των δεδομένων. Η διεπαφή χρήστη (View) είναι υπεύθυνη για το πώς τα δεδομένα αυτά θα παρουσιαστούν στον χρήστη. Αποτελεί εκείνο το μέρος της εφαρμογής που περιλαμβάνει την παραγωγή κώδικα HTML και την αλληλεπίδραση με το χρήστη. Ο ελεγκτής χειρίζεται τα αιτήματα του πελάτη σε συνεργασία με το μοντέλο (Model) και την διεπαφή χρήστη (View). Δέχεται το αίτημα του χρήστη, το επεξεργάζεται, το αναλύει και στη συνέχεια στέλνει το αντίστοιχο ερώτημα στο μοντέλο (Model). Η απάντηση που θα πάρει από το μοντέλο (Model) θα την προωθήσει στη διεπαφή χρήστη (View) για την κατάλληλη μορφοποίηση.



***Σχήμα 5.25**: Ένα βασικό MVC αίτημα*

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Η τεχνολογία ενισχύει την ικανότητα των εκπαιδευτικών ιδρυμάτων και ινστιτούτων να προσφέρουν εκπαιδευτικό περιεχόμενο σε μορφή λιγότερο ή

περισσότερο δομημένη σε όλους ανεξάρτητα από γεωγραφικά σύνορα. Χαρακτηριστικά παραδείγματα των παραπάνω αποτελούν το Coursera, το edX, το Khan Academy και το YouTube EDU. Επισκεφτείτε τους παραπάνω ιστότοπους και παρουσιάστε στην τάξη τους στόχους τους, το είδος του εκπαιδευτικού περιεχομένου καθώς και το πώς θα μπορούσαν να χρησιμοποιηθούν για την υποστήριξη των παραδοσιακών τρόπων παροχής της εκπαίδευσης.

2. Στον τομέα της υγείας, από την πλευρά του ασθενούς, πώς θα νιώθατε σε περίπτωση που χρειάζονταν να γίνει διάγνωση από έναν γιατρό ο οποίος θα βρίσκονταν εκατοντάδες ή χιλιάδες χιλιόμετρα μακριά από εσάς; Τι θα περιμένατε ή τι θα σας προβλημάτιζε περισσότερο αν βρισκόσασταν στην θέση αυτή;

Βιβλιογραφία

O'Brien J.A & Marakas G.(2008). *Introduction to Information Systems*, 14th/edition, McGraw-Hill, New York.

Quickly E. & Gargenta M. (2009). *PHP and MySQL by Example, 1st Edition*, Prentice Hall, Boston, MA.

McPeak J. & Wilton P. (2015). *Beginning JavaScript*, 5th edition, WROX.

Robson E. & Freeman E. (2005). *Head first HTML with CSS and XHTML*, O'Reilly Media.

Satzinger J.W., Jackson R.B. & Burd S.D (2009). *Systems Analysis and Design in a Changing World*, Boston, MA: Thomson Course Technology.

Shklar L. & Rosen R (2003). *Web Application Architecture, Principles, protocols and practices.* John Wiley & Sons, England.

Διαδικτυακές Πηγές

- HTML The language for building web pages http://www.w3schools.com/default.asp
- Intro to HTML/CSS: Making webpages https://www.khanacademy.org/computing/computer-programming/html-css
- Learn how to create websites by structuring and styling your pages with HTML and CSS http://www.codecademy.com/en/tracks/web
- Learn to program in PHP http://www.codecademy.com/learn
- PHP 5 tutorial http://www.w3schools.com/php/
- PHP Documentation http:// php.net

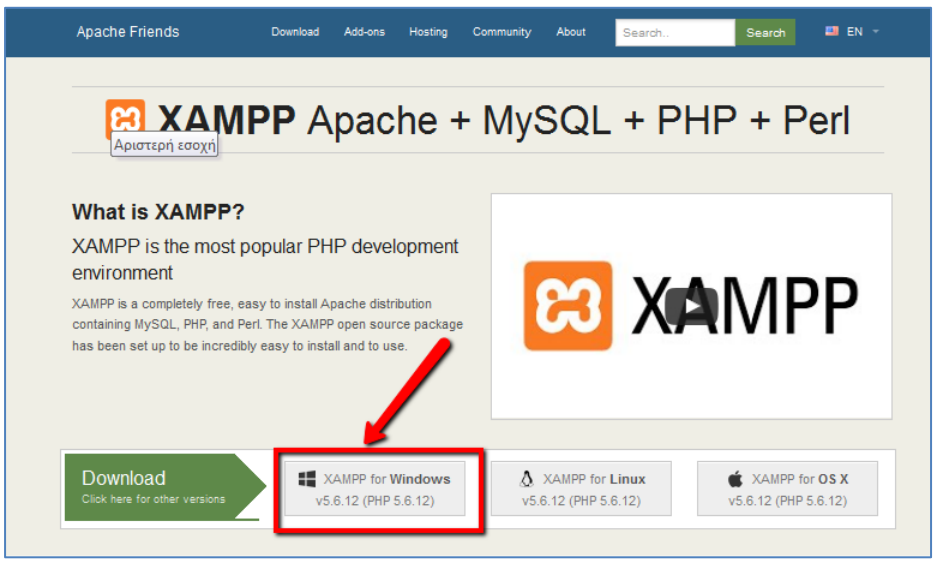
ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα αυτό παρέχονται γενικές οδηγίες για το πώς μπορούμε να εγκαταστήσουμε τοπικά php (γλώσσα προγραμματισμού για δημιουργία δυναμικών ιστοσελίδων), mysql (βάση δεδομένων), apache (web server) και phpmyadmin (εργαλείο διαχείρισης της MySQL) με τη βοήθεια του xampp. Παρουσιάζονται επιπλέον, απλά προγράμματα σε PHP διαχείρισης βάσεων δεδομένων, όπως σύνδεση με την βάση δεδομένων, δημιουργία πίνακα, εισαγωγή δεδομένων και εμφάνιση δεδομένων πίνακα. Τέλος, παρατίθενται ενδεικτικά παραδείγματα διαχείρισης βάσεων δεδομένων μέσω phpmyadmin. Ας σημειώσουμε, ότι μπορούμε να εγκαταστήσουμε τα συστατικά του xampp μεμονωμένα, ξεχωριστά το ένα από το άλλο.

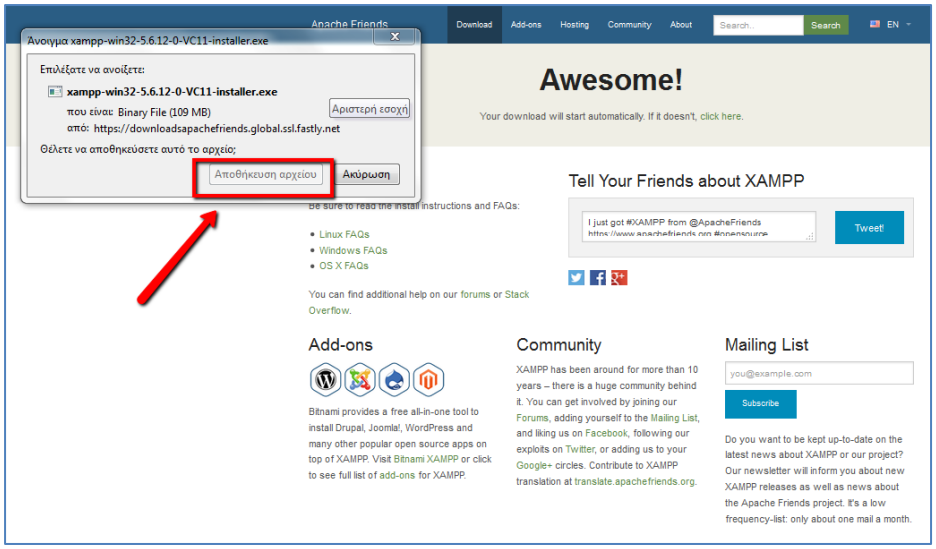
Εγκατάσταση xampp

Το XAMPP είναι ένα δωρεάν πρόγραμμα για διάφορα λειτουργικά συστήματα που δημιουργεί ένα περιβάλλον ενός τοπικού διακομιστή (Server) στον υπολογιστή μας, ο οποίος μπορεί να μεταφράσει δεδομένα της γλώσσας προγραμματισμού PHP και της γλώσσας βάσεων δεδομένων. Το XAMPP είναι ιδανικό για τις δοκιμές μας, χωρίς αυτό να σημαίνει ότι δεν θα μπορούσαμε να αναπτύξουμε παρόμοιες εφαρμογές και για ένα απομακρυσμένο διακομιστή καθώς οι περισσότεροι web διακομιστές υποστηρίζουν τα ίδια συστατικά με αυτά του xampp.

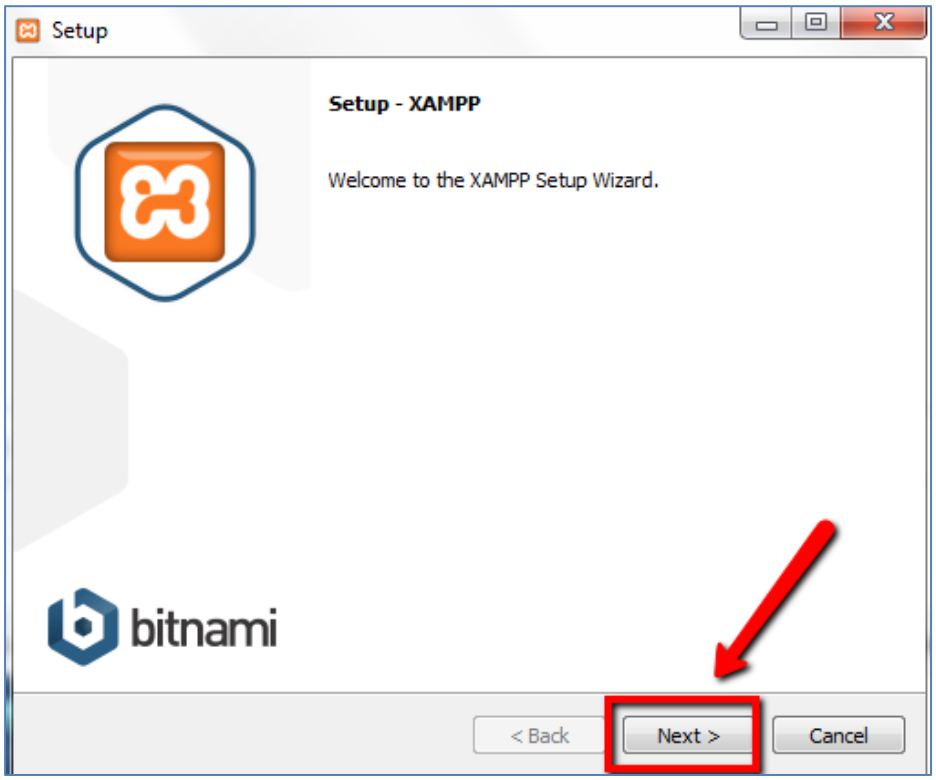
Μεταβαίνουμε αρχικά στην ιστοσελίδα https://www.apachefriends.org



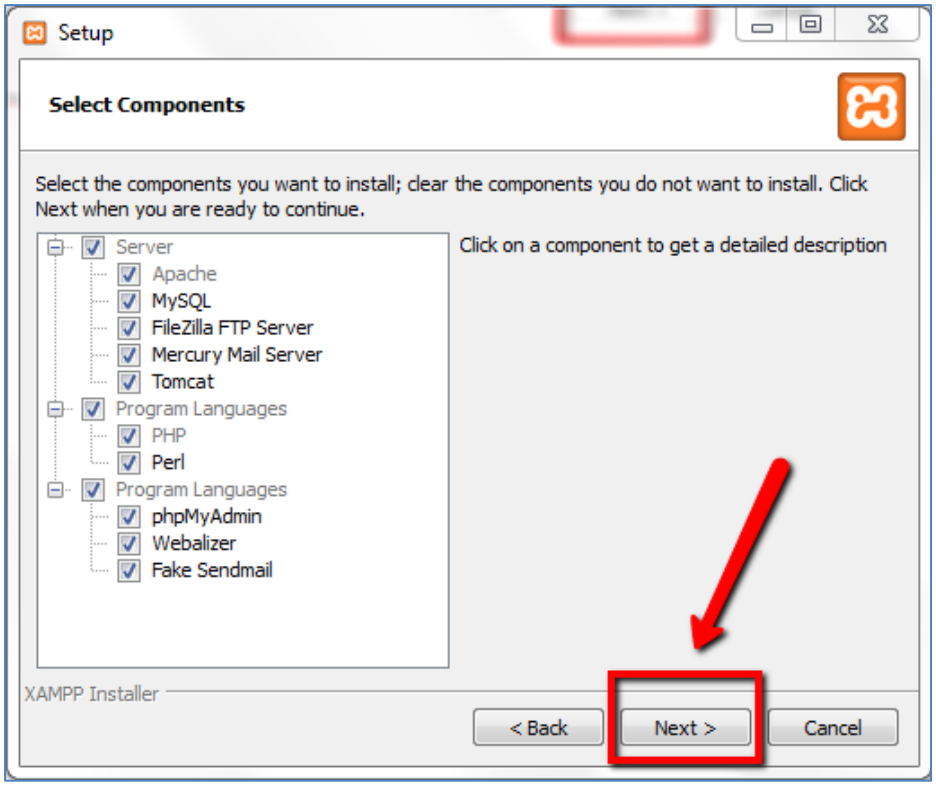
Κατεβάζουμε εκείνη την έκδοση του xampp σύμφωνα με το λειτουργικό μας σύστημα.



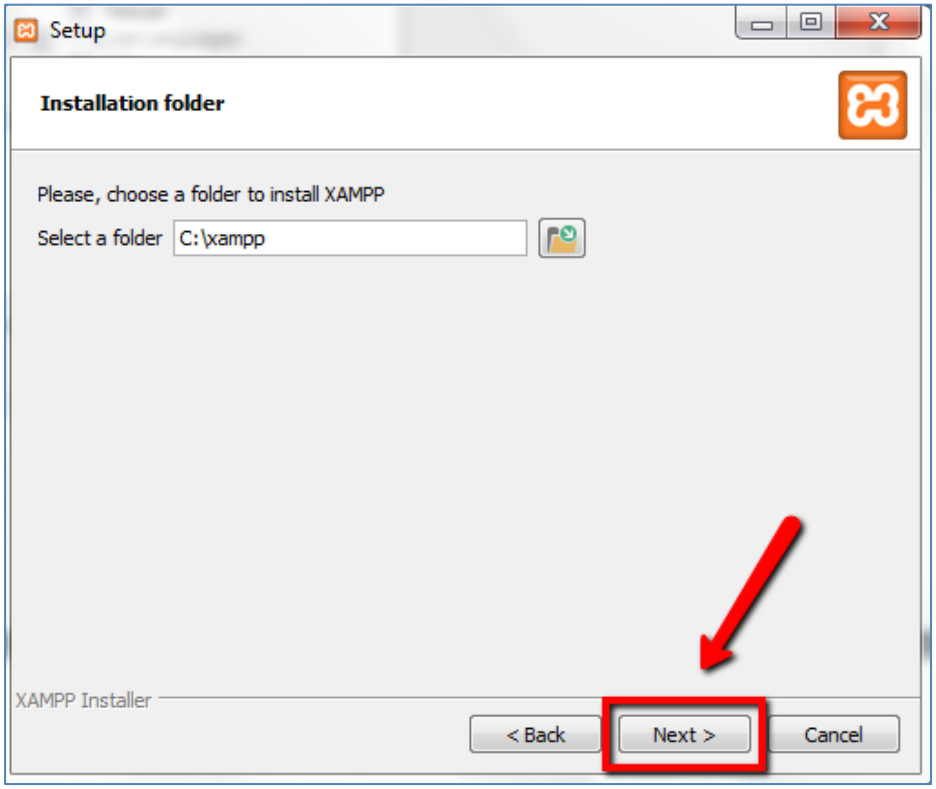
Αποθηκεύουμε και εγκαθιστούμε το αντίστοιχο αρχείο .exe. Κατά την εγκατάσταση είναι δυνατό να εμφανιστούν διάφορα μηνύματα, όπως να κλείσουμε το antivirus και άλλα. Πατάμε «Yes» στα αντίστοιχα πλαίσια διαλόγου για να συνεχιστεί η εγκατάσταση.



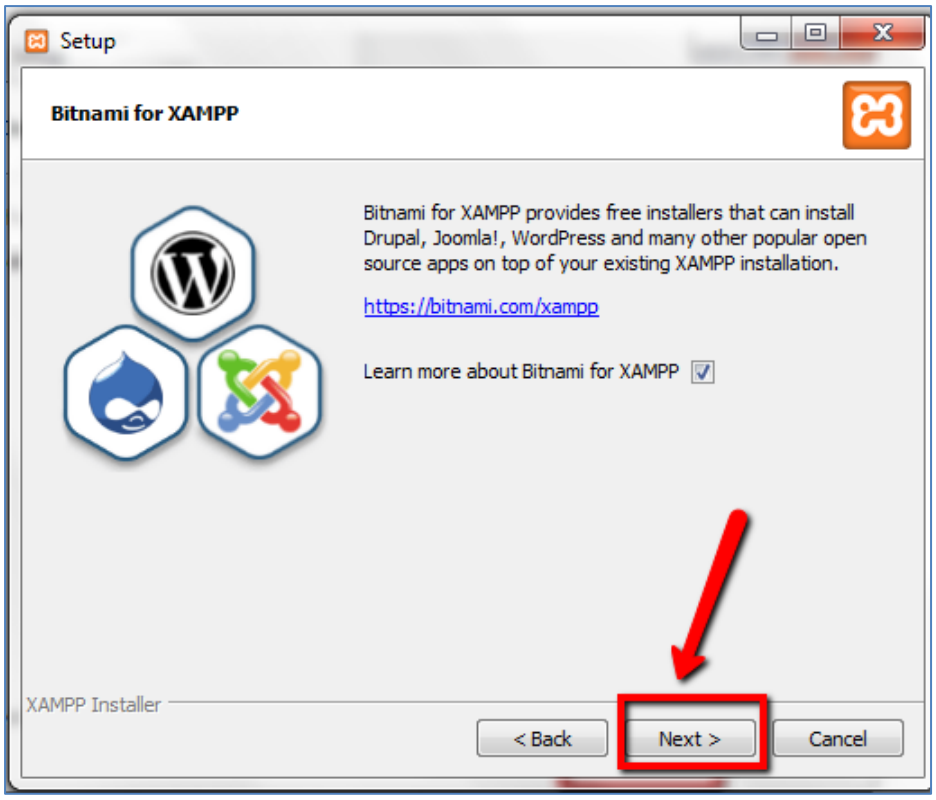
Παραμετροποιούμε την εγκατάσταση σύμφωνα με τις ανάγκες μας.



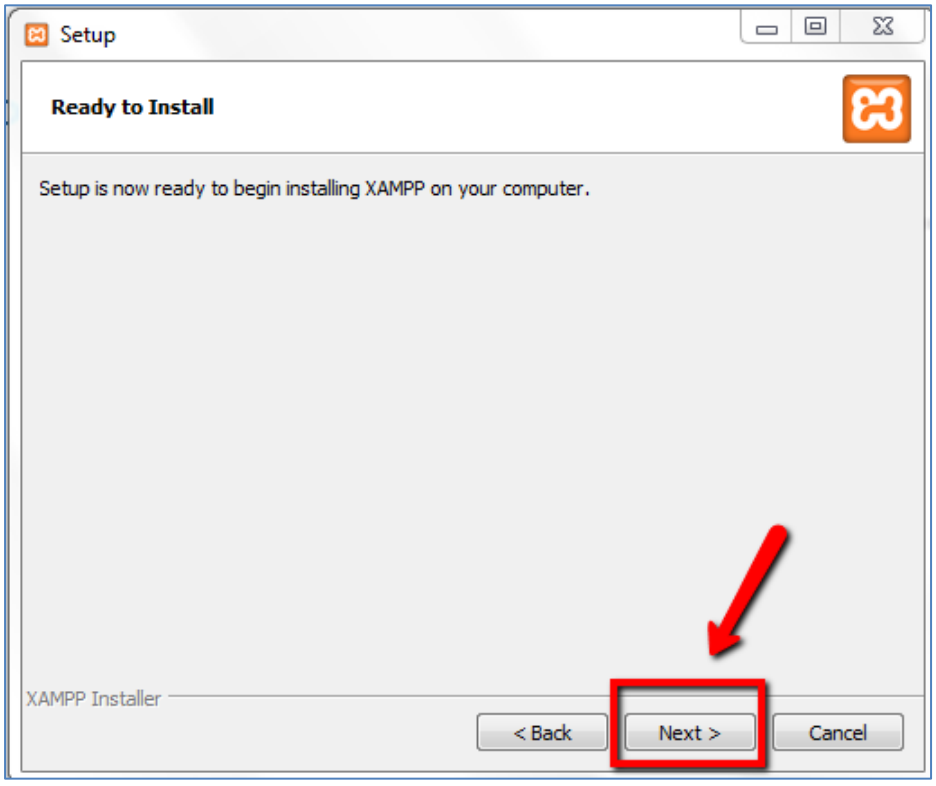
Το πρόγραμμά μας θα εγκατασταθεί στον φάκελο c:\xampp. Η τοποθεσία εγκατάστασης μπορεί να αλλάξει αν το επιθυμούμε.

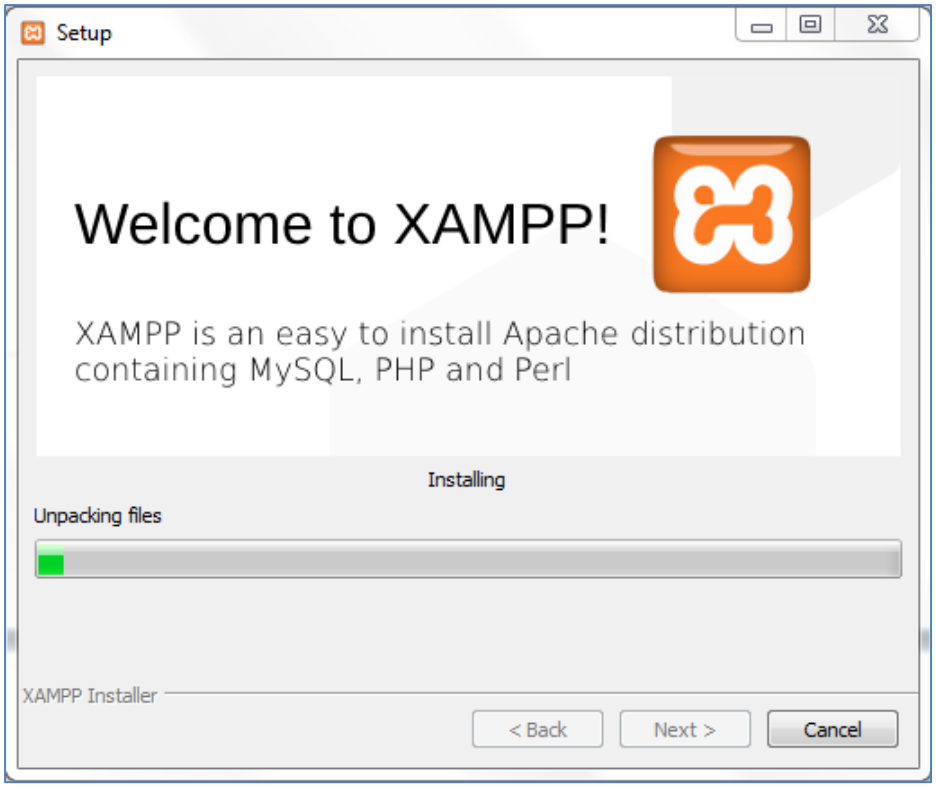


Η εγκατάσταση είναι απλή και τετριμμένη. Ακολουθώντας τις οδηγίες, όπως αυτές εμφανίζονται στην οθόνη μας και κάνοντας κλικ στο εικονίδιο Next κάθε φορά, μεταβαίνουμε από το ένα στάδιο της εγκατάστασης στο άλλο.

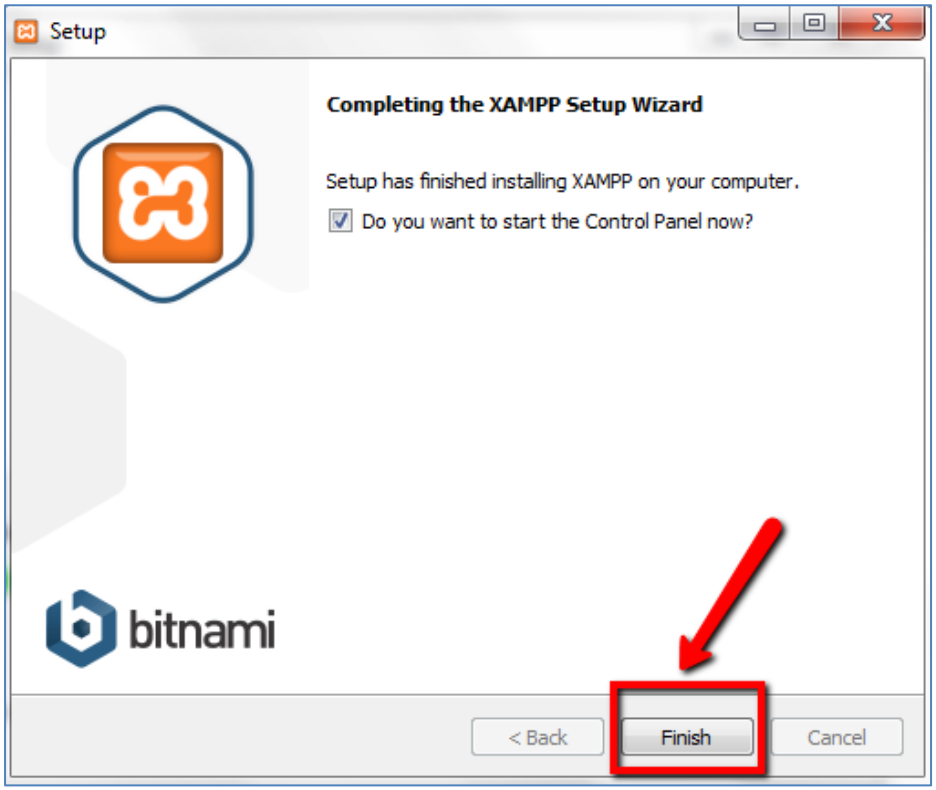


Συνεχίζεται η εγκατάσταση του xampp, όπως παρατηρείτε στις επόμενες οθόνες σας.



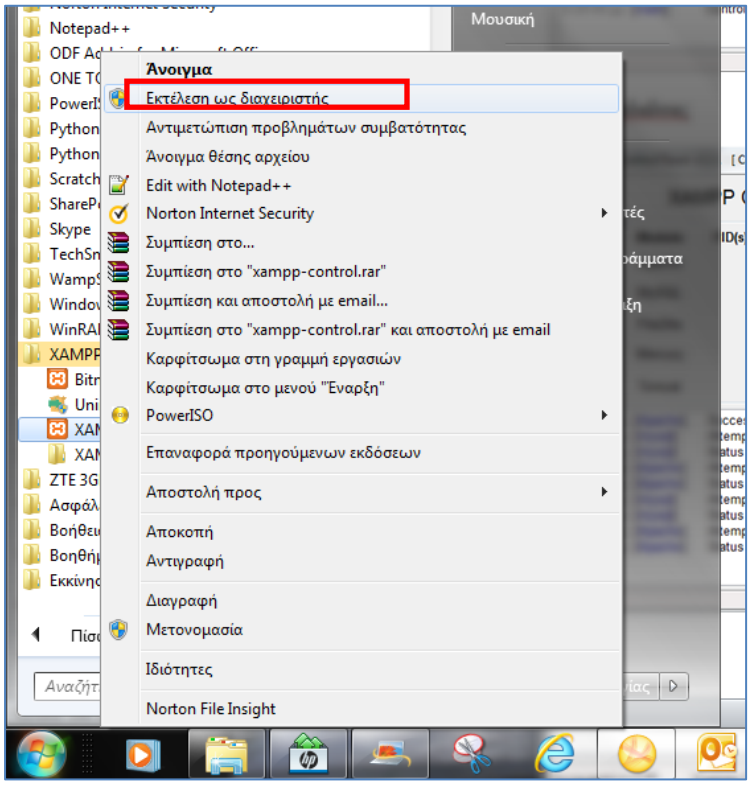


Η εγκατάσταση έχει ολοκληρωθεί. Κάνουμε κλικ στο εικονίδιο «Finish».



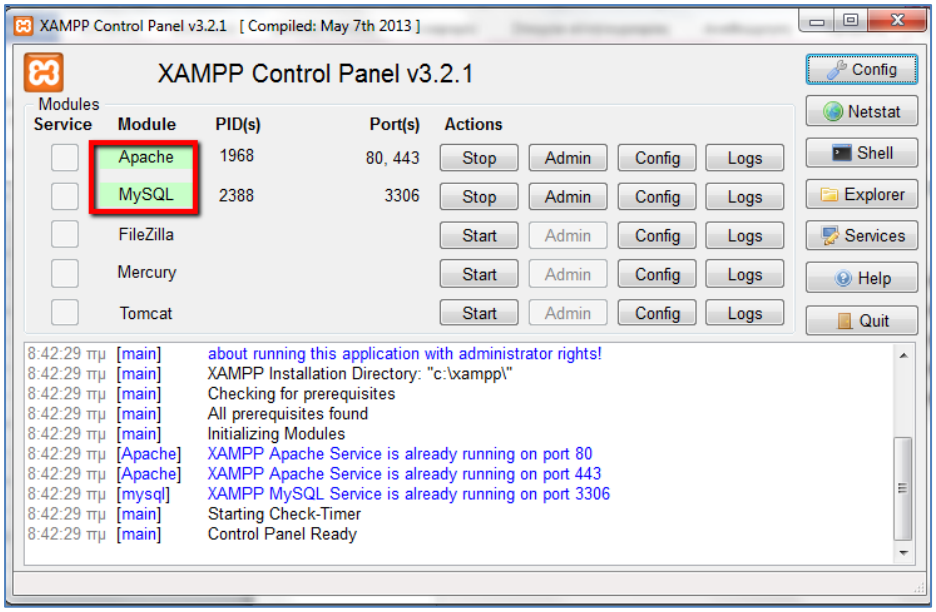
Επιλέγοντας προηγουμένως την επιλογή «Do you want to start the Control Panel now?" με την ολοκλήρωση της εγκατάστασης ξεκινάει το πρόγραμμα Control Panel για να ξεκινήσουμε/σταματήσουμε τους διακομιστές ή να εγκαταστήσουμε/

απεγκαταστήσουμε υπηρεσίες). Πατάμε start στα module του Apache και της MySql. Αν υπάρχει πρόβλημα στην ενεργοποίηση των υπηρεσιών δοκιμάστε να το εκετέσετε το πρόγραμμα Control Panel ως διαχειριστές. Στην περίπτωση αυτή, μπορείτε να βρείτε το πρόγραμμα XAMPP Control Panel από το εικονίδιο της έναρξης και να κάνετε δεξί κλικ σε αυτό επιλέγοντας την επιλογή «Εκτέλεση ως διαχειριστής».

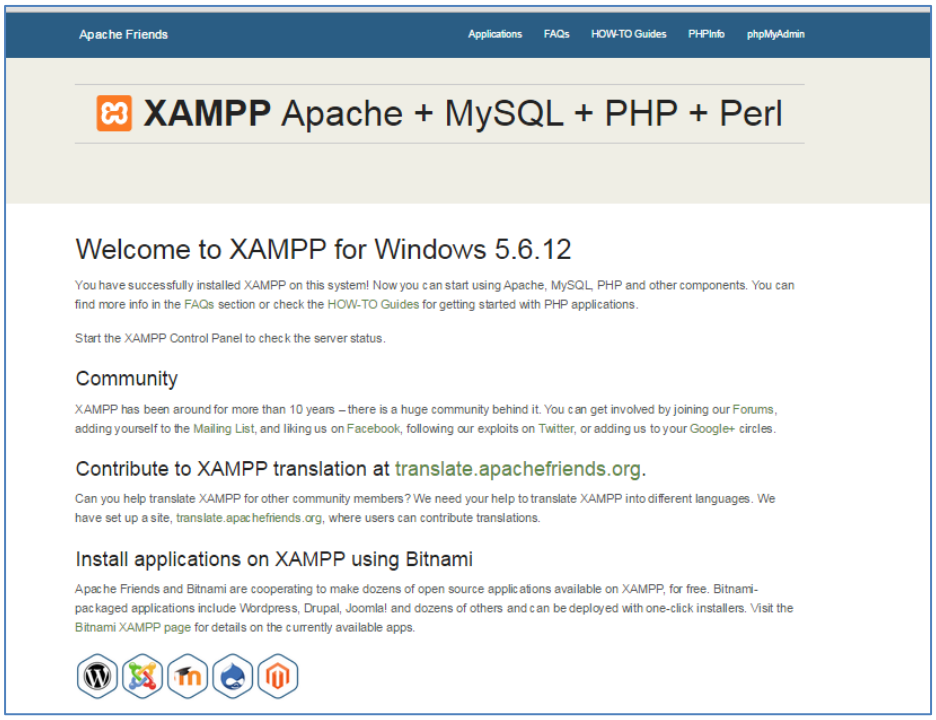


Η οθόνη του XAMPP Control Panel εμφανίζεται στην παρακάτω εικόνα. Μην ξεχνάτε να τρέχετε κάθε φορά το XAMPP Control Panel και να ενεργοποιείτε τα module του Apache και της MySql πριν ξεκινήσετε να εκτελείτε τις PHP εφαρμογές σας.

Όπως παρατηρείτε στην παρακάτω εικόνα τα module Apache και MySQL έχουν αποκτήσει πράσινο χρώμα. Αν δεν συμβαίνει αυτό προσπαθήστε να εξετάσετε αν είναι κατειλημμένη η αντίστοιχη θύρα από κάποιο άλλο πρόγραμμα, όπως είναι το skype που χρησιμοποιεί και αυτό την θύρα (port) 80. Στην περίπτωση αυτή θα πρέπει να κλείσουμε όποιο άλλο πρόγραμμα χρησιμοποιεί την ίδια θύρα ή να αλλάξουμε την θύρα στην οποία "τρέχει" ο Apache.

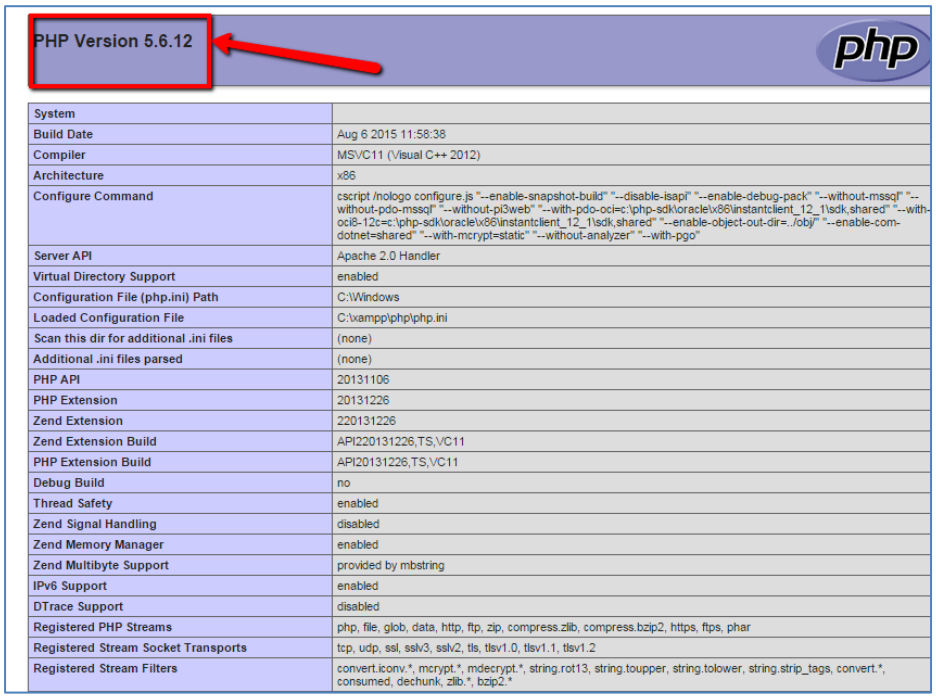


Για να βεβαιωθούμε ότι όλα πήγαν καλά, πληκτρολογείστε στο πρόγραμμα περιήγησής σας την διεύθυνση http://localhost ή http://127.0.0.1 . Θα πρέπει να δείτε την παρακάτω εικόνα στην οθόνη σας.



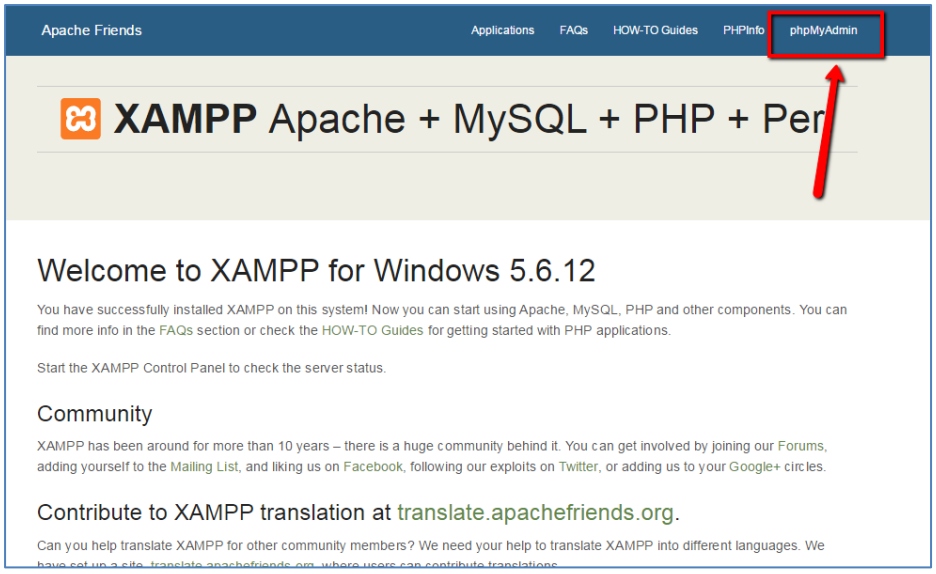
Μεταβείτε στην καρτέλα PHPInfo για να δείτε την έκδοση PHP που έχετε εγκαταστήσει.

Η έκδοση PHP είναι η 5.6.12.

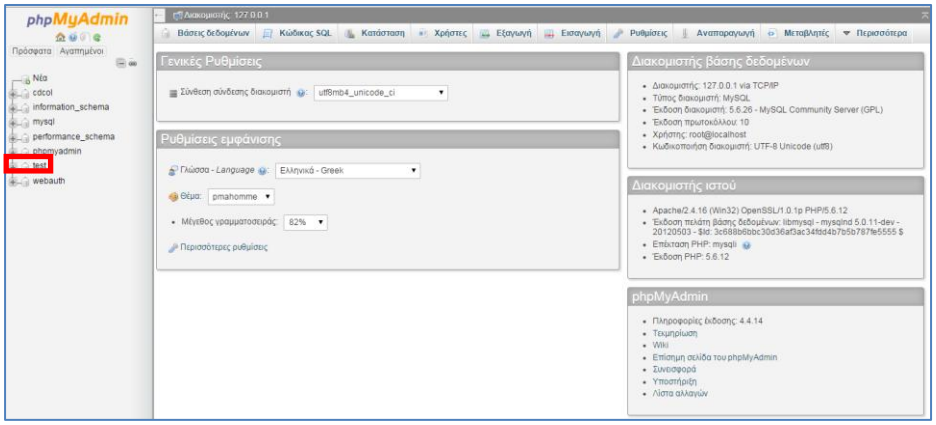


System	
Build Date	Aug 6 2015 11:58:38
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,TS,VC11
PHP Extension Build	API20131226,TS,VC11
Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, https, ftps, phar
Registered Stream Socket Transports	tcp, udp, ssl, sslv3, sslv2, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	convert.iconv.*, mcrypt.*, mdecrypt.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, zlib.*, bzip2.*

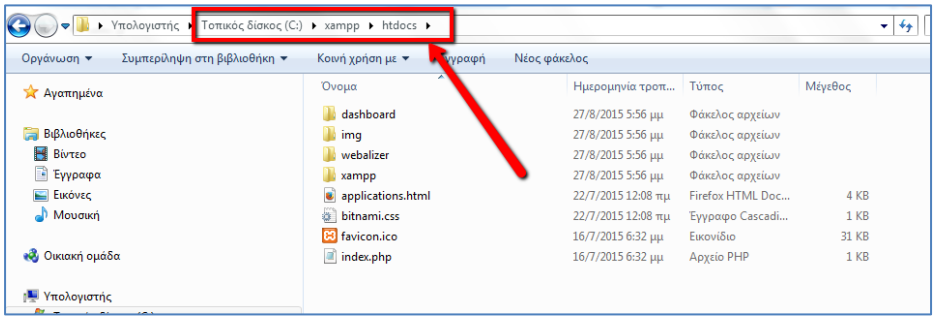
Μεταβείτε στην καρτέλα phpMyAdmin για να διαχειριστείτε την βάση δεδομένων σας.



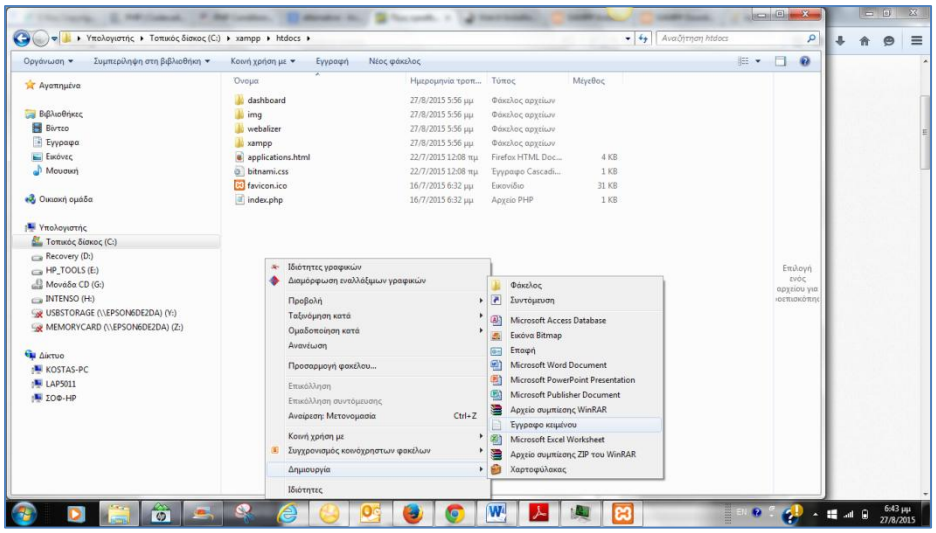
Μέσα από το περιβάλλον της phpmyadmin μπορείτε να διαχειριστείτε την βάση δεδομένων σας. Ορισμένα ενδεικτικά παραδείγματα διαχείρισης της MySQL παρουσιάζονται σε επόμενη ενότητα. Παρατηρείστε ότι με την εγκατάσταση του xampp δημιουργήθηκε μία βάση δεδομένων με το όνομα test. Η βάση αυτή δεδομένων θα χρησιμοποιηθεί στη συνέχεια για τα πρώτα προγράμματα που θα φτιάξουμε πριν δημιουργήσουμε την δική μας βάση δεδομένων.



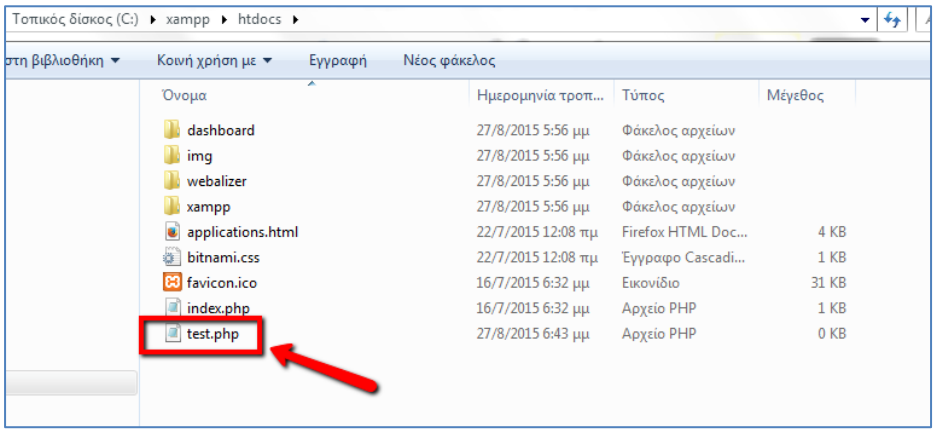
Μεταβείτε στον φάκελο c:\xampp\htdocs για να δημιουργήσετε το πρώτο πρόγραμμα σε php.



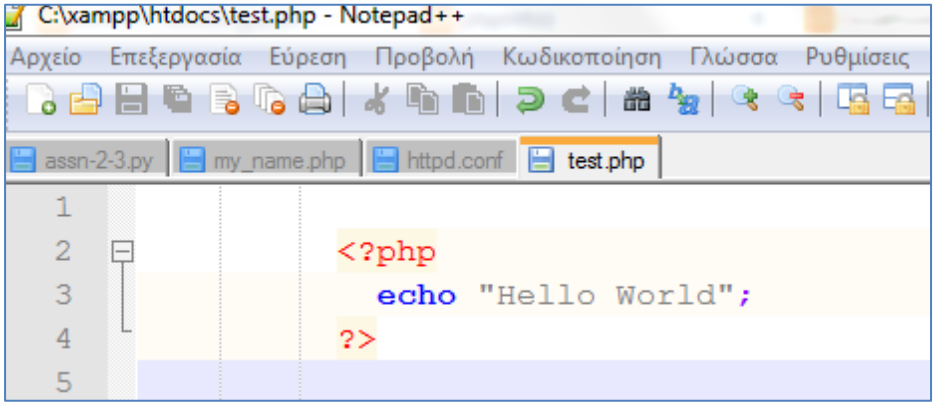
Στον φάκελο αυτό δημιουργούμε ένα αρχείο κειμένου και το ονομάζουμε test.php όπως φαίνεται στην παρακάτω οθόνη. Καλό είναι να δημιουργήσουμε έναν επιπλέον φάκελο μέσα στον φάκελο htdocs και να τοποθετήσουμε εκεί όλα τα αρχεία μας.



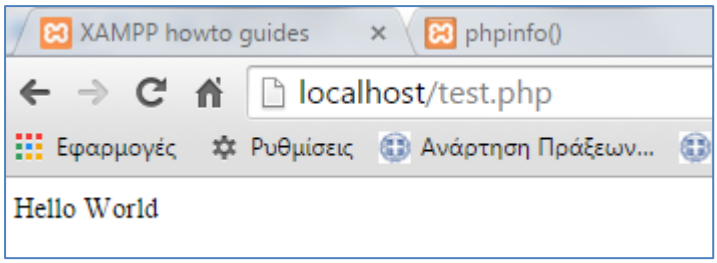
Το αρχείο test.php έχει δημιουργηθεί.



Με έναν editor για παράδειγμα με το Notepad++ πληκτρολογείστε τον ακόλουθο κώδικα, όπως φαίνεται στην παρακάτω οθόνη.



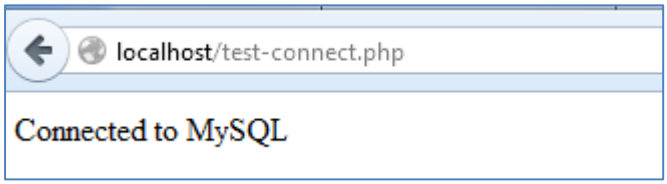
Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση http://localhost/test.php για να εκτελέσετε το πρόγραμμα test.php. Τι παρατηρείτε; Εμφανίστηκε το μήνυμα «Hello World»;



Για να σιγουρευτούμε ότι η PHP συνδέεται με τον διακομιστή της MySQL δημιουργήστε ένα νέο αρχείο με το όνομα test-connect.php στον φάκελο c:\xampp\htdocs. Ο κώδικας που θα περιέχει αυτό το αρχείο παρουσιάζεται στην συνέχεια. Χρησιμοποιείται η συνάρτηση mysql_connect για την σύνδεση με τον διακομιστή της MySQL.

```
<?php

mysql_connect("localhost", "root", "") or die(mysql_error());

echo "Connected to MySQL<br />";

?>
```

Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση http://localhost/test-connect.php για να εκτελέσετε το πρόγραμμα test-connect.php. Τι παρατηρείτε; Εμφανίστηκε το μήνυμα " Connected to MySQL";



Η συνάρτηση **mysql_connect** δέχεται τρεις παραμέτρους: το όνομα του διακομιστή (server), το όνομα του χρήστη (username) και τον κωδικό του χρήστη (password). Στο παράδειγμά μας, οι τρείς αυτοί παράμετροι έχουν τις ακόλουθες τιμές:

- **server** = " localhost" (Οι διακομιστές PHP και MySQL είναι εγκατεστημένοι στον ίδιο τοπικό υπολογιστή)
- **username** = "root"
- **password** = " " (δεν έχουμε βάλει κωδικό πρόσβασης για τον root)

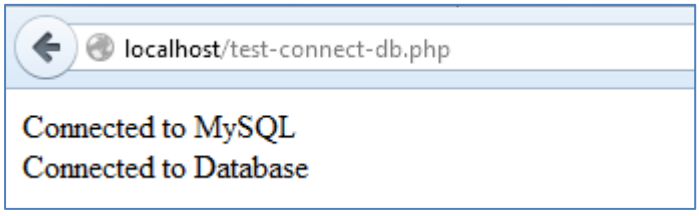
Η συνάρτηση die(mysql_error()) εμφανίζει ένα μήνυμα λάθους στο πρόγραμμα περιήγησής σας στην περίπτωση που παρουσιαστεί πρόβλημα κατά την προσπάθεια σύνδεσης σας με τον διακομιστή της MySQL.

Διαχείριση βάσεων δεδομένων μέσω PHP

Μετά τη δημιουργία σύνδεσης με τον διακομιστή της MySQL ας δημιουργήσουμε κάποια απλά προγράμματα PHP τα οποία θα «συνομιλούν» με την MySQL. Ας ξεκινήσουμε από ένα πρόγραμμα με το οποίο θα επιλέγουμε ποια βάση δεδομένων θα χρησιμοποιούμε με τη σύνδεση αυτή της MySQL. Για τον σκοπό αυτό θα χρησιμοποιήσουμε την συνάρτηση **mysql_select_db**, όπως φαίνεται στον παρακάτω κώδικα. Θα προσπαθήσουμε να συνδεθούμε με την βάση δεδομένων **test** που προϋπάρχει με την εγκατάσταση του xampp. Δημιουργήστε ένα νέο αρχείο με το όνομα test-connect-db.php στον φάκελο c:\xampp\htdocs και πληκτρολογείστε τον παρακάτω κώδικα.

```
<?php

mysql_connect("localhost", "root", "") or die(mysql_error());

echo "Connected to MySQL<br />";

mysql_select_db("test") or die(mysql_error());

echo "Connected to Database";

?>
```

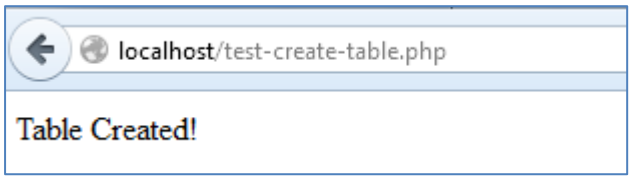
Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση http://localhost/test-connect-db.php για να εκτελέσετε το πρόγραμμα test-connect-db.php. Τι παρατηρείτε; Εμφανίστηκαν τα μηνύματα σύνδεσης με την MySQL και την βάση δεδομένων «test»;



Ας δημιουργήσουμε έναν πίνακα με το όνομα example στην βάση δεδομένων test. Θα χρησιμοποιήσουμε την συνάρτηση **mysql_query** για να πούμε στην MySQL ότι θέλουμε να κάνει κάτι για εμάς. Στέλνουμε δηλαδή ένα ερώτημα (query). Αυτό το κάτι είναι να δημιουργήσει έναν πίνακα (με την εντολή **CREATE TABLE)** με το όνομα «example». Ο πίνακας αυτός θα περιέχει μόνο μία στήλη με το όνομα message. Στον πίνακα μπορούμε να αποθηκεύσουμε «φράσεις». Αν μετρήσουμε τους χαρακτήρες σε κάθε «φράση» που θα καταχωρούμε δεν θα πρέπει να ξεπερνάμε τους 30. Δημιουργήστε ένα νέο αρχείο με το όνομα test-create-table.php στον φάκελο c:\xampp\htdocs και πληκτρολογείστε τον παρακάτω κώδικα.

```
<?php

// Σύνδεση με την MySQL

mysql_connect("localhost", "root", "") or die(mysql_error());

mysql_select_db("test") or die(mysql_error());

// Δημιουργία πίνακα στην επιλεγμένη βάση “test”

mysql_query("CREATE TABLE example(message VARCHAR(30))") or

die(mysql_error());

echo "Table Created!";

?>
```

Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση http://localhost/test-create-table.php για να εκτελέσετε το πρόγραμμα test-create-table.php. Τι παρατηρείτε; Εμφανίστηκε το μήνυμα δημιουργίας πίνακα;

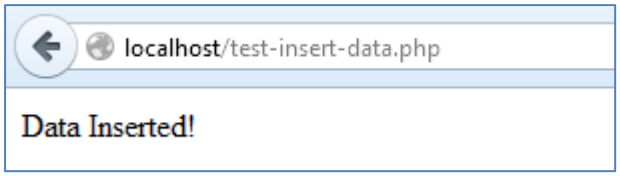


Στη συνέχεια, ας προσπαθήσουμε να εισάγουμε κάποιες πληροφορίες/φράσεις στον πίνακά μας και συγκεκριμένα την φράση «I like solving problems». Χρησιμοποιούμε την συνάρτηση mysql_query. Η εντολή **INSERT INTO** σημαίνει ότι θέλουμε να καταχωρήσουμε δεδομένα σε έναν πίνακα και συγκεκριμένα στον πίνακα «example». Δημιουργήστε ένα νέο αρχείο με το όνομα test-insert-data.php στον φάκελο c:\xampp\htdocs και πληκτρολογείστε τον παρακάτω κώδικα.

```
<?php

// Σύνδεση με την MySQL

mysql_connect("localhost", "root", "") or die(mysql_error());

mysql_select_db("test") or die(mysql_error());

// Εισαγωγή μιας εγγραφής στον πίνακα “example”

mysql_query("INSERT INTO example

(message) VALUES('I like solving problems') ")
```

```
or die(mysql_error());

echo "Data Inserted!";

?>
```

Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση http://localhost/test-insert-data.php για να εκτελέσετε το πρόγραμμα test-insert-data.php. Τι παρατηρείτε; Εμφανίστηκε το μήνυμα εισαγωγής δεδομένων;



Ας εμφανίσουμε στην οθόνη μας την φράση «I like solving problems» που μόλις καταχωρήσαμε στον πίνακά μας. Χρησιμοποιούμε και εδώ την συνάρτηση mysql_query. Με την εντολή **SELECT** ζητάμε από την MySQL να μας στείλει τα περιεχόμενα του πίνακα example (στην συγκεκριμένη περίπτωση για λόγους απλότητας υπάρχει καταχωρημένη μόνο μία φράση). Για να μπορέσουμε τα περιεχόμενα αυτά να τα χρησιμοποιήσουμε (ή να τα εμφανίσουμε) αργότερα μέσα στο PHP πρόγραμμά μας, τα αποθηκεύουμε στην μεταβλητή $result.

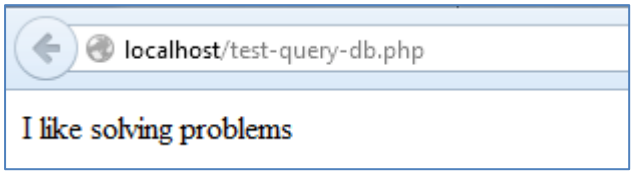
Η συνάρτηση **mysql_fetch_array** μας επιστρέφει το πρώτο αποτέλεσμα από το ερώτημα SELECT με την μορφή πίνακα. Με την βοήθεια της PHP συνάρτησης mysql_fetch_array μπορούμε να εμφανίσουμε το περιεχόμενο του πίνακα example στην επιθυμητή μορφή. Σε περίπτωση, που είχαμε περισσότερα που ένα αποτελέσματα έπρεπε να τοποθετήσουμε την συνάρτηση mysql_fetch_array μέσα σε μία δομή επανάληψης για να μπορέσουμε να τα εμφανίσουμε όλα**.**

Δημιουργήστε ένα νέο αρχείο με το όνομα test-quey-db.php στον φάκελο c:\xampp\htdocs και πληκτρολογείστε τον παρακάτω κώδικα.

```
<?php

// Σύνδεση με την MySQL

mysql_connect("localhost", "root", "") or die(mysql_error());

mysql_select_db("test") or die(mysql_error());

// Ανάκτηση των δεδομένων από τον πίνακα “example”

$result = mysql_query("SELECT * FROM example")

or die(mysql_error());
```

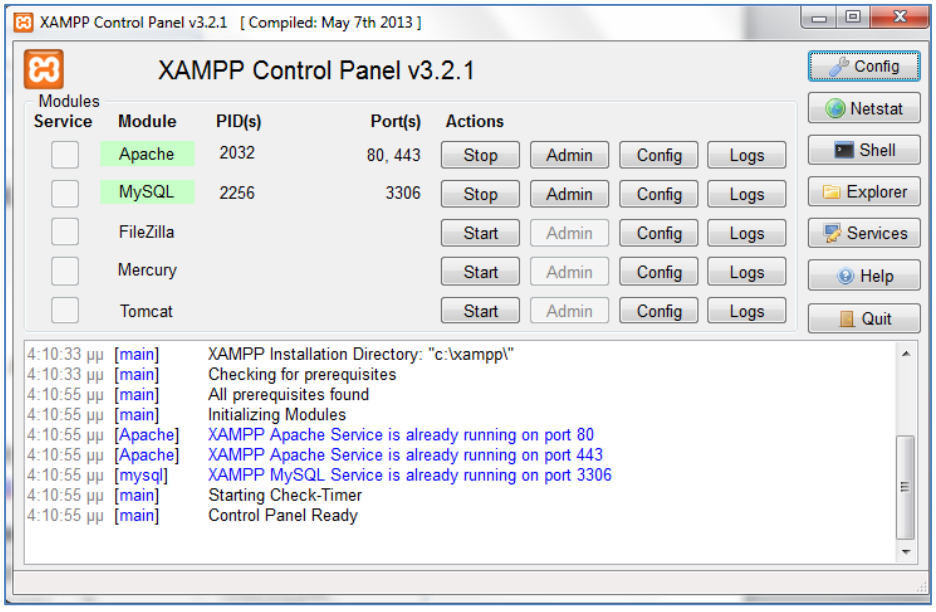
```
// Αποθήκευση της πρώτης εγγραφής του πίνακα “example” στην μεταβλητή $row
//που έχει την μορφή πίνακα

$row = mysql_fetch_array( $result );

// Εμφάνιση της τιμής της μεταβλητής $row

echo $row['message'];

?>
```

Στο πρόγραμμα περιήγησή σας πληκτρολογήστε την διεύθυνση http://localhost/test-query-db.php για να εκτελέσετε το πρόγραμμα test-query-db.php. Τι παρατηρείτε; Εμφανίστηκε το περιεχόμενο του πίνακα «example»;

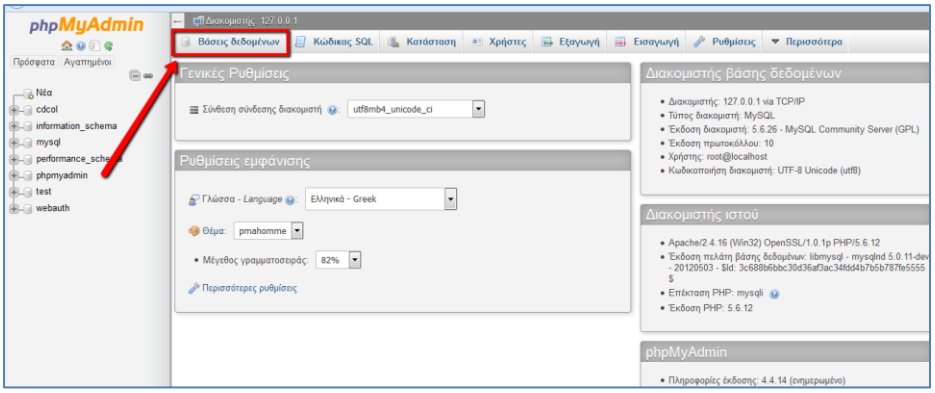


Δημιουργία βάσης δεδομένων με phpmyadmin

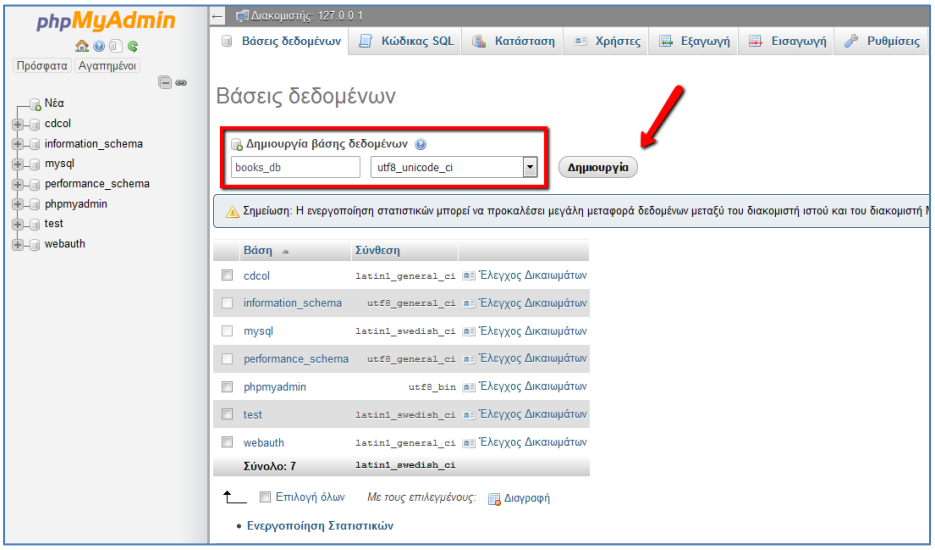
Εκκινήστε το Control Panel του XAMPP και ενεργοποιήστε τις υπηρεσίες Apache και MySQL αν χρειάζεται.



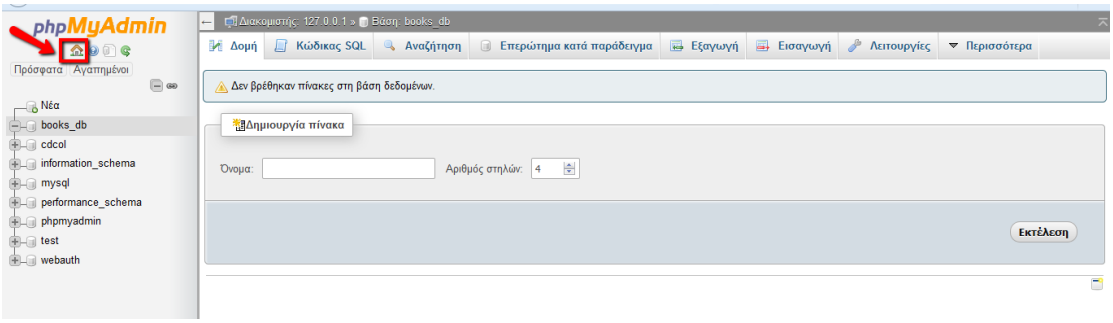
Πληκτρολογείστε http://localhost/phpmyadmin και κάντε κλικ στην καρτέλα **Βάσεις δεδομένων.**



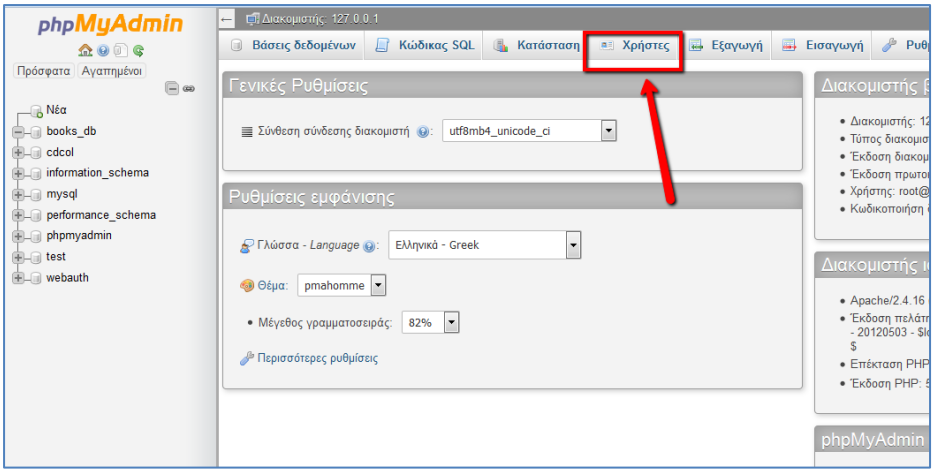
Πληκτρολογήστε το όνομα της Βάσης στο σχετικό πεδίο και από το μενού **Σύνδεση Σύνθεσης Διακομιστή** επιλέξτε **utf8_unicode_ci**. Πατήστε «**Δημιουργία**».



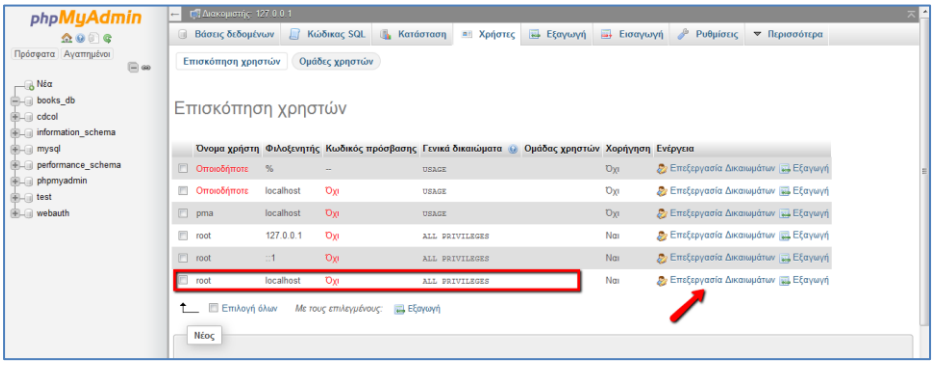
Δημιουργήθηκε η βάση μας. Στο σημείο αυτό αν θέλετε μπορείτε να δημιουργήσετε πίνακα. Ας μεταβούμε στην αρχική καρτέλα κάνοντας κλικ στο αντίστοιχο εικονίδιο, όπως φαίνεται στην παρακάτω εικόνα.



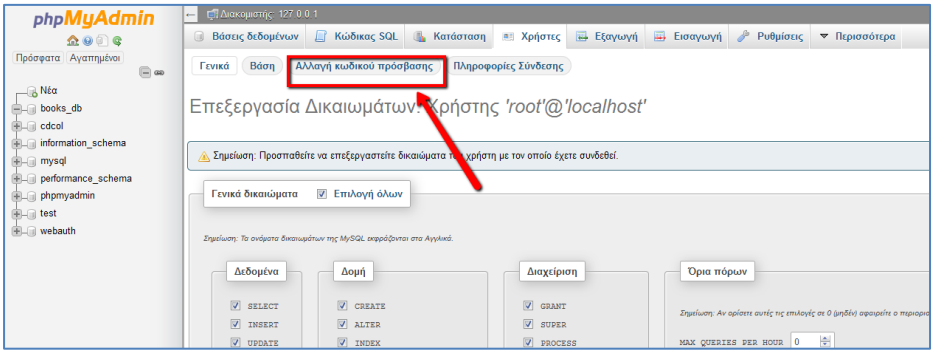
Ας αλλάξουμε το password του «**root**» της MySQL για λόγους ασφαλείας, όπου η default ρύθμιση είναι χωρίς κωδικό. Μεταβείτε στην καρτέλα **Χρήστες**.



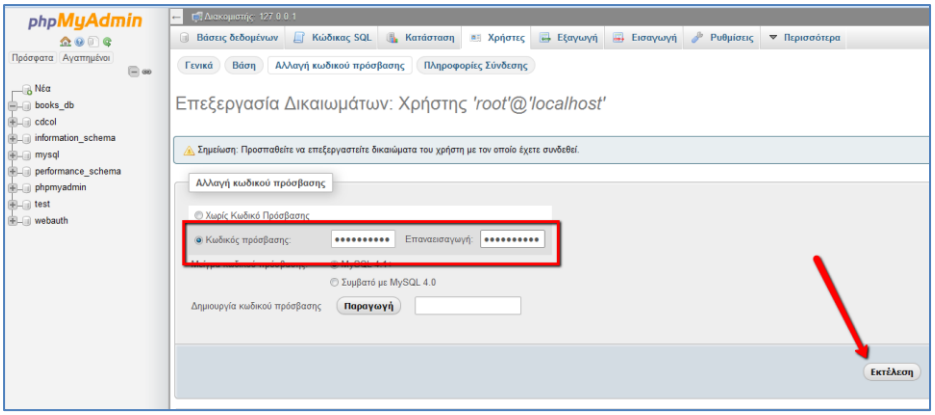
Επιλέγουμε τον χρήστη "root" όπως φαίνεται στην παρακάτω εικόνα και κάνουμε κλικ στην επιλογή στο «**Επεξεργασία Δικαιωμάτων**».



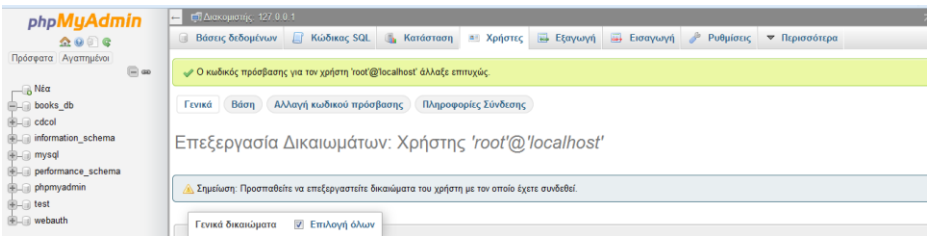
Στη συνέχεια επιλέγουμε "**Αλλαγή κωδικού πρόσβασης**".



Ορίζουμε τον επιθυμητό κωδικό, για παράδειγμα «mypassword» για λόγους απλότητας και κάνουμε κλικ στο «**Εκτέλεση**».



Η αλλαγή του κωδικού έγινε με επιτυχία.



Τέλος, θα πρέπει να ορίσουμε το νέο κωδικό στο αρχείο config.inc.php (που βρίσκεται στον φάκελο C:\xampp\phpMyAdmin) στη παρακάτω σειρά:

$cfg['Servers'][$i]['password'] = 'mypassword';

Αν θελήσουμε να εκτελέσουμε, όπως και στην αρχή της προηγούμενης ενότητας, ένα πρόγραμμα PHP σύνδεσης με τον διακομιστή της MySQL θα πρέπει να πληκτρολογήσουμε τον παρακάτω κώδικα δίνοντας ως username το «root» και ως password το «mypassword».

```
<?php

mysql_connect("localhost", "root", "mypassword") or die(mysql_error());

echo "Connected to MySQL<br />";

?>
```

Διαδικτυακές Πηγές

- Εγκατάσταση Apache, php, MySQL, phpMyAdmin: ΚΕ.ΠΛΗ.ΝΕΤ Δυτικής Θεσσαλονίκης
  http://dide-v.thess.sch.gr/plinet/install-apache-php-mysql-phpmyadmin

- Εγκατάσταση xampp: Πανελλήνιο Σχολικό Δίκτυο
  http://www.sch.gr/3187-server#