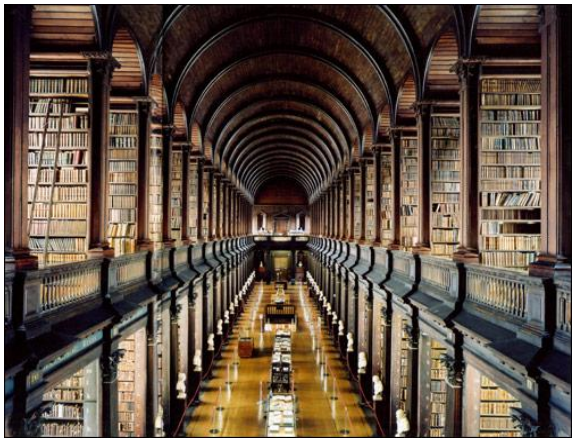
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΑΤΡΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΣΤΗΝ ΔΙΟΙΚΗΣΗ ΚΑΙ ΟΙΚΟΝΟΜΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ



# Σχεδιασμός και Υλοποίηση ενός διαδικτυακού online ηλεκτρονικού καταστήματος

# Design and Implementation of an online web shop

Σπουδαστές: Αλέξανδρος Γκαντής

Ειρήνη Μπούρα

Εισηγητής: Βλάχος Μάριος

ΑΜΑΛΙΑΔΑ 2011

## Περιεχόμενα

## Περίληψη

Η παρούσα πτυχιακή αποσκοπεί στην ανάλυση, τον σχεδιασμό και την ανάπτυξη μιας εφαρμογής διαχείρισης ψηφιακής βιβλιοθήκης.

Η εφαρμογή θα υποστηρίζει τόσο τις βασικές εργασίες που θα εκτελεί ο γενικός διαχειριστής, όσο και τις υπηρεσίες των μελών-χρηστών της βιβλιοθήκης.

Στην βάση δεδομένων θα καταγράφονται το σύνολο των βιβλίων της βιβλιοθήκης, με αναλυτικές πληροφορίες για καθένα από αυτά.

Θα παρέχεται δυνατότητα αναζήτησης στην βάση δεδομένων καθώς επίσης και διαχείρισης των βιβλίων (εισαγωγή, τροποποίηση πληροφοριών των βιβλίων, διαγραφή των βιβλίων).

Στη βάση δεδομένων θα καταχωρούνται και τα προσωπικά στοιχειά των μελών-χρηστών που ζητούνται κατά την εγγραφή τους καθώς και τα στοιχειά πρόσβασης που είναι απαραίτητα για την είσοδο του στην εφαρμογή.

Κάθε μέλος θα έχει τον δικό του λογαριασμό και δυνατότητα πρόσβασης στην εφαρμογή. Θα μπορεί να :

1. κάνει αναζήτηση βιβλίων,
2. να κατεβάσει τα βιβλία που τον ενδιαφέρουν,
3. καθώς και να δει την καρτέλα με τα βιβλία που έχει κατεβάσει.

Ανάλογα με την ιδιότητα του κάθε χρήστη (απλός χρήστης, εκδοτικός οίκος, ελεγκτής περιεχομένου βιβλίων ,γενικός διαχειριστής) θα έχει και τα αντίστοιχα δικαιώματα πρόσβασης στις λειτουργίες της εφαρμογής.

## Abstract

The present study aims to analyze, design and develop an application for the management of a digital library.

The application will support the basic tasks that the general administrator will execute, as well as the services of the member-users of the library.

All the books of the library will be recorded on the database, with detailed information for each one. The capability of searching to the database as well to the management of the books (introduction, amendment of the information, deletion of the books), will be provided.

On the database will also be recorded the personal information of the member-users that is necessary for their registration, as well as the access codes that are necessary in order to access the application. Each member will maintain its account and will have the capability of accessing the application. It will be possible for the members to:

1. Search for books.
2. Download interesting books.
3. View the relevant card of the downloaded books.

Depending on the status of each user (simple user, publishing house, checker of book content, general administrator), each one will have the appropriate authority to access the application.

# 1 Γενικά-Εισαγωγικά

Στην αρχή θα πρέπει να αναφέρουμε ποίος είναι ο λόγος δημιουργίας της παρούσας εφαρμογής.

Η μνήμη των ανθρώπων είναι υψίστης σημασίας για την διατήρηση της πνευματικής προσωπικότητας, για το γεφύρωμα του παρελθόντος με το παρόν και για τη διαμόρφωση του μέλλοντος. Η καταγεγραμμένη κληρονομιά αποτελεί το κύριο μέρος αυτής της μνήμης και αντανακλά στην ποικιλία των ανθρώπων, των γλωσσών και των πολιτισμών.

Η μνήμη αυτή όμως είναι εύθραυστη. Πολύτιμα και μοναδικά τεκμήρια, καμιά φορά και ολόκληρες συλλογές βιβλιοθηκών και αρχείων, μπορούν να χαθούν πολύ περισσότερο στην σημερινή εποχή της δημιουργίας ψηφιακών βιβλιοθηκών. Τα ψηφιακά τεκμήρια όπως και τα συμβατικά έχουν ανάγκη διατήρησης και οι βιβλιοθήκες, αρχεία κλπ έχουν χρέος να φροντίζουν για τη σωστή διατήρηση τους.

## 1.1 Αναλυτική περιγραφή του προβλήματος

Οι βιβλιοθήκες συνεπικουρούμενες από την ανάπτυξη της τεχνολογίας σήμερα, αποτελούν σύνθετους οργανισμούς με πολλαπλούς στόχους που δεν έχουν προηγούμενο. Αυτό δεν συνέβαινε ανέκαθεν. Επί αιώνες δημιουργούσαν έντυπες συλλογές και έντυπες βάσεις δεδομένων, τους γνωστούς δελτίο - καταλόγους.

Όταν οι τεχνολογίες των οπτικοακουστικών αναπτύχθηκαν και διαδόθηκαν ευρέως, οι βιβλιοθήκες απέκτησαν μεγάλες συλλογές για εκπαιδευτικούς και ψυχαγωγικούς σκοπούς. Ακόμη και στην εκπαίδευση των χρηστών, ακόμη και στην απομακρυσμένη εκπαίδευση έπαιξαν πρωτεύοντα ρόλο τα οπτικοακουστικά μέσα.

Μέχρι αυτή τη στιγμή η οργάνωση και η εξέλιξη των βιβλιοθηκών κινείτο κανονικά σε κάθε στάδιο, χωρίς χάσματα, ραγδαίες αλλαγές ή ανατροπές στη σκέψη και τη πράξη. Οι βιβλιοθήκες δρούσαν με επίκεντρο τη συλλογή τους και αποφάσιζαν την οργάνωσή τους με κριτήριο τη διαφύλαξη και διατήρηση της.

Αλλά κατά τη δεκαετία του ’80, η νέα τεχνολογία με την επονομασία “συστήματα βιβλιοθηκών” ήρθε στο προσκήνιο. Οι βιβλιοθήκες κατανόησαν ότι θα μπορούσαν να απλοποιήσουν και να επεκτείνουν τη δουλειά τους και άρχισαν να υιοθετούν τα συστήματα αυτά μαζικά. Ήταν η πρόκληση των μεγάλων προγραμμάτων αυτοματοποίησης των βιβλιοθηκών. Αυτή τη στιγμή η υλοποίηση της νέας τεχνολογίας έφερνε και επίδραση στην οργάνωση των βιβλιοθηκών. Πέραν των ανωτέρω, το διαδίκτυο, το ηλεκτρονικό ταχυδρομείο και ο παγκόσμιος ιστός όρμισαν στη ζωή μας.

Με την πληθώρα νέων πηγών, τοπικών και απομακρυσμένων, τις νέες δυνατότητες για παραλαβή υλικού, την απουσία ομοιομορφίας στα διάφορα interfaces των διαρκώς μεταβαλλόμενων μηχανών αναζήτησης και τις τεχνικές δεξιότητες που απαιτούντο, το προσωπικό των Βιβλιοθηκών και των κέντρων πληροφόρησης δυσκολεύτηκε να αφυπνισθεί, πράγμα που ενέτεινε η αίσθηση ότι είναι ευάλωτο και ανεπαρκές.

Η αβεβαιότητα που συνδυάζεται με την οργανωτική αλλαγή είναι προφανώς αναπόφευκτη, αφού ο εργαζόμενος αισθάνεται να διακυβεύεται ο επαγγελματισμός του, στις μεταβολές.

Πάντως, αυτό που έχει διαπιστωθεί από καιρού εις καιρόν είναι ότι η τεχνολογία αποτελεί μόνο το πρόσχημα της οργανωτικής αλλαγής.

Σήμερα, οι στόχοι και οι σκοποί των βιβλιοθηκών έχουν μάλλον εμπλουτιστεί σε σύγκριση με το παρελθόν, η ταυτότητα της βιβλιοθήκης έχει τροποποιηθεί, όχι όμως και ο σκοπός και η αποστολή της.

Οι βιβλιοθήκες, εκτός από τεχνολογικά εξακολουθούν να είναι κοινωνικά συστήματα, να επικοινωνούν με ανθρώπους, να υποστηρίζουν τις ανάγκες τους σε εκπαίδευση και ψυχαγωγία. Εξακολουθούν να προστατεύουν την ανθρώπινη ιστορία και σκέψη δια μέσου των αιώνων και να εκπαιδεύουν το κοινό για το πώς θα χρησιμοποιεί το υλικό που χρειάζεται. Το οποίο υλικό η βιβλιοθήκη έχει πριν από αυτούς αναζητήσει, εντοπίσει, αξιολογήσει, αποκτήσει, οργανώσει, διαμοιράσει.

Επομένως, η τεχνολογία δεν αποτελεί από μόνη της το θεμελιώδες χαρακτηριστικό της σύγχρονης βιβλιοθήκης, απλά εμπλουτίζει τις μορφές και τον όγκο των πηγών που η τελευταία προσφέρει.

Συνεκδοχικά, η τεχνολογία προκαλεί ραγδαίες οργανωτικές αλλαγές και αποκαλύπτει πώς αυτές υλοποιούνται και πώς δρουν στο περιβάλλον της βιβλιοθήκης.

# 2 Μοντέλα ανάπτυξης λογισμικού

## 2.1 Σκοπός:

Στο κεφάλαιο αυτό παρουσιάζονται τα πιο σημαντικά μοντέλα ανάπτυξης λογισμικού. Ο σκοπός του κεφαλαίου είναι να παρουσιάσει για κάθε μοντέλο τις εργασίες που περιλαμβάνει, την σειρά που θα πρέπει αυτές να γίνουν, προϋποθέσεις καθώς και τα κριτήρια με τα οποία θα ολοκληρωθούν. (Per Kroll, 2003)

## 2.2 Έννοιες Κλειδιά:

• Μοντέλα Ανάπτυξης Λογισμικού
• Συμβατικά και Εναλλακτικά Μοντέλα Ανάπτυξης Λογισμικού
• Μεθοδολογία Ανάπτυξης
• Μοντέλο Καταρράκτη
• Μοντέλο Προτυποποίησης
• Μοντέλο Λειτουργικής Επαύξησης
• Λειτουργικό Μοντέλο
• Μοντέλο Αυτόματου Προγραμματισμού
• Μοντέλο Επαναχρησιμοποίησης Λογισμικού
• Αντικειμενοστραφές Μοντέλο
• Σπειροειδές Μοντέλο
• Διοίκηση Έργων Λογισμικού

## 2.3 Εισαγωγικές Παρατηρήσεις:

Είναι πολύ σημαντικό για την Επιστήμη της Μηχανικής (Βεσκούκης, 2000), η εύρεση και θεμελίωση μεθόδων για να περιγράφεται, να κατασκευάζεται και να συντηρείται λογισμικό καλής ποιότητας, παρέχοντας οδηγίες που αφορούν στις ενέργειες που θα πρέπει να πραγματοποιηθούν. Ακόμα, την σειρά που θα πρέπει οι ενέργειες αυτές να γίνουν, τις προϋποθέσεις καθώς και τα κριτήρια με τα οποία θα ολοκληρωθεί μια ενέργεια για να γίνει μετάβαση στην επόμενη.

Δεν υπάρχει μοναδιαίος τρόπος για τον προσδιορισμό των ενεργειών ανάπτυξης λογισμικού αν και σε γενικές γραμμές θα πρέπει να περιλαμβάνονται οι διαδικασίες της προδιαγραφής, της ανάπτυξης, της επαλήθευσης και της εξέλιξης (Βεσκούκης, 2000).

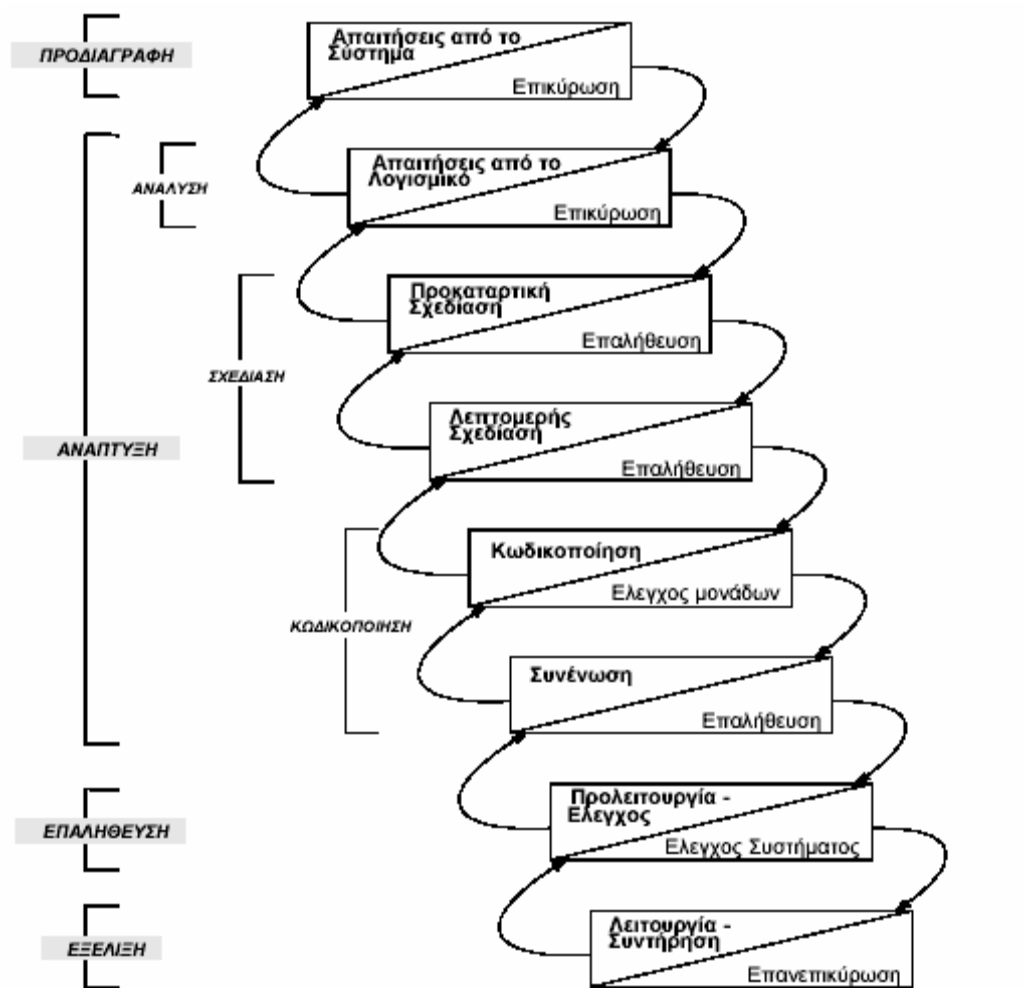
Έχουν αναπτυχθεί αρκετά μοντέλα ανάπτυξης λογισμικού και στο κείμενο που ακολουθεί περιγράφονται μερικά από αυτά. Στην πρώτη ενότητα παρατίθενται

συνοπτικά τα συμβατικά μοντέλα ανάπτυξης λογισμικού με μεγαλύτερη έμφαση στο μοντέλο της προτυποποίησης ενώ στην δεύτερη ενότητα αναλύονται κάποια εναλλακτικά μοντέλα με μεγαλύτερη έμφαση στο σπειροειδές μοντέλο και στην τρίτη ενότητα προτείνεται το καταλληλότερο μοντέλο για την ανάπτυξη λογισμικού.
Τέλος, στην τέταρτη ενότητα αναπτύσσεται συνοπτικά το μεγάλο ζήτημα της διαχείρισης / διοίκησης των μονάδων / σπουδών ανάπτυξης λογισμικού. (Βεσκούκης, 2000)

## 2.4 Μοντέλα Ανάπτυξης Λογισμικού

### 2.4.1 Μοντέλο Καταρράκτη και παραλλαγές του

Το μοντέλο του Καταρράκτη (waterfall ή linear sequential model) αναπτύχθηκε από τον Royce, το 1970 [Royce, 1970; Boehm, 1976; Boehm, 1984; Ramamoorthy, 1984] και περιλαμβάνει 8 διακριτές φάσεις.




**Εικόνα 1: Φάσεις Μοντέλου Καταρράκτη**

Ήταν το πρώτο μοντέλο που δημιουργήθηκε και έγινε ευρέως αποδεκτό, ενώ ακόμα παραμένει δημοφιλές ιδιαίτερα για μικρά ή μεσαία μεγέθη εφαρμογών αφού συμβάλλει στην επιτυχή κατασκευή αξιόπιστων προϊόντων σε μικρό χρονικό διάστημα.

Στο μοντέλο αυτό οι διάφορες φάσεις διαχωρίζονται και ακολουθούνται σειριακά. Η κάθε φάση παράγει ενδιάμεσα προϊόντα τα οποία χρησιμοποιούντα από τις επόμενες φάσεις και κορυφώνεται από μια διαδικασία επικύρωσης ή επαλήθευσης των προϊόντων που παράγονται, με σκοπό να απαλειφθούν τυχόν σφάλματα.

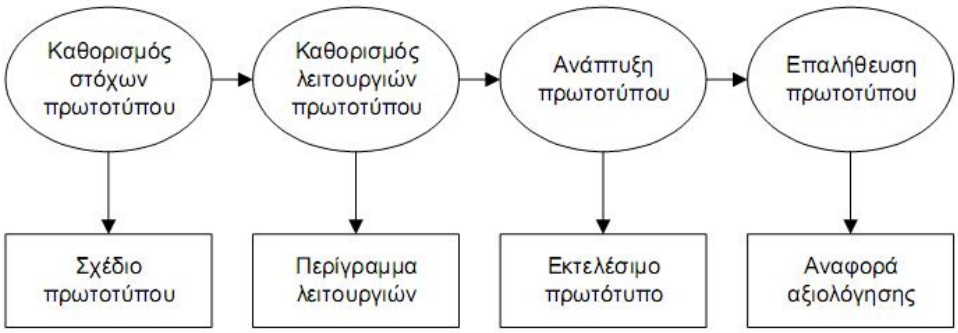
Τα μειονεκτήματα που παρουσιάζει το μοντέλο του καταρράκτη δηλαδή το ότι γνωρίζουμε αν θα είναι ικανοποιημένος ο χρήστης μόνο στο τελικό στάδιο (ουσιαστικά αργά), επίσης το ότι δεν προβλέπει επαναχρησιμοποίηση του λογισμικού που πιθανά υπάρχει και ότι είναι απαιτητικό τόσο σε χρόνο όσο και σε κόστος και αρκετά δύσκαμπτο εργαλείο, προσπαθούν να απαλείψουν τα μοντέλα IEEE (IEEE

variant of a life cycle model) και το μοντέλο V (V model) που αποτελούν παραλλαγές του.

Οι παραλλαγές αυτές διαφοροποιούνται περισσότερο στον τρόπο που γίνεται ο έλεγχος, η διόρθωση, η επαλήθευση, η επικύρωση καθώς και ο τρόπος που θα πραγματοποιηθούν αυτές δηλαδή σε ποια φάση θα πρέπει να επιστρέψουμε αν χρειαστεί ενώ παράλληλα οι φάσεις που περιλαμβάνονται χαρακτηρίζονται είτε από την σύμπτυξη είτε από τον επιμέρους διαχωρισμό των αντίστοιχων φάσεων του αντίστοιχου μοντέλου του καταρράκτη.

### 2.4.2 Μοντέλο Προτυποποίησης

Ένας τρόπος να δούμε την προτυποποίηση είναι ως μια τεχνική για μείωση του ρίσκου. Το πιο σημαντικό ρίσκο στην ανάπτυξη λογισμικού είναι τα λάθη και περισσότερο οι παραλείψεις που προκύπτουν από μη σαφείς απαιτήσεις των χρηστών για το τελικό σύστημα. Το κόστος της διόρθωσης αυτών των λαθών και παραλείψεων σε επόμενα στάδια μπορεί να είναι πολύ υψηλό. Είναι προφανές ότι η δημιουργία ενός πρωτοτύπου μπορεί να μειώσει τον αριθμό των προβλημάτων των απαιτήσεων και ως εκ τούτου να μειώσει το συνολικό κόστος ανάπτυξης. Μια σχηματική αναπαράσταση της διαδικασίας ανάπτυξης ενός πρωτοτύπου φαίνεται στο παρακάτω σχήμα.




**Εικόνα 2: Σχηματική αναπαράσταση της διαδικασίας ανάπτυξης ενός πρωτοτύπου**

Αρχικά θα πρέπει να καθοριστούν επακριβώς οι στόχοι του πρωτότυπου. Το πρωτότυπο σύστημα μπορεί να αφορά τη διεπιφάνεια χρήστη ή να περιέχει τις λειτουργίες εκείνες που θεωρούνται περισσότερο κρίσιμες. Είναι προφανές ότι ένα πρωτότυπο δεν μπορεί να καλύπτει όλες τις απαιτήσεις του συστήματος. Για τον λόγο αυτό κάθε φορά θα πρέπει να ορίζονται πλήρως οι απαιτήσεις που αυτό θα καλύπτει,

αλλιώς τελικά μπορεί να μην λάβουμε τα πλεονεκτήματα που μας προσφέρει η μέθοδος αυτή. Το επόμενο στάδιο αφορά στο να αποφασιστεί ποιες ενδεχομένως λειτουργίες του τελικού συστήματος δεν θα πρέπει να συμπεριληφθούν γιατί έχει πολύ υψηλό κόστος εάν το πρωτότυπο δημιουργηθεί με όλες τις λειτουργίες του τελικού συστήματος.

Βέβαια θα μπορούσε να αποφασιστεί να περιλαμβάνονται όλες οι λειτουργίες που έχει αποφασιστεί αλλά σε μειωμένο επίπεδο (πχ. χωρίς διαχείριση λαθών). Η τελευταία φάση, μετά την ανάπτυξη του πρωτοτύπου, είναι η επαλήθευση του πρωτότυπου και είναι ίσως η πιο σημαντική φάση. Θα πρέπει να καταγραφούν συμπεράσματα για το πως νιώθουν οι χρήστες με το σύστημα, αν γίνεται κατανοητό το περιβάλλον και η λειτουργία του και να βρεθούν τυχόν λάθη και προβλήματα.

Τα **πλεονεκτήματα** της χρήσης πρωτότυπου είναι ότι ανακαλύπτονται και διορθώνονται:

• Παρεξηγήσεις μεταξύ των χρηστών και των δημιουργών.

• Παραλειπόμενες υπηρεσίες στο σύστημα

• Δυσκολίες στη χρήση

• Ασυνέχειες και κενά στις προδιαγραφές

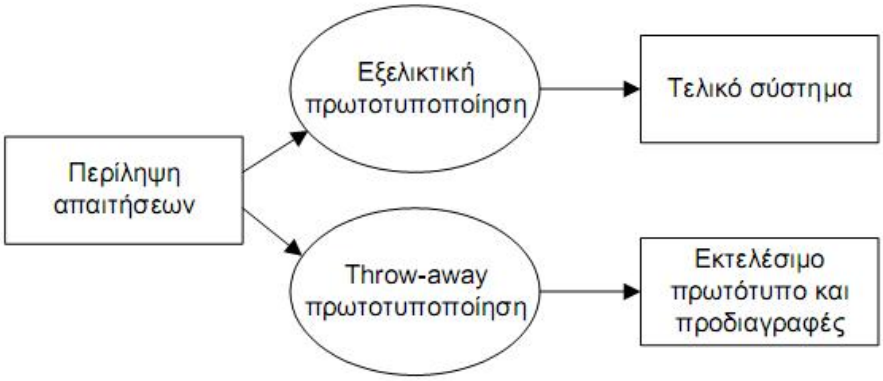
Η προτυποποίηση μπορεί να χρησιμοποιηθεί και για άλλους σκοπούς, όπως στην εκπαίδευση των χρηστών, δηλαδή το πρωτότυπο να χρησιμοποιηθεί ως εκπαιδευτικό εργαλείο για την εκμάθηση του τελικού συστήματος, αλλά ταυτόχρονα είναι και ένας τρόπος μείωσης του ρίσκου, αφού περιορίζονται τα λάθη και οι παραλείψεις. Αν τα λάθη αφεθούν για διόρθωση στις τελευταίες φάσεις του κύκλου ζωής το κόστος αυξάνεται κατακόρυφα.

Το βασικό **μειονέκτημα** του μοντέλου της προτυποποίησης είναι ότι το κόστος ανάπτυξής του αποτελεί ένα μεγάλο μέρος του συνολικού κόστους του συστήματος που αναπτύσσεται. Πολλές φορές είναι οικονομικά πιο συμφέρον να μεταβληθεί το τελικό προϊόν από το να δημιουργηθεί ένα πρωτότυπο.

Είναι προφανές, ότι είναι πολύ δύσκολο να προβλεφτεί ποιες ακριβώς δυσκολίες θα αντιμετωπίσει ο τελικός χρήστης από την καθημερινή χρήση ενός νέου συστήματος λογισμικού. Ιδιαίτερα εάν αναφερόμαστε σε μεγάλα συστήματα λογισμικού η δυσκολία αυτή μπορεί να καταφανεί μόνο όταν το ολοκληρωμένο σύστημα αναπτυχθεί και τεθεί σε λειτουργία. Για να αντιμετωπιστεί αυτή η δυσκολία μπορεί να χρησιμοποιηθεί η **εξελικτική (evolutionary) προτυποποίηση**. Κατά τη προσέγγιση αυτή δημιουργείται μια περιορισμένη (ατελής) έκδοση του συστήματος

πάνω στην οποία γίνονται διορθώσεις και προσθήκες καθώς απαιτήσεις είτε διευκρινίζονται είτε ανακαλύπτονται είτε βελτιώνονται μέχρι να καταλήξουμε σε μια έκδοση που να είναι επαρκής και ικανοποιητική.

Το βασικό **πρόβλημα** του μοντέλου της εξελικτικής προτυποποίησης είναι ότι με τις συνεχείς διορθωτικές και προσθετικές παρεμβολές παράγεται *'μπλεγμένος' κώδικας* (spaghetti code) που είναι πολύ δύσκολα συντηρήσιμος. Εναλλακτικά και προκειμένου να αποφύγουμε το φαινόμενο του 'μπλεγμένου κώδικα', θα μπορούσε να δημιουργηθεί ένα πρωτότυπο το οποίο σκοπό αποκλειστικό θα είχε την αποσαφήνιση των απαιτήσεων και να παράσχει πληροφορίες για την εκτίμηση του ρίσκου του τελικού συστήματος. Μετά την αξιολόγηση το πρωτότυπο σύστημα '**πετιέται**' και δεν χρησιμοποιείται για την ανάπτυξη του συστήματος **(throw-away prototype)**.



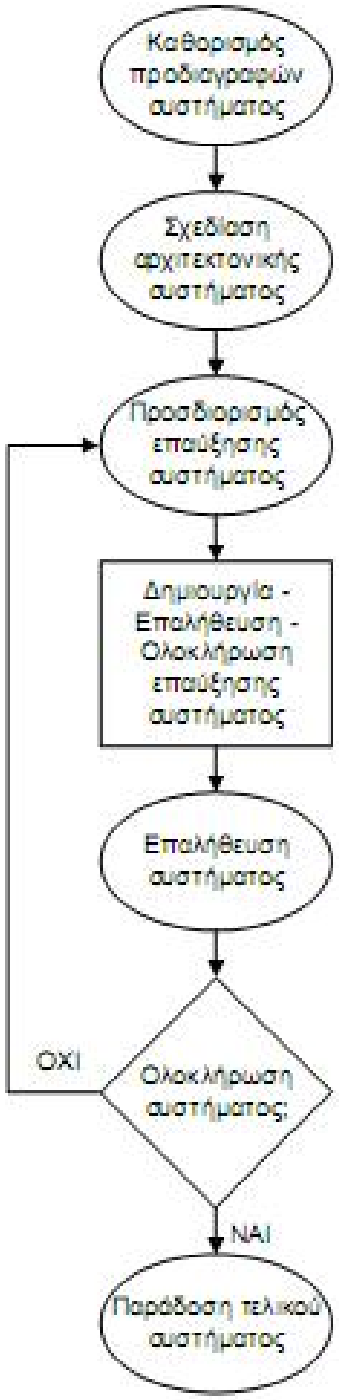

**Εικόνα 3: Throw-away prototype**

Ο χρόνος που απαιτείται για την ανάπτυξη ενός συστήματος μπορεί να μειωθεί ακόμα περισσότερο εάν κάποια τμήματα του συστήματος μπορούν να επαναχρησιμοποιηθούν. Έτσι τα πρωτότυπα μπορούν να κατασκευαστούν ακόμα πιο γρήγορα εάν υπάρχει μια βιβλιοθήκη με επαναχρησιμοποιήσιμες ψηφίδες (reusable components) και φυσικά κάποιος τρόπος σύνθεσης των ψηφίδων αυτών. Η επαναχρησιμοποίηση ψηφίδων αρμόζει περισσότερο στην throw – away προσέγγιση της πρωτοτυποποίησης. (Beck, 1999)

### 2.4.3 Μοντέλο Λειτουργικής Επαύξησης

Μια εναλλακτική διαδικασία που συνδυάζει τα πλεονεκτήματα της εξελικτικής προσέγγισης με τον έλεγχο που απαιτείται για μεγάλα συστήματα είναι η **λειτουργική επαύξηση (incremental development)**. Σύμφωνα με το μοντέλο αυτό

αρχικά αναπτύσσεται μια έκδοση του συστήματος που περιέχει τις περισσότερο σημαντικές και κρίσιμες λειτουργίες. Από την χρησιμοποίηση αυτής της έκδοσης κερδίζεται εμπειρία η οποία χρησιμοποιείται για την βελτίωσή της. Στην συνέχεια γίνεται μια προσαύξηση η οποία επεκτείνει την προηγούμενη έκδοση η οποία περιέχει και άλλες λειτουργίες. Η νέα έκδοση εκλεπτύνεται και προσαυξάνεται με την σειρά της με τον ίδιο τρόπο έως ότου κατασκευαστεί η τελική έκδοση. Το μοντέλο ενδείκνυται στις περιπτώσεις που υπάρχει σαφής γνώση και πολύ μικρή ή καθόλου μεταβλητότητα των απαιτήσεων του υπό ανάπτυξη λογισμικού. Άρα πρόκειται για μοντέλο που χρησιμοποιείται σε λίγες περιπτώσεις μια και το βασικό πρόβλημα της ανάπτυξης λογισμικού είναι η ασάφεια (σε μικρότερο ή μεγαλύτερο βαθμό) των απαιτήσεων του συστήματος.




**Εικόνα 4: Μοντέλο Λειτουργικής Επαύξησης**

Με αυτό το μοντέλο αποφεύγονται προβλήματα που προκύπτουν από τις συνεχείς αλλαγές, όπως στην εξελικτική προτυποποίηση. Η αρχιτεκτονική του συστήματος καθορίζεται σχετικά νωρίς, και λειτουργεί σαν πλαίσιο. Τα μέρη που αποτελούν το σύστημα αναπτύσσονται με επαυξήσεις και παραδίδονται με αυτό τον τρόπο (βλέπε και Βεσκούκης, 2000).

### 2.4.4 Εναλλακτικά Μοντέλα Ανάπτυξης Λογισμικού

Τα εναλλακτικά μοντέλα έχουν προταθεί ως μια εναλλακτική λύση στα συμβατικά με σκοπό να ελαττώσουν τα προβλήματα και τις αδυναμίες που έχουν τα συμβατικά μοντέλα (Σκορδαλάκης, 1991; Sommerville, 2001).

Το Λειτουργικό Μοντέλο (operational model) [Zave, 1984] χρησιμοποιεί τις προδιαγραφές που είναι λειτουργικές και περιγράφουν τι θα κάνει το σύστημα έμμεσα, μέσα από μια περιγραφή η οποία δείχνει πως αυτό θα λειτουργεί. Η περιγραφή αυτή γίνεται σε μια γλώσσα που να μπορεί να εκτελεστεί ώστε να αξιολογηθεί και έτσι να γίνει φανερή η συμπεριφορά του συστήματος. Οι λειτουργικές προδιαγραφές που χρησιμοποιούνται σύμφωνα με το μοντέλο αυτό είναι ένα είδος πρωτότυπου στο οποίο είναι εμφανής όλη η λειτουργική συμπεριφορά του συστήματος, χρησιμοποιώντας διαφορετικά μέσα από αυτά που θα χρησιμοποιούσε το τελικό σύστημα.

Αξιολογώντας τη συμπεριφορά του συστήματος, οι χρήστες μπορούν να κάνουν παρατηρήσεις και αλλαγές στις λειτουργικές προδιαγραφές. Ο κύκλος αυτός αξιολόγησης – αλλαγών επαναλαμβάνεται έως ότου θεωρηθεί ότι το σύστημα έχει την επιθυμητή λειτουργικότητα. Έτσι ολοκληρώνεται η φάση των απαιτήσεων και στην συνέχεια μπορεί να χρησιμοποιηθεί για παράδειγμα το μοντέλο του καταρράκτη από την φάση της σχεδίασης και κάτω.

Βασικό πρόβλημα του μοντέλου αποτελεί ότι χρησιμοποιεί εκτελέσιμες γλώσσες προδιαγραφών (executable specification languages) που είναι αυστηρά τυπικές (formal) και απαιτούν ιδιαίτερες γνώσεις από την ομάδα ανάπτυξης αλλά και εργαλεία υλοποίησης (compilers – interpreters, specification languages). (Madnick, 1991) (Ian Sommerville, 2008)

### 2.4.5 Μοντέλο Αυτόματου Προγραμματισμού

Το μοντέλο αυτόματου προγραμματισμού (automatic programming model) βασίζεται στην ιδέα της δημιουργίας ενός συστήματος που να μπορεί να δημιουργήσει λογισμικό αυτόματα αφού πρώτα του δωθούν οι προδιαγραφές του προβλήματος (Σχήμα 3.4)


**Εικόνα 5: Μοντέλο αυτόματου προγραμματισμού**

Η αυτοματοποιημένη δημιουργία λογισμικού είναι μια παλιά ιδέα που χρησιμοποιήθηκε και για την δημιουργία μεταφραστών γλωσσών προγραμματισμού (μετα-φραστές / meta-translators). Το μοντέλο αυτόματου προγραμματισμού ενδείκνυται για τις περιπτώσεις που οι απαιτήσεις του συστήματος είναι σαφώς καθορισμένες ώστε να μπορούν να περιγραφούν με ένα πολύ τυπικό τρόπο όπως είναι οι γραμματικές χωρίς συμφραζόμενα κλπ (Πιντέλας, 2001).

Τα εργαλεία που χρησιμοποιούνται (μετα-μεταφραστές και οι αντίστοιχες γραμματικές) απαιτούν υψηλές γνώσεις πληροφορικής και επομένως πολύ εξειδικευμένη και έμπειρη ομάδα ανάπτυξης.

### 2.4.6 Μετασχηματιστικό Μοντέλο

Το μετασχηματιστικό μοντέλο (transformational model) [Partsch, 1983; Agresti, 1986] είναι μια παραλλαγή του μοντέλου αυτόματου προγραμματισμού και υποστηρίζει τον μετασχηματιστικό προγραμματισμό. Αρχικά ορίζονται οι προδιαγραφές του συστήματος με ένα τυπικό τρόπο και στη συνέχεια με μια σειρά από αυτόματους μετασχηματισμούς μετατρέπονται σε κώδικα.

**Πλεονέκτημα** του μοντέλου είναι ότι δεν χρειάζεται έλεγχος ορθότητας, παρά μόνον για τις τυπικές προδιαγραφές. Η συντήρηση γίνεται με αλλαγές στις τυπικές προδιαγραφές.

**Μειονεκτήματα** του μοντέλου αυτού είναι τα ίδια με εκείνα του μοντέλου του αυτόματου προγραμματισμού.

### 2.4.7 Μοντέλο Επαναχρησιμοποίησης Λογισμικού

Με το μοντέλο επαναχρησιμοποίησης λογισμικού (software reusability model) γίνεται χρήση ήδη υπάρχοντος και δοκιμασμένου λογισμικού, σχεδίων και κώδικα. Οι υπάρχουσες ψηφίδες λογισμικού (με ελεγμένη ορθότητα) ενσωματώνονται σε νέα προϊόντα λογισμικού.

Η διαδικασία αυτή δεν είναι εύκολη, αφού παρουσιάζονται δυσκολίες, λόγω της ανυπαρξίας εργαλείων και τεχνικών καταλλήλων για αυτή τη δουλειά, αλλά και της έλλειψης προτύπων κατασκευής ψηφίδων λογισμικού που να μπορούν να επαναχρησιμοποιηθούν.

Τα βασικά πλεονεκτήματα του μοντέλου είναι η συντόμευση του χρονικού διαστήματος κατασκευής λογισμικού αλλά και η βελτίωση της αξιοπιστίας του αφού στηρίζεται σε έτοιμα, δοκιμασμένα και άρα αξιόπιστα τμήματα λογισμικού. Τα συστατικά λογισμικού που θα επαναχρησιμοποιηθούν μπορεί να είναι πολλών και διαφόρων μεγεθών, όπως για παράδειγμα :

• Επαναχρησιμοποίηση ολόκληρων συστημάτων εφαρμογών (application system reuse) τα οποία είτε ενσωματώνονται στο καινούριο σύστημα χωρίς αλλαγή είτε δημιουργούνται ολόκληρες οικογένειες εφαρμογών που μπορούν να τρέξουν σε διαφορετικές πλατφόρμες ώστε να ικανοποιήσουν συγκεκριμένες ανάγκες.

• Επαναχρησιμοποίηση ψηφίδων (component reuse) όπου διάφορα ‘συστατικά’ μιας εφαρμογής από ένα υποσύστημα μέχρι ένα μεμονωμένο αντικείμενο μπορούν να επαναχρησιμοποιηθούν. Για παράδειγμα αντικείμενο τύπου ‘χρονόμετρο’ μπορεί να επαναχρησιμοποιηθεί πολλές φορές στην ανάπτυξη ενός λογισμικού φυσικής, χημείας κλπ.

• Επαναχρησιμοποίηση κάποιας λειτουργίας ή συνάρτησης (function reuse) όπου συστατικά (συναρτήσεις, διαδικασίες) που υλοποιούν μια απλή λειτουργία όπως μια μαθηματική συνάρτηση επαναχρησιμοποιούνται σε άλλες εφαρμογές. Αυτού του είδους η επαναχρησιμοποίηση που βασίζεται στις βιβλιοθήκες είναι και η πιο διαδεδομένη.

Το προφανές κέρδος από την επαναχρησιμοποίηση είναι η μείωση του κόστους ανάπτυξης δεδομένου ότι λιγότερα συστατικά του συστήματος χρειάζεται να προσδιοριστούν, να σχεδιαστούν, να υλοποιηθούν και να αξιολογηθούν.

### 2.4.8 Αντικειμενοστραφές Μοντέλο

Το αντικειμενοστραφές μοντέλο (object-oriented model) βασίζεται στον αντικειμενοστραφή προγραμματισμό (βλέπε και Βεσκούκης, 2000). Αναπτύσσεται με τρόπο παρόμοιο με το μοντέλο του καταρράκτη, αλλά διαφέρει σε δύο βασικά σημεία:

• Οι διάφορες φάσεις υπερκαλύπτονται μεταξύ τους

• Η ανάπτυξη του, αν χρειαστεί οπισθοδρομεί στην προηγούμενη φάση, εκτός από την τελευταία που οπισθοδρομεί στην αρχή.

Το κύριο **πλεονέκτημα** του μοντέλου είναι ότι κάνει χρήση επαναχρησιμοποιήσιμων μονάδων και με αυτό τον τρόπο συντομεύεται τόσο η φάση της ανάπτυξης όσο και η φάση της συντήρησης.

### 2.4.9 Σπειροειδές μοντέλο

Το σπειροειδές μοντέλο ή αλλιώς μοντέλο του Boehm (spiral model ή Boehm's spiral model) (Boehm, 1988) είναι ένα δημοφιλές μοντέλο που ουσιαστικά είναι γενίκευση των μοντέλων της λειτουργικής επαύξησης και της πρωτοτυποποίησης και διαφέρει από όλα τα άλλα μοντέλα που βασίζονται στο ακολουθιακό μοντέλο του καταρράκτη. Το μοντέλο του Boehm έχει τη μορφή μιας σπείρας (βλέπε και Βεσκούκης, 2000) όπου κάθε γύρος στην σπείρα αναπαριστά και μια φάση έτσι όπως αυτή καθορίζεται κάθε φορά από την διοίκηση.

Η διαδοχή των φάσεων δεν γίνεται ούτε σταθερά ούτε γραμμικά, ενώ η εκτέλεσή τους μπορεί να γίνει είτε με τη φορά της σπείρας, είτε με την αντίθετη φορά, ανάλογα με το ρίσκο που λαμβάνεται και το οποίο αποτελεί θεμελιώδη έννοια στο σπειροειδές μοντέλο.

Σε κάθε γύρο που διανύεται υπάρχουν οι εξής φάσεις:

• Καθορισμός στόχων, εναλλακτικών λύσεων και υπολογισμός περιορισμών.

• Ανάλυση και υπολογισμός του ρίσκου και προσπάθεια μείωσης του.

• Ανάπτυξη και επαλήθευση ενδιάμεσου προϊόντος – εφόσον η προηγούμενη φάση δεν έδειξε κάποιο σοβαρό ρίσκο – πρόσθεση νέων λειτουργικών προδιαγραφών.

• Σχεδιασμός των επόμενων βημάτων.

Η βασική διαφορά του παραπάνω μοντέλου από τα υπόλοιπα μοντέλα είναι ότι σε αυτό υπολογίζεται πριν την έναρξη κάθε φάσης το ρίσκο, γεγονός που ουσιαστικά

αποτελεί και το βασικό του **πλεονέκτημα** αν και ο υπολογισμός και η ανάλυση του ρίσκου δεν είναι εύκολη υπόθεση πρακτικά.

#### *2.4.9.1 Σχόλιο μελέτης:*

Η έννοια ρίσκο είναι δύσκολο να καθοριστεί πλήρως. Μια απλή εξήγηση που θα μπορούσαμε να δώσουμε για το ρίσκο είναι ότι είναι οτιδήποτε θεωρείται ότι μπορεί να πάει στραβά. Για παράδειγμα, αν χρησιμοποιηθεί ένα καινούριο πακέτο για τρισδιάστατα γραφικά, ρίσκο θα αποτελέσει κατά πόσο είναι δύσκολο να το μάθουν γρήγορα οι εμπλεκόμενοι στην ομάδα ανάπτυξης. Άλλο σημαντικό ρίσκο στην ανάπτυξη λογισμικού είναι τα λάθη και περισσότερο οι παραλείψεις των απαιτήσεων του συστήματος. Τα ρίσκα είναι συνέπεια ελλιπούς πληροφόρησης. Για την μείωση ή εξάλειψη τους χρειαζόμαστε περισσότερες πληροφορίες. Για το παραπάνω παράδειγμα, το ρίσκο μπορεί να μειωθεί κάνοντας στην ομάδα ανάπτυξης ένα ταχύρυθμο πρόγραμμα εκμάθησης, ή προσλαμβάνοντας νέα ομάδα ανάπτυξης με γνώση του συγκεκριμένου πακέτου.[1] Η μελέτη των παραπάνω μοντέλων ήταν απαραίτητη για την απόκτηση της αναγκαίας γνώσης. Αφού εξετάσαμε τα μοντέλα ανάπτυξης λογισμικού, πρέπει να διαλέξουμε εκείνο που εξυπηρετεί καλύτερα τις ανάγκες μας.

## 2.5 Ποιο είναι το Κατάλληλο Μοντέλο για Ανάπτυξη Λογισμικού;

Από όσα αναφέρθηκαν στις προηγούμενες ενότητες η επιλογή του κατάλληλου μοντέλου ανάπτυξης λογισμικού είναι λογικό να μην περιλαμβάνει τα μοντέλα αυτόματου προγραμματισμού. Και αυτό γιατί η ανάπτυξη του λογισμικού είναι μια δημιουργική διαδικασία που εμπλέκει πολλών περιοχών ειδικούς και όχι μια αυτόματη διαδικασία. Πολλά, επίσης, μοντέλα όπως το λειτουργικό χρησιμοποιούνται περισσότερο για την ανάπτυξη ειδικών περιπτώσεων λογισμικού.
Το μοντέλο του καταρράκτη, και όλα τα μοντέλα αυτής της μορφής, αν και είναι δημοφιλής επιλογή, είναι επίσης απορριπτέα γιατί όπως έχει αναφερθεί, ξέρουμε εάν έχουμε κατασκευάσει αυτό που θέλαμε μόνο όταν η διαδικασία ανάπτυξης έχει ολοκληρωθεί. Επίσης, δεδομένου ότι το προϊόν κάθε φάσης θεωρείται τελικό είναι σχεδόν αδύνατο να παρέμβουμε σε αυτό όταν βρισκόμαστε σε αρκετά μεταγενέστερο

[1] (Ian Sommerville, 2008)

στάδιο ανάπτυξης στο οποίο θα παρουσιαστεί κάποιο πρόβλημα ή λάθος. Και αυτό γιατί αυξάνεται γεωμετρικά το κόστος της διόρθωσης.

Το μοντέλο επαναχρησιμοποίησης λογισμικού δεν θα μπορούσε να οδηγήσει σε ολοκληρωμένη ανάπτυξη λογισμικού παρόλα αυτά θα μπορούσε να βοηθήσει ιδιαίτερα ως τμήμα ενός άλλου μοντέλου.

Το μοντέλο πρωτοτυποποίησης θα μπορούσε να δώσει έναν τρόπο ανάπτυξης δεδομένου ότι ειδικά στην εξελικτική του μορφή μπορεί να δώσει πληροφορίες που βοηθούν στην ανάπτυξη πολύ πιο γρήγορα από το μοντέλο του καταρράκτη, τόσο ως προς την εφικτότητα κατασκευής του τελικού συστήματος όσο και ως προς την ακρίβεια των προδιαγραφών των απαιτήσεων από το λογισμικό.

Επίσης, το μοντέλο της throw-away πρωτοτυποποίησης είναι μια σοβαρή υποψηφιότητα για την ανάπτυξη λογισμικού διότι αφενός δίνει πληροφορίες για την εφικτότητα του τελικού συστήματος αλλά και γιατί αποσαφηνίζει τις απαιτήσεις του συστήματος. Αυτό συμβαίνει διότι το τελικό σύστημα όπως προκύπτει με την χρήση αυτού του μοντέλου είναι πιο συντηρήσιμο, τελικά για ολόκληρη της διάρκεια ζωής του λογισμικού προκύπτει σημαντική μείωση του συνολικού κόστους αν και όπως έχει ήδη αναφερθεί επειδή το αρχικό πρωτότυπο θα ‘πεταχτεί’, αυξάνεται ως ένα βαθμό το κόστος ανάπτυξης από το γεγονός αυτό.

Το σπειροειδές μοντέλο ανάπτυξης διαφέρει από όλα τα προηγούμενα κυρίως στο ότι σε κάθε φάση ανάπτυξης υπάρχει μια άμεση εκτίμηση του ρίσκου. Καταρχήν σε αυτό καθορίζονται οι στόχοι, οι εναλλακτικές λύσεις και οι περιορισμοί τους και στη συνέχεια αξιολογούνται αυτές οι λύσεις και καθορίζονται στρατηγικές επίλυσης του ρίσκου εάν αυτό είναι μεγάλο. Με αυτό τον τρόπο είναι δυνατό να αναπτυχθεί το λογισμικό χωρίς να υπάρχει ο κίνδυνος της μη ολοκλήρωσης μιας φάσης ανάπτυξής του και συγχρόνως μετά από κάθε φάση έχουμε ένα ενδιάμεσο πρωτότυπο που μπορεί σταδιακά να αξιολογείται.

Λαμβάνοντας υπόψη ότι η σύγχρονη τάση στην ανάπτυξη λογισμικού ουσιαστικά χρησιμοποιεί κάποιες γενικές κατευθύνσεις από τις υπάρχουσες ιδέες (μοντέλα) αλλά αφήνει αρκετούς βαθμούς ελευθερίας στον κατασκευαστή, μπορούμε να συμπεράνουμε ότι το μοντέλο καταρράκτη μπορεί να αποτελέσει βάση για την ανάπτυξη λογισμικού με ορισμένες παραλλαγές οι οποίες εξειδικεύονται κάθε φορά ανάλογα με το περιβάλλον ανάπτυξης, το συγκεκριμένο πρόβλημα κλπ.[2]

[2] (Madachy, 2008)

## 2.6 Διοίκηση Έργων και Διοίκηση Έργων Λογισμικού

### 2.6.1 Κλασική Διοίκηση Έργων

Η **διοίκηση και διαχείριση έργων (project management)** αναπτύχθηκε ως ξεχωριστό γνωστικό πεδίο από την εφαρμογή των αρχών της διοίκησης αλλά και της επιχειρησιακής έρευνας σε διάφορους τομείς εφαρμογής, όπως αυτός των κατασκευών, της μηχανολογίας, των μεγάλων στρατιωτικών προγραμμάτων.

#### *2.6.1.1 Ιστορία*

Πατέρας του γνωστικού πεδίου της διαχείρισης έργων θεωρείται ο Χένρι Γκαντ, Αμερικανός μηχανικός και κοινωνικός επιστήμονας, ο οποίος εισήγαγε τις αρχές του προγραμματισμού και ελέγχου στη διαχείριση έργων. Το γνωστό διάγραμμα Γκαντ, ένα ραβδόγραμμα που παρουσιάζει τις δραστηριότητες του έργου, ονομάστηκε έτσι από αυτόν. Ο Γκαντ μαζί με τον Φρέντερικ Τέιλορ έθεσαν τις θεμέλιες αρχές της διαχείρισης έργων. Ο Τέιλορ έθεσε τις αρχές της επιστημονικής διαχείρισης (scientific management).

Οι σύγχρονες αρχές της διαχείρισης έργων οι οποίες έκαναν τη διαχείριση έργων ένα διακριτό γνωστικό αντικείμενο αλλά και ένα επάγγελμα αναπτύχθηκαν την δεκαετία του 1950. Την δεκαετία αυτή αναπτύχθηκαν δύο βασικά μαθηματικά μοντέλα χρονοπρογραμματισμού δραστηριοτήτων, οι μέθοδοι PERT και CPM οι οποίες αποτέλεσαν σταθμό στη διαχείριση έργων.

Η μέθοδος PERT (Program Evaluation and Review Technique) αναπτύχθηκε από το Ναυτικό των Ηνωμένων Πολιτειών για το έργο για της ανάπτυξης των πυραυλικών συστημάτων Polaris. Αντίστοιχα η μέθοδος CPM (Critical Path Method) γνωστή στα ελληνικά και ως μέθοδος κρίσιμου διαδρομής αναπτύχθηκε από τις εταιρείες DuPont Corporation και Remington Rand Corporation με σκοπό την διαχείριση έργων συντήρησης. Η διάδοση και αποδοχή των μεθόδων αυτών έγινε με ταχύτατο τρόπο έτσι ώστε σήμερα αποτελούν βασικές μεθόδους για τη διαχείριση έργων.

Σήμερα ο χώρος της διαχείρισης έργων θεωρείται ιδιαίτερα αναπτυγμένος και προσελκύει ιδιαίτερο ενδιαφέρον τόσο στον ιδιωτικό, δημόσιο τομέα όσο και στην ακαδημαϊκή κοινότητα. Απόδειξη του γεγονότος αυτού αποτελεί η ύπαρξη πολλών και ιδιαίτερα δραστήριων διεθνών οργανισμών που έχουν ως στόχο την ανάπτυξη του

γνωστικού πεδίου της διαχείρισης έργων. Μεταξύ αυτών ξεχωρίζουν οι International Project Management Association και το Project Management Institute.

### 2.6.2 Τι είναι έργο

Έργο είναι ένα 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 παρόμοια προϊόντα ή υπηρεσίες.

Οι ιδιότητες αυτές των έργων, να είναι προσωρινά αλλά και μοναδικά εγχειρήματα, έρχονται σε αντίθεση με τη δομή που έχουν οι περισσότερες επιχειρήσεις που λειτουργούν βάση διαδικασιών που έχουν σταθερό και μόνιμο χαρακτήρα. Η διαχείριση αυτών των ιδιοτήτων είναι συχνά δύσκολη μια και απαιτεί ιδιαίτερες ικανότητες από διαφορετικά γνωστικά πεδία.

Έτσι η πρώτη πρόκληση που αντιμετωπίζουμε στη διαχείριση έργων είναι να εξασφαλίσουμε ότι το έργο εκτελείται και παραδίδεται λαμβάνοντας υπόψη καθορισμένους περιορισμούς. Περιορισμοί που μπορεί να είναι ανεπαρκής διαθέσιμος χρόνος, περιορισμένος προϋπολογισμός κ.α. Η δεύτερη πρόκληση που είναι και πιο φιλόδοξη, είναι η βελτιστοποίηση που απαιτείται να γίνει σε όλους τους παράγοντες που επηρεάζουν την εκτέλεση ενός έργου. Επομένως, ένα έργο είναι ένα προσεκτικά επιλεγμένο σύνολο δραστηριοτήτων που επιλέγονται για τη βέλτιστη χρήση των πόρων (χρόνος, χρήματα, άνθρωποι, υλικά, μηχανήματα, ενέργεια, χώρος κ.α.) με απώτερο σκοπό την επίτευξη των προκαθορισμένων στόχων του έργου.

Έτσι καταλήγουμε σε ένα δεύτερο ορισμό για το έργο:

Έργο είναι ένα εγχείρημα κατά το οποίο ανθρώπινοι πόροι, μηχανές, οικονομικοί 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 μια επωφελής μεταβολή η οποία ορίζεται μέσω ποσοτικών και ποιοτικών στόχων.

Παραδοσιακά, οι βασικοί περιορισμοί που αναφέρονται είναι τρεις και είναι: το αντικείμενο του έργου (project scope), χρόνος που απαιτείται για την εκτέλεση του έργου και το κόστος εκτέλεσης του έργου. Η τριάδα αυτή των περιορισμών συχνά αναφέρεται επίσης ως τρίγωνο διαχείρισης του έργου, όπου κάθε πλευρά αντιπροσωπεύει έναν περιορισμό. Αλλαγή στη μια πλευρά του τριγώνου που μεταφράζεται σε αλλαγή των περιορισμών προκαλεί αλλαγή στους περιορισμούς που σχετίζονται με τους άλλους παράγοντες. Έτσι αλλαγή στο αντικείμενο των εργασιών του έργου προκαλεί αλλαγή στους περιορισμούς του χρόνου και του κόστους π.χ. αύξηση της διάρκειας του έργου, αύξηση του προϋπολογισμού.

Στην επόμενη λίστα παρουσιάζουμε τα βασικά χαρακτηριστικά ενός έργου:

- Αποτελείται από μη επαναλαμβανόμενες δραστηριότητες οι οποίες στη γενική περίπτωση μπορούν να περιγραφούν από τον κύκλο ζωής λογισμικού.
- Απαιτείται σχεδιασμός ώστε να επιτύχουμε το τελικό αποτέλεσμα.
- Το τελικό αποτέλεσμα είναι μοναδικό.
- Η εκτέλεσή του απαιτεί την ύπαρξη ομάδας.
- Έχει έναρξη και λήξη.
- Υπόκειται σε περιορισμούς διαφόρων ειδών (χρόνου, κόστους ποιότητας κ.α.)
- Οι διαθέσιμοι πόροι είναι περιορισμένοι.
- Είναι μεγάλα και πολύπλοκα.

### 2.6.3 Παράγοντες Επιτυχίας στη Διοίκηση Έργου

Ο διαχειριστής έργου (project manager) πρέπει να επιτελεί ταυτόχρονα πολλές εργασίες. Πρέπει, επίσης, να λαμβάνονται συνεχώς αποφάσεις σε όλα τα επίπεδα σχετικά με τη χρήση πόρων, προσαρμογές του χρονοδιαγράμματος, θέματα προϋπολογισμός, διαχείριση ανθρώπινων σχέσεων, επικοινωνία, και τεχνικά προβλήματα. Πρέπει συνεπώς να προσδιοριστούν τα κύρια θέματα, στρατηγικά, τακτικά ή λειτουργικά, ώστε να οριστούν οι προτεραιότητες και να μπορέσει ο project manager να εστιάσει στα κρίσιμα θέματα, τα οποία εναλλάσσονται ανάλογα με τη φάση στην οποία βρίσκεται το έργο. Ο Balachandra (1984) έχει ορίσει τους παρακάτω 10 παράγοντες επιτυχίας:

1. Στόχος έργου: Ο ορισμός ξεκάθαρων στόχων αποτελεί κλειδί για τον προγραμματισμό και την εκτέλεση ενός έργου. Η κατανόηση των μέτρων απόδοσης και αξιολόγησης είναι σημαντική ώστε να γίνεται καλός συντονισμός. Συνεπώς, όλοι οι εμπλεκόμενοι πρέπει να είναι εξ αρχής ενήμεροι για τους στόχους του έργου.

2. Υποστήριξη από την πλευρά της διοίκησης: Ο ανταγωνισμός για πόρους σε συνδυασμό με το μεγάλο βαθμό αβεβαιότητας που διέπουν ένα έργο συχνά οδηγούν σε σύγκρουση και κρίση. Η συνεχής παρουσία της διοίκησης σε ολόκληρο το κύκλο ζωής του έργου βοηθά στην κατανόηση του στόχου του έργου και της σημασίας του. Αυτή η συνειδητοποίηση οδηγεί σε υποστήριξη η οποία μπορεί να αποδειχθεί ανεκτίμητη για την επίλυση προβλημάτων σε περιπτώσεις σύγκρουσης ή κρίσης ή όταν προκύπτει αβεβαιότητα. Συνεπώς, η ξεκάθαρη και συχνή επικοινωνία μεταξύ του project manager και της διοίκησης δρα καταλυτικά για την επιτυχία ενός έργου.

3. Προγραμματισμός έργου. Η μετατροπή του στόχου, και των μέτρων απόδοσης σε ένα εφικτό πλάνο είναι ο σύνδεσμος μεταξύ της φάσης θεωρητικός σχεδιασμός και της φάσης παραγωγής. Ένα λεπτομερειακό πλάνο που να καλύπτει θέματα τεχνικά, οικονομικά, οργανωτικά, επικοινωνίας, ελέχγου και χρονοδιαγράμματος είναι η βάση για την υλοποίηση. Ο προγραμματισμός δεν τελειώνει όταν ξεκινά η εκτέλεση καθώς οι ανάγκη για αλλαγές ή μετατροπές είναι πάγια. Ο προγραμματισμός είναι συνεπώς δυναμικός και συνεχής και συνδέει τους εναλλασσόμενους στόχους και την απόδοση με τα τελικά αποτελέσματα.

4. Συνεργασία με τον Πελάτη. Ο τελικός χρήστης του έργου είναι και ο τελικός κριτής της επιτυχίας του. Ένα έργο το οποίο τελείωσε εγκαίρως, σύμφωνα με τα επιθυμητά τεχνικά χαρακτηριστικά, και εντός προϋπολογισμού, αλλά δε χρησιμοποιήθηκε ποτέ ή σπανίως μπορεί με βεβαιότητα να θεωρηθεί αποτυχία. Στη φάση θεωρητικός σχεδιασμός είναι πολύ σημαντικό να έχουμε καλή επικοινωνία με τον πελάτη ώστε οι στόχοι που θα τεθούν να είναι πλήρως ευθυγραμμισμένοι με τις ανάγκες του πελάτη. Στις επόμενες φάσεις είναι απαραίτητη η συνεχής συνεργασία με τον πελάτη ώστε να διορθώνονται πιθανά λάθη κατά τη μετατροπή των στόχων σε μέτρα απόδοσης. Ωστόσο, λόγω εναλλασσόμενων αναγκών και συνθηκών, παρόλο που στις πρώτες φάσεις είναι χρήσιμο να υπάρχει μη δήλωση των ακριβών αναγκών του πελάτη, αυτό πιθανά παύει να ισχύει στη φάση προγραμματισμού ή υλοποίησης.

Το σύστημα configuration management αποτελεί σύνδεσμο μεταξύ υπάρχοντων πλάνων και change requests του πελάτη και της ομάδας του έργου.

5. Θέματα προσωπικού. Η ικανοποιητική επίτευξη των τεχνικών στόχων χωρίς να παραβιαστεί το χρονοδιάγραμμα ή ο προϋπολογισμός δε σημαίνει ότι το έργο είναι απολύτως επιτυχημένο, ακόμη και αν ο πελάτης είναι ευχαριστημένος. Εάν οι διάφοροι εμπλεκόμενοι στο έργο δε διατηρούν καλές σχέσεις η επιτυχία του έργου είναι αμφισβητούμενη, καθώς η καλή συνεργασία και η αφοσίωση στο έργο είναι απαραίτητες για την επιτυχία.

6. Τεχνικά θέματα. Η τεχνική κατάρτιση του προσωπικού και η εκπλήρωση των τεχνικών προδιαγραφών πρέπει να είναι από τις πρώτες έννοιες ενός project manager καθώς χωρίς αυτές ένα έργο δεν μπορεί να ολοκληρωθεί.

7. Αποδοχή από τον πελάτη. Η συνεχής consultation με τον πελάτη καθ’ όλο τον κύκλο ζωής ενός έργου αυξάνει την πιθανότητα επιτυχίας όσον αφορά την αποδοχή χρήστη. Στα τελικά στάδια της υλοποίησης, ο πελάτης πρέπει να κρίνει το έργο που προκύπτει και να αποφασίσει εάν είναι αποδεκτό ή όχι. Σε περίπτωση που το έργο δε γίνει αποδεκτό σε αυτό το στάδιο, τότε είναι αποτυχημένο.

8. Έλεγχος έργου. Η συνεχής ροή πληροφοριών σχετικά με την πραγματική πρόοδο και ο μηχανισμός ανατροφοδότησης επιτρέπουν στον project manager να αντιμετωπίζει την αβεβαιότητα. Συγκρίνοντας την πραγματική πρόοδο με τα τρέχοντα πλάνα, ο project manager μπορεί να διακρίνει παρεκκλίσεις, να αναμένει προβλήματα και να ξεκινά διορθωτικές κινήσεις. Οι όποιες παρεκκλίσεις από τα αρχικά πλάνα μπορούν να διορθωθούν όταν γίνουν αντιληπτές νωρίς.

9. Επικοινωνία. Η επιτυχής μετάβαση ανάμεσα στις φάσεις του κύκλου ζωής ενός έργου και ο καλός συντονισμός ανάμεσα στους συμμετέχοντες σε κάθε φάση απαιτεί μία συνεχή ανταλλαγή πληροφοριών. Γενικά, η επικοινωνία διευκολύνεται εάν η γραμμή της εξουσίας είναι ξεκάθαρη. Η οργανωτική δομή του έργου πρέπει να περιγράφει τα κανάλια επικοινωνίας και το είδος της πληροφορίας που πρέπει να διέρχεται από αυτά. Επιπλέον, πρέπει να υπάρχει ξεκάθαρη οδηγία σχετικά με το πόσο συχνά πρέπει αυτές οι πληροφορίες να παράγονται και να μεταδίδονται. Οι επίσημες γραμμές επικοινωνίας, καθώς και η ανεπίσημη ροή ανάμεσα στα μέλη της ομάδας συνηγορούν στην επιτυχία του έργου.

10. Επίλυση προβλημάτων. Το σύστημα ελέγχου είναι σχεδιασμένο να μπορεί να βρίσκει τις προβληματικές περιοχές και εάν είναι εφικτό να βρίσκει την πηγή τους. Επειδή η αβεβαιότητα είναι συχνή πληγή για την ολοκλήρωση των έργων, η ανάπτυξη ενός contingency plan είναι καλό προληπτικό μέτρο. Η διαθεσιμότητα προετοιμασμένων πλάνων και διαδικασιών για τη διαχείριση προβλημάτων μπορεί να μειώσει τον κόπο που μπορεί να χρειαστεί ώστε να επιλυθούν εάν δεν υπήρχαν αυτά.

Οι παραπάνω παράγοντες αποτελούν γενικές γραμμές και καθώς κάθε έργο είναι μοναδικό και οι ιδιαιτερότητες του απαιτούν διαφορετικό χειρισμό κατά περίπτωση.

### 2.6.4 Διοίκηση έργου Λογισμικού

Είναι προφανές ότι ανάλογα με το μέγεθος του έργου του λογισμικού που αναλαμβάνεται να διεκπεραιωθεί καθώς και από το μέγεθος της ομάδας / μονάδας ανάπτυξης τα μοντέλα στην πραγματικότητα εφαρμόζονται με μεγαλύτερη ευελιξία και με μεγαλύτερη παραμετρικοποίηση ώστε να γίνεται η μεγαλύτερη δυνατή προσαρμογή στην αποδοτικότητα του μοντέλου στην συγκεκριμένη υπό ανάπτυξη εφαρμογή.

Έτσι είναι φανερό ότι δεν είναι και τόσο σημαντική η αυστηρή τήρηση της δομής των αναφερόμενων μοντέλων όσο και η χρονική αλληλουχία του συνόλου των ενδιάμεσων προϊόντων που παράγονται όσο η προσαρμογή των μοντέλων στις εκάστοτε συνθήκες ανάπτυξης και στα χρησιμοποιούμενα εργαλεία.

Το παραπάνω γίνεται περισσότερο προφανές αν κάποιος λάβει υπόψη του την συνθετότητα και ποικιλότητα των εργαλείων ανάπτυξης όπως οι σύγχρονες γλώσσες προγραμματισμού, οι ψηφίδες λογισμικού, τα περιβάλλοντα συγγραφής κώδικα κλπ.

Λόγω των πολλαπλών απαιτήσεων που υπάρχουν στην ανάπτυξη λογισμικού καθίσταται ακόμα πιο απαραίτητη η διοίκηση του έργου η οποία εκτός του γενικότερου συντονισμού και οργάνωσης των (ανθρώπινων) πόρων που χρησιμοποιούνται σε ένα έργο ανάπτυξης λογισμικού, θα πρέπει να λαμβάνει μέτρα ώστε το έργο που θα αναληφθεί θα ολοκληρωθεί εντός της προθεσμίας, εντός του προβλεφθέντος προϋπολογισμού και με την ορισθείσα ποιότητα με δεδομένο ότι τόσο η χρονική προθεσμία όσο και ο προϋπολογισμός θα πρέπει να υπολογιστούν από αυτή.

Η διοίκηση έργων λογισμικού εκτελείται παράλληλα με τις εργασίες ανάπτυξης επιτελώντας έργο οργανωτικό και διαχειριστικό χωρίς να ασχολείται με αυτή καθεαυτή την ανάπτυξη.

Η διοίκηση ενός έργου λογισμικού λοιπόν θα πρέπει να μπορεί όχι μόνο να υπολογίζει αλλά και να οργανώνει σωστά τους (ανθρώπινους) πόρους που απαιτούνται και να βρίσκει τρόπους επικοινωνίας μεταξύ των διαφόρων ομάδων που μετέχουν στην ανάπτυξη αλλά επιπλέον θα πρέπει να έχει την ικανότητα να αναγνωρίσει έγκαιρα και να κάνει σωστές επιλογές όταν βρίσκεται μπροστά σε πολλές επιλογές ή σε αβεβαιότητα.

Θα πρέπει ακόμα να τονιστεί το γεγονός ότι στην ανάπτυξη έργων λογισμικού είναι απαραίτητη η συνεργασία επιστημόνων όπως ειδικοί γραφίστες και ειδικοί στον σχεδιασμό του περιβάλλοντος διεπαφής και τέλος προγραμματιστές) οι οποίοι δεν έχουν, στο μεγαλύτερο ποσοστό τους, γνώσεις λογισμικού, δεν έχουν μεγάλη επιστημονική συνοχή, συνήθως δεν βρίσκονται στον ίδιο χώρο γεγονότα που δυσκολεύουν ακόμα περισσότερο τον συντονισμό των ανθρώπινων πόρων των έργων αυτών.

Έτσι, οι άνθρωποι που στελεχώνουν την διοίκηση ενός έργου θα πρέπει να διαθέτουν εκτός των γενικών προσόντων που απαιτούνται όπως γνώσεις διοίκησης, γνώσεις λογισμικού, γνώσεις θεωριών αποφάσεων και γνώσεις επικοινωνιακών ζητημάτων και κάποιου είδους εξοικείωση με παιδαγωγικά μοντέλα και εκπαιδευτικά συστήματα και ζητήματα.[3]

## 2.7 Σύνοψη

Στο κεφάλαιο αυτό γνωρίσαμε τα πιο σημαντικά μοντέλα ανάπτυξης λογισμικού παρουσιάζοντας για κάθε μοντέλο τις εργασίες που περιλαμβάνει, την σειρά που θα πρέπει αυτές να γίνουν, υπό ποίες προϋποθέσεις καθώς και με ποια κριτήρια αυτές θα ολοκληρωθούν.

Στην πρώτη ενότητα παρουσιάστηκαν συνοπτικά τα συμβατικά μοντέλα ανάπτυξης λογισμικού όπως το μοντέλο του Καταρράκτη και δύο παραλλαγές του το μοντέλο V και IEEE, το μοντέλο Λειτουργικής Επαύξησης και το μοντέλο της Πρωτοτυποποίησης στο οποίο δόθηκε και μεγαλύτερη έμφαση. Στην δεύτερη ενότητα παρουσιάστηκαν κάποια εναλλακτικά μοντέλα όπως το Λειτουργικό μοντέλο, το μοντέλο Αυτόματου Προγραμματισμού, το Μετασχηματιστικό μοντέλο, το μοντέλο Επαναχρησιμοποίησης Λογισμικού, το Αντικειμενοστραφές μοντέλο και το Σπειροειδές μοντέλο στο οποίο δόθηκε μεγαλύτερη έμφαση.

[3] (Ian Sommerville, 2008)

Από τα μοντέλα που παρουσιάστηκαν, στην πράξη περισσότερο χρήσιμα είναι αυτά που αφήνουν ελευθερία εξειδίκευσης στις συνθήκες και στο πρόβλημα που επιλύουν και δεν περιορίζουν με αυστηρότητα τις ενέργειες και τις φάσεις που θα πρέπει να ακολουθηθούν. Έτσι στην τρίτη ενότητα θεωρήσαμε το σπειροειδές μοντέλο ως το μοντέλο το οποίο μπορεί να αποτελέσει βάση για την ανάπτυξη λογισμικού με ορισμένες παραλλαγές οι οποίες εξειδικεύονται κάθε φορά ανάλογα με το περιβάλλον ανάπτυξης, το συγκεκριμένο πρόβλημα κλπ.. Τέλος, αναπτύχθηκε συνοπτικά η αναγκαιότητα της ύπαρξης διοίκησης στα έργα ανάπτυξης λογισμικού, ο ρόλος που διαδραματίζει και παρουσιάστηκαν τα βασικά χαρακτηριστικά και οι απαιτήσεις αυτών που την στελεχώνουν.

# 3 Διαδικτυακές υπηρεσίες (Web Services)

Προτού περιγράψουμε την τεχνολογία των διαδικτυακών υπηρεσιών κάνουμε μια σύντομη εισαγωγή στην XML.

## 3.1 XML

Η XML (Extended Mark-up Language) είναι μια δηλωτική γλώσσα πού ήρθε μετά την ASN1. Κύρια διαφορά είναι ότι στην XML χρησιμοποιούνται αποκλειστικά χαρακτήρες (character based, έναντι της ASN1 πού είναι bit coded). Επιπλέον η δομή της είναι ταυτοχρόνως πιο απλή αλλά και πιο εύκαμπτη. Από άλλη οπτική γωνία, στην XML ένα κείμενο χρησιμοποιείται για την αναπαράσταση δομημένης πληροφορίας στο διαδίκτυο. Όπως και στην HTML (Hyper Text Mark-up Language - και η XML - είναι γλώσσα πού επισημαίνει επιπλέον πληροφορία αναφερόμενη στο βασικό κείμενο. Στην XML προσδιορίζουμε την πληροφορία αυτή με την βοήθεια ετικετών (tags), δηλ. με ένα αναγνωριστικό περικλειόμενο από <>. Με τις ετικέτες (tags) επιτυγχάνουμε το “markup”. Tags υπάρχουν και στην HTML, π.χ. το ζεύγος <B> </B> ή το μοναδικό <Br>, αυτά όμως είναι προκαθορισμένα και σχετίζονται μόνο με την εμφάνιση του κειμένου. Τα tags της XML είναι προσδιορίσιμα ελεύθερα από τον χρήστη, ο οποίος μπορεί να τους δίδει και όποια σημασία αυτός ορίσει. Για τούτο πρέπει και πάντα να ζευγαρώνονται, δηλ. <text1>my car is here</text1>. Το αναγνωριστικό <text1> </text1> δεν προσδιορίζει αναγκαστικά το πώς θα εμφανισθεί το περικλειόμενο κείμενο (όπως θα έκανε το <B> </B> της HTML), αλλά μπορεί π.χ. να συνδυάζεται με μία ενέργεια, όπως θα καθορίζει δικός μας κώδικας που θα επεξεργασθεί το αρχείο XML. Το ζεύγος <text1> και </text1> ορίζει ένα στοιχείο (element), του οποίου το όνομα είναι το text1 και το περιεχόμενο οτιδήποτε περικλείεται μεταξύ του <text1> (opening tag) και του </text1> (closing tag). Το επόμενο βήμα είναι η ενθυλάκωση (nesting) που δίνει την δύναμη να κατασκευασθούν στοιχεία με πολύπλοκη εσωτερική δενδρική δομή.

Η σημασία της XML έγκειται στο γεγονός ότι διατίθενται έτοιμοι τρόποι σύνδεσης της δομής των κειμένων της με τα πλέον σύγχρονα προγραμματιστικά περιβάλλοντα. Στην διαδικασία αυτή εμπλέκεται πάντα ένας parser. Ο parser κατέχει κεντρικό ρόλο

στην αντιστοίχηση και αλληλοσύνδεση της δενδρικής δομή του κειμένου XML με την αντίστοιχη (συνήθως αντικειμενοστραφή) δομή στην χρησιμοποιούμενη γλώσσα, με δεδομένο ότι το κείμενο XML είναι ένας συρμός χαρακτήρων ανταλλασσόμενος μέσω μίας επικοινωνιακής σύνδεσης, ενός αρχείου στον δίσκο, κτλ. Ο parser αποκτά τέτοια σημασία, πού θεωρείται πλέον *core technology*. Ενδεικτικά παρουσιάζουμε εν συνεχεία έναν από τους πιο δημοφιλής parsers που βασίζεται στο Simple API for XML (SAX). [4]

### 3.1.1 XML και SAX

Το Simple API for XML (SAX) προσφέρεται από τον SAX parser, τον οποίο μπορούμε να φανταστούμε σαν έναν σειριακό αναγνώστη (SAX reader) του κειμένου XML, χαρακτήρα προς χαρακτήρα. Ο SAX parser αναγνωρίζοντας τα διάφορα μέρη του XML αρχείου, δημιουργεί αντίστοιχα συμβάντα (events), στα οποία το δικό μας πρόγραμμα οφείλει να ανταποκριθεί σύμφωνα με τις επιθυμίες μας. Ο SAX parser διαβάζει την ταινία (συρμό χαρακτήρων) του XML κειμένου και ανιχνεύει και σταματά σε συγκεκριμένα σημεία. Τούτο δημιουργεί ένα συμβάν (event) προς τον δικό μας event handler. Μόλις ο event handler αντιμετωπίσει το event, συνεχίζει ο parser μέχρι το επόμενο σημείο. Στο επόμενο σχήμα, φαίνεται η σειριακή είσοδος του κειμένου XML στον SAX parser. Αυτός ελέγχεται από την εφαρμογή στην οποία δηλώνει πίσω (callbacks) συμβάντα, δηλαδή την ανίχνευση των επί μέρους μερών του υπό ανάγνωση XML. [5]

## 3.2 Εισαγωγή στα web services

Ένα web service είναι ένα κομμάτι λογισμικού (software module) το οποίο εκτελεί μια ασυνεχή εργασία ή σύνολο εργασιών το οποίο μπορεί να κληθεί πάνω από ένα δίκτυο και ιδιαίτερα το διαδίκτυο. Ο προγραμματιστής μπορεί να δημιουργεί μία εφαρμογή πελάτη (client) η οποία να καλεί μια σειρά από διαδικτυακές υπηρεσίες (web services) μέσω remote procedure calls (RPC) ή μέσω υπηρεσιών μηνυμάτων για να ολοκληρώσει μέρος ή ολόκληρο το λογικό κομμάτι της εφαρμογής. Μια δημοσιευμένη διαδικτυακή υπηρεσία περιγράφει τον εαυτό της έτσι ώστε οι

[4] (en.wikipedia.org)
[5] (en.wikipedia.org)

προγραμματιστές να μπορούν να εντοπίσουν την υπηρεσία και να αξιολογήσουν την καταλληλότητά της για τις ανάγκες τους.

Για παράδειγμα, μια εταιρία θα μπορούσε να παρέχει μια διαδικτυακή υπηρεσία στους πελάτες της η οποία να ελέγχει τον κατάλογο των απογραφέντων προϊόντων πριν γίνει μια παραγγελία. Άλλο παράδειγμα είναι η υπηρεσία Federal Express package tracking service την οποία οι πελάτες μπορούν να χρησιμοποιήσουν για να ακολουθήσουν τις αποστολές εμπορευμάτων. Οι διαδικτυακές υπηρεσίες χρησιμοποιούν το SOAP (Simple Object Access Protocol) για να μεταφέρουν μηνύματα XML και χρησιμοποιούν ως μέσο μεταφοράς των SOAP μηνυμάτων το πρωτόκολλο HTTP. Τα μηνύματα SOAP είναι ουσιαστικά έγγραφα XML τα οποία στέλνονται μεταξύ μιας διαδικτυακής υπηρεσίας και της εφαρμογής που την καλεί.

Οι διαδικτυακές υπηρεσίες μπορούν να γραφούν σε οποιαδήποτε γλώσσα προγραμματισμού και να τρέξουν σε οποιαδήποτε πλατφόρμα. Ένας πελάτης (client) μιας διαδικτυακής υπηρεσίας μπορεί επίσης να είναι γραμμένος σε οποιαδήποτε γλώσσα προγραμματισμού και να τρέχει σε οποιαδήποτε πλατφόρμα. Έτσι, για παράδειγμα, ένας πελάτης γραμμένος στη γλώσσα Delphi ο οποίος τρέχει σε Windows θα μπορούσε να καλέσει μια διαδικτυακή υπηρεσία γραμμένη σε Java η οποία να τρέχει σε Linux. [6]

### 3.2.1 Αρχιτεκτονική των web services

Η αρχιτεκτονική των διαδικτυακών υπηρεσιών επιτρέπει την ανάπτυξη τέτοιων διαδικτυακών υπηρεσιών οι οποίες ενθυλακώνουν όλα τα επίπεδα λειτουργικότητας. Με άλλα λόγια μια διαδικτυακή υπηρεσία μπορεί να είναι πολύ απλή, όπως για παράδειγμα μια υπηρεσία η οποία επιστρέφει την τρέχουσα θερμοκρασία, ή μπορεί να είναι μια πολύπλοκη εφαρμογή. Η αρχιτεκτονική τους, επιπλέον, επιτρέπει σε πολλαπλές διαδικτυακές υπηρεσίες να συνδυάζονται και να δημιουργούν νέα λειτουργικότητα.

Η αρχιτεκτονική των διαδικτυακών υπηρεσιών έχει τρεις ξεχωριστούς ρόλους: Έναν πάροχο (provider), τον καλών την υπηρεσία (requestor), και έναν μεσολαβητή (broker). Ο provider δημιουργεί τη διαδικτυακή υπηρεσία και την κάνει διαθέσιμη σε πελάτες (clients) οι οποίοι επιθυμούν να τη χρησιμοποιήσουν. Ένας requestor είναι

[6] (www.w3schools.com)

μια εφαρμογή client η οποία καταναλώνει την διαδικτυακή υπηρεσία. Η υπηρεσία η οποία καλείται μπορεί με τη σειρά της να καλεί άλλες διαδικτυακές υπηρεσίες παίζοντας το ρόλο πελάτη. Ο broker, όπως για παράδειγμα ένας κατάλογος υπηρεσιών (service registry), παρέχει έναν τρόπο ώστε να επικοινωνούν ο provider και ο requestor μιας διαδικτυακής υπηρεσίας.

Οι τρεις ρόλοι, αυτοί του provider, του requestor και του broker αλληλεπιδρούν μεταξύ τους μέσω των λειτουργιών δημοσίευσης (publish), εύρεσης (find) και σύνδεσης (bind). Ένας provider ενημερώνει τον broker για την ύπαρξη μιας διαδικτυακής υπηρεσίας χρησιμοποιώντας τη διεπαφή δημοσίευσης του broker ώστε να κάνει την υπηρεσία προσβάσιμη στους clients. Οι πληροφορίες που δημοσιεύονται περιγράφουν την υπηρεσία και καθορίζουν το πού βρίσκεται η υπηρεσία. Ο requestor συμβουλεύεται τον broker για να εντοπίσει μια δημοσιευμένη διαδικτυακή υπηρεσία. Με τις πληροφορίες που παίρνει από τον broker για την υπηρεσία, ο requestor μπορεί πλέον να συνδεθεί, ή να καλέσει τη διαδικτυακή υπηρεσία. [7]

## 3.3 Τα πρότυπα των web services

Τα πρότυπα πάνω στα οποία βασίζεται η ανάπτυξη των διαδικτυακών υπηρεσιών είναι εξελισσόμενες τεχνολογίες. Τα κυριότερα από αυτά είναι το SOAP (Simple Object Access Protocol), το WSDL (Web Services Description Language), το UDDI (Universal Description, Discovery and Integration), και το WSIL (Web Services Inspection Language).

### 3.3.1 Simple Object Access Protocol (SOAP)

Το SOAP είναι ένα transport-independent πρωτόκολλο μηνυμάτων. Κάθε μήνυμα SOAP είναι ένα έγγραφο XML. Το SOAP χρησιμοποιεί μηνύματα μίας κατεύθυνσης παρόλο που γίνεται να συνδυάσει μηνύματα σε ακολουθίες request-and-response μηνυμάτων. Ο ορισμός του SOAP καθορίζει τη μορφή του μηνύματος XML αλλά όχι το περιεχόμενό του και το πώς ακριβώς θα αποσταλεί. Το SOAP, εντούτοις, καθορίζει το πώς τα μηνύματα SOAP δρομολογούνται πάνω από HTTP.

Κάθε έγγραφο SOAP έχει ένα root <Envelope> element. Το root element, το πρώτο element μέσα σε ένα έγγραφο XML, περιέχει όλα τα άλλα elements που υπάρχουν

[7] (www.it.uom.gr)

μέσα στο έγγραφο. Μέσα στον «φάκελο» (envelope) υπάρχουν δύο μέρη: μια επικεφαλίδα (header) και ένα σώμα (body). Η επικεφαλίδα περιέχει πληροφορίες δρομολόγησης ή context data. Μπορεί να είναι κενό. Το σώμα περιέχει το κυρίως μήνυμα. Μπορεί επίσης να είναι κενό.

### 3.3.2 Web Services Description Language (WSDL)

Μια διαδικτυακή υπηρεσία είναι χρήσιμη μόνο εφόσον οποιοσδήποτε μπορεί να βρει τι κάνει και με ποιον τρόπο να την καλέσει. Οι προγραμματιστές πρέπει να γνωρίζουν επαρκείς πληροφορίες για μια διαδικτυακή υπηρεσία έτσι ώστε να μπορούν να γράψουν ένα πρόγραμμα client το οποίο να την καλεί. Το WSDL είναι μια XML-based γλώσσα η οποία χρησιμοποιείται για να ορίσει διαδικτυακές υπηρεσίες και να περιγράψει το πώς μπορεί κανείς να έχει πρόσβαση σε αυτές. Ειδικότερα, περιγράφει το πώς πρέπει να είναι τα δεδομένα και τα μηνύματα που προσφέρει η διαδικτυακή υπηρεσία. Εξετάζοντας το έγγραφο WSDL μιας διαδικτυακής υπηρεσίας, οι προγραμματιστές γνωρίζουν τις μεθόδους που είναι διαθέσιμες και το πώς να τις καλούν χρησιμοποιώντας τις κατάλληλες παραμέτρους.

### 3.3.3 Universal Description, Discovery and Integration (UDDI)

Το UDDI είναι ένα αναπτυσσόμενο πρότυπο για την περιγραφή, δημοσίευση και ανακάλυψη των διαδικτυακών υπηρεσιών που προσφέρει μία εταιρία. Πρόκειται για έναν ορισμό για έναν κατανεμημένο κατάλογο πληροφοριών για διαδικτυακές υπηρεσίες. Μόλις αναπτυχθεί μια διαδικτυακή υπηρεσία και δημιουργηθεί και ένα έγγραφο WSDL το οποίο την περιγράφει, πρέπει στη συνέχεια να υπάρχει ένας τρόπος να έρθει η πληροφορία του WSDL στα χέρια των χρηστών οι οποίοι θέλουν να χρησιμοποιήσουν τις διαδικτυακές υπηρεσίες τις οποίες αυτό περιγράφει. Μόλις μια διαδικτυακή υπηρεσία δημοσιεύεται σε έναν κατάλογο UDDI, πιθανοί χρήστες έχουν τη δυνατότητα να ψάξουν και να μάθουν για την ύπαρξη της υπηρεσίας αυτής.

Το περιεχόμενο ενός καταλόγου UDDI είναι παρόμοιο με αυτό ενός τηλεφωνικού καταλόγου. Στον τομέα των «λευκών σελίδων» του καταλόγου υπάρχει πληροφορία όπως το όνομα, η διεύθυνση και ο τηλεφωνικός αριθμός της εταιρίας η οποία παρέχει μία η περισσότερες διαδικτυακές υπηρεσίες. Ο τομέας των «κίτρινων σελίδων» καθορίζει τον τύπο της εταιρίας και την κατηγοριοποιεί ανά βιομηχανικό τομέα. Οι

«πράσινες σελίδες» παρέχουν τα δεδομένα για τις διαδικτυακές υπηρεσίες που παρέχει η εταιρία.

### 3.3.4 Web Services Inspection Language (WSIL)

Το WSIL, όπως και το UDDI, παρέχει μια μέθοδο για ανακάλυψη υπηρεσιών για τις διαδικτυακές υπηρεσίες. Αντίθετα με το UDDI, το WSIL χρησιμοποιεί ένα από-κεντροποιημένο, κατανεμημένο μοντέλο. Τα έγγραφα WSIL, τα οποία είναι βασικά δείκτες σε λίστες υπηρεσιών, επιτρέπουν στους καταναλωτές των διαδικτυακών υπηρεσιών να βρίσκουν διαθέσιμες υπηρεσίες σε ιστοσελίδες. Ο ορισμός του WSIL παρέχει πρότυπα για τη χρήση XML-formatted εγγράφων για να εξετάζουν μια ιστοσελίδα για υπηρεσίες και ένα σύνολο κανόνων για το πώς οι πληροφορίες αυτές θα γίνονται διαθέσιμες. Ένα έγγραφο WSIL συγκεντρώνει πολλαπλές αναφορές σε προ-υπάρχοντα έγγραφα περιγραφής υπηρεσιών μέσα σε ένα έγγραφο. Το έγγραφο WSIL στη συνέχεια φιλοξενείται από τον πάροχο της υπηρεσίας και έτσι οι καταναλωτές μπορούν να μάθουν για τις διαθέσιμες υπηρεσίες.

### 3.3.5 Java API for XML-based Remote Procedure Call (JAX-RPC)

Το JAX-RPC ορίζει ένα Java API το οποίο μπορούν να χρησιμοποιήσουν οι προγραμματιστές Java στις εφαρμογές τους για να αναπτύσσουν και να καταναλώνουν διαδικτυακές υπηρεσίες. Χρησιμοποιώντας το JAX-RPC, ένας Java client μπορεί να καταναλώνει μια διαδικτυακή υπηρεσία που τρέχει σε έναν απομακρυσμένο server στο διαδίκτυο, ακόμη και αν η υπηρεσία είναι γραμμένη σε διαφορετική γλώσσα προγραμματισμού και τρέχει σε διαφορετική πλατφόρμα. Μια υπηρεσία JAX-RPC μπορεί επίσης να καταναλωθεί από non-Java clients.

Το JAX-RPC χρησιμοποιεί ένα XML-messaging πρωτόκολλο, όπως είναι το SOAP, για τη μεταφορά μιας απομακρυσμένης κλήσης μιας διαδικασίας πάνω από ένα δίκτυο. Για παράδειγμα, μια διαδικτυακή υπηρεσία η οποία επιστρέφει μια αναφορά αποθεμάτων θα λάμβανε ένα SOAP HTTP request το οποίο θα περιείχε την κλήση μιας μεθόδου από τον client. Χρησιμοποιώντας JAX-RPC, η υπηρεσία εξάγει την κλήση της μεθόδου από το μήνυμα SOAP, τη μεταφράζει σε μία πραγματική κλήση της μεθόδου και την καλεί. Στη συνέχεια η υπηρεσία χρησιμοποιεί JAX-RPC για να μετατρέψει το response της μεθόδου ξανά σε SOAP και να στείλει τα αποτελέσματα

πίσω στον client. Ο client λαμβάνει το μήνυμα SOAP και χρησιμοποιεί JAX-RPC για να εξάγει το response.

Το JAX-RPC runtime δημιουργεί stubs και ties, τα οποία είναι κλάσεις που επιτρέπουν επικοινωνία μεταξύ του client και της υπηρεσίας. Ένα stub, το οποίο βρίσκεται στην πλευρά του client, είναι ένα τοπικό αντικείμενο το οποίο αντιπροσωπεύει μια απομακρυσμένη υπηρεσία και ενεργεί ως πληρεξούσιος για την υπηρεσία. Ένα tie, το οποίο είναι στην πλευρά του server, ενεργεί ως πληρεξούσιος στον server.

Μετά την μελέτη των παραπάνω κρίνετε αναγκαίο η περιγραφή της έννοιας της πολυνηματικότητας.

## 3.4 Πολυνηματικότητα (Multithreaded)

Πολυνηματικότητα (multithreading) καλείται η τεχνική δημιουργίας εφαρμογών που βασίζονται στα πολλαπλά «νήματα» (threads). Ένα νήμα είναι μία ακολουθία εντολών η οποία μπορεί να εκτελείται παράλληλα με άλλες παρόμοιες. Είναι δηλαδή ένα είδος παραλληλισμού του κώδικα, ο οποίος παρ' όλα αυτά μπορεί να τρέχει και σε έναν επεξεργαστή. Η δυσκολία του να γραφούν προγράμματα που να αντιμετωπίζουν καταστάσεις όπου πολλά πράγματα ταυτόχρονα πρέπει να συμβαίνουν, ανάγεται στη δυσκολία εξαγωγής ενός παράλληλου προγράμματος από το αντίστοιχο σειριακό, με τα αντίστοιχα προβλήματα συγχρονισμού, αδιεξόδων, αποτυχιών.

Άλλα χαρακτηριστικά της πολυνηματικότητας είναι η καλύτερη αλληλεπίδραση πραγματικού χρόνου. Σε αυτόνομα περιβάλλοντα Java, η απόκριση προσεγγίζει πολύ τις απαιτήσεις απόκρισης πραγματικού χρόνου. Αν όμως αυτό το περιβάλλον βρίσκεται κάτω από ένα διαφορετικό λειτουργικό σύστημα (π.χ. UNIX, Windows, Macintosh) η απόδοση αυτή φυσικό είναι να παρουσιάζεται μειωμένη. [8]

Πριν προχωρήσουμε στην περιγραφή των εργαλείων που χρησιμοποιήσαμε θα πρέπει να αναφέρουμε τα είδη προγραμματισμού που υπάρχουν.

[8] (Multithreading, 2007 )

## 3.5 Έιδη προγραμματισμού

### 3.5.1 Δομημένος προγραμματισμός

Στην επιστήμη υπολογιστών δομημένος προγραμματισμός (structured programming) ή διαδικαστικός προγραμματισμός (procedural programming) είναι μία προσέγγιση στον προγραμματισμό[1], η οποία βασίζεται στην έννοια της κλήσης διαδικασίας. Η διαδικασία, γνωστή επίσης και ως ρουτίνα, υπορουτίνα, μέθοδος ή συνάρτηση (δεν σχετίζεται άμεσα με τη μαθηματική έννοια της συνάρτησης), είναι απλά ένα αυτοτελές σύνολο εντολών προς εκτέλεση.

Ο δομημένος προγραμματισμός βασίζεται στην αρχή του διαίρει και βασίλευε, καθώς διασπά το βασικό πρόβλημα σε μικρότερα υποπροβλήματα (γνωστά επίσης και ως εργασίες). Κάθε εργασία με πολύπλοκη περιγραφή διαιρείται σε μικρότερες, έως ότου οι εργασίες να είναι αρκετά μικρές, περιεκτικές και εύκολες προς κατανόηση.

#### *3.5.1.1 Ιστορική διαδρομή*

Ιστορικά ο δομημένος προγραμματισμός αναπτύχθηκε ύστερα από έρευνα κατά τη δεκαετία του 1960, ως βελτίωση του ήδη υπάρχοντος διαδικαστικού προγραμματισμού. Ένα από τα πιο σημαντικά αποτελέσματα αυτής της έρευνας ήταν η ανάπτυξη της γλώσσας Pascal (γλώσσα προγραμματισμού), από τον Νίκλαους Βιρτ (Niklaus Wirth) το 1971, η οποία σύντομα έγινε η προτιμώμενη γλώσσα διδασκαλίας σε πολλά πανεπιστήμια[2]. Η έννοια της διαδικασίας επομένως ήταν προϋπάρχουσα αλλά δεν έπαιζε τόσο σημαντικό ρόλο στη δομή των υπό συγγραφή εφαρμογών, καθώς τα δεδομένα ήταν αρκετά διαχωρισμένα από τις διαδικασίες και έπρεπε ο προγραμματιστής να θυμάται για κάθε διαδικασία ποια άλλη καλούσε, αλλά και ποια δεδομένα διαφοροποιούνταν. Καθώς όμως οι περισσότερες διαδικαστικές γλώσσες γρήγορα υιοθέτησαν στοιχεία ώστε να υποστηρίζουν δομημένο προγραμματισμό, οι δύο όροι σήμερα έχουν πρακτικώς ταυτιστεί. Με τον καιρό οι δομημένες γλώσσες έφτασαν να μην επαρκούν για τη συγγραφή προγραμμάτων, επεκτάθηκαν και ως λύση υιοθετήθηκε ο αντικειμενοστραφής προγραμματισμός.

### 3.5.2 Πλεονεκτήματα

Ο δομημένος προγραμματισμός βοηθάει την ευκολότερη συγγραφή πολύπλοκων προγραμμάτων. Με τον δομημένο προγραμματισμό γίνεται πιο εύκολη και ταχύτερη η διαχείριση, η συντήρηση και η αποσφαλμάτωση, καθώς βασίζεται σε μεγαλύτερες, πολυπλοκότερες και περιεκτικότερες μονάδες, όπως οι διαδικασίες, αντί για μεμονωμένες εντολές.

### 3.5.3 Συναρτησιακός προγραμματισμός

Στην επιστήμη υπολογιστών, συναρτησιακός προγραμματισμός είναι ένα προγραμματιστικό παράδειγμα που αντιμετωπίζει τον υπολογισμό ως την αποτίμηση μαθηματικών συναρτήσεων και αποφεύγει την κατάσταση προγράμματος και τα μεταβλητά δεδομένα. Δίνει έμφαση στην εφαρμογή συναρτήσεων, σε αντίθεση με τον προστακτικό προγραμματισμό, ο οποίος δίνει έμφαση στις αλλαγές κατάστασης. Ο συναρτησιακός προγραμματισμός έχει τις ρίζες του στο λογισμό λάμδα, ένα τυπικό σύστημα που αναπτύχθηκε τη δεκαετία 1930 για τη διερεύνηση του ορισμού συναρτήσεων, της εφαρμογής συναρτήσεων και της αναδρομής. Πολλές συναρτησιακές γλώσσες προγραμματισμού μπορούν να θεωρηθούν ως επεκτάσεις του λογισμού λάμδα.

Πρακτικά, η διαφορά μεταξύ μιας μαθηματικής συνάρτησης και της έννοιας της "συνάρτησης" που χρησιμοποιείται στον προστακτικό προγραμματισμό είναι ότι οι προστακτικές συναρτήσεις μπορούν να έχουν παρενέργειες, αλλάζοντας την τιμή των ήδη αποτιμημένων υπολογισμών. Λόγω αυτού, δεν έχουν διαφάνεια αναφορικότητας, δηλαδή η ίδια έκφραση της γλώσσας μπορεί να δώσει διαφορετικές τιμές ανάλογα με την κατάσταση του εκτελούμενου προγράμματος. Αντίστροφα, σε συναρτησιακά προγράμματα, η τιμή που επιστρέφει μια συνάρτηση εξαρτάται μόνο από τα ορίσματα που αποτελούν την είσοδο της συνάρτησης. Έτσι, η κλήση μιας συνάρτησης f με ένα όρισμα x θα δώσει το ίδιο αποτέλεσμα f(x) και τις δύο φορές. Η εξάλειψη των παρενεργειών μπορεί να κάνει πολύ ευκολότερο το να κατανοηθεί και να προβλεφθεί η συμπεριφορά ενός προγράμματος. Αυτό είναι ένα από τα κίνητρα για την ανάπτυξη του συναρτησιακού προγραμματισμού.

Οι συναρτησιακές γλώσσες προγραμματισμού, και ειδικότερα οι αμιγώς συναρτησιακές συναντώνται περισσότερο στο ακαδημαϊκό παρά στο εμπορικό ή βιομηχανικό περιβάλλον. Παρ' όλα αυτά, αξιοσημείωτες συναρτησιακές γλώσσες που

χρησιμοποιούνται στη βιομηχανία και στην ανάπτυξη εμπορικών εφαρμογών είναι ανάμεσα σε άλλες η Erlang, η OCaml, η Haskell, η Scheme, αλλά και γλώσσες ειδικών πεδίων όπως η R για τη στατιστική, η Mathematica για τα συμβολικά μαθηματικά, ή η K για την χρηματιστηριακή ανάλυση. Διαδεδομένες γλώσσες ειδικού πεδίου όπως η SQL και τα Lex/Yacc χρησιμοποιούν στοιχεία συναρτησιακού προγραμματισμού, ειδικά για να αποφύγουν μεταβλητές τιμές. Τα λογιστικά φύλλα μπορούν επίσης να θεωρηθούν συναρτησιακές γλώσσες προγραμματισμού.

Προγραμματισμός σε συναρτησιακό στυλ μπορεί να επιτευχθεί και σε μη συναρτησιακές γλώσσες όπως η C, η Java, η Python, που δεν είναι ειδικά σχεδιασμένες για τέτοια χρήση.

### 3.5.4 Προστακτικός προγραμματισμός

Στην επιστήμη υπολογιστών καλούμε προστακτικό προγραμματισμό, σε αντίθεση με το δηλωτικό προγραμματισμό, ένα προγραμματιστικό υπόδειγμα όπου το ζητούμενο κατασκευάζεται / υπολογίζεται αλλάζοντας την κατάσταση του υπολογιστή μέσω εντολών.

Το υπόδειγμα αυτό ακολουθούν οι διαδικαστικές γλώσσες προγραμματισμού, όπως οι Pascal, C , Fortran κ.α., αλλά και οι αντικειμενοστρεφείς γλώσσες όπως η Java, η C++, η C# κ.α.

### 3.5.5 Δηλωτικός προγραμματισμός

Στην επιστήμη υπολογιστών δηλωτικός προγραμματισμός (declarative programming) είναι ένα προγραμματιστικό υπόδειγμα όπου, σε αντίθεση με τον προστακτικό προγραμματισμό, το ζητούμενο αποτέλεσμα υπολογίζεται περιγράφοντας απλώς τις επιθυμητές ιδιότητες του. Παραδείγματα γλωσσών δηλωτικού προγραμματισμού είναι η Haskell, η Prolog, η SQL, η HTML και η CSS.

### 3.5.6 Αντικειμενοστρεφής προγραμματισμός

Στην επιστήμη υπολογιστών αντικειμενοστρεφή προγραμματισμό (object-oriented programming), ή ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός

σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων ονόματι κλάση. Η κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους αυτή υπήρξε η πρωταρχική καινοτομία του ΑΠ.

Έτσι μπορεί να οριστεί μία προδιαγραφή δομής αποθήκευσης (π.χ. μία κλάση "τηλεόραση") η οποία να περιέχει τόσο ιδιότητες (π.χ. μία μεταβλητή "τρέχον κανάλι") όσο και πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία "άνοιγμα της τηλεόρασης"). Στο εν λόγω παράδειγμα κάθε υλική τηλεόραση (κάθε αντικείμενο αποθηκευμένο πραγματικά στη μνήμη) αναπαρίσταται ως ξεχωριστό, "φυσικό" στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Επομένως μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς "καλούπια". Οι αιτίες που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), όμως τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.

### 3.5.7 Λογικός Προγραμματισμός

Ο Λογικός Προγραμματισμός (logic programming) αποτελεί ένα σημαντικό κλάδο της Τεχνικής Νοημοσυνής που βασίζεται σε μεγάλο βαθμό στη Μαθηματική Λογική, αλλά την προεκτείνει εισάγοντας νέες μεθόδους αυτοματοποιημένης συμπερασματολογίας.

Η μελέτη του Λογικού Προγραμματισμού είναι να εξηγηθεί η λογική βάση του. Χωρίς αυτήν την μαθηματική εξήγηση δεν θα γίνει αντιληπτή η σπουδαιότητα του λογικού προγραμματισμού.

Η βάση αυτή θα φανεί χρήσιμη σε διάφορα θέματα όπως :

- στη περιγραφή ενός συστήματος (δηλαδή στην κωδικοποίηση της γνώσης) με τη μορφή ενός συνόλου λογικών προτάσεων (δηλαδή με τη μορφή ενός λογικού προγράμματος) με ποιό τρόπο κινούμαστε από την Μαθηματική Λογική στο λογικό προγραμματισμό και στη συνέχεια στην γλώσσα προγραμματισμού Prolog.

Θα μας εφοδιάσει λ.χ. με το τι μπορούν ή δεν μπορούν να εκφράσουν ή υπολογίσουν τα λογικά προγράμματα, εν συντομία, τι σημαίνουν. Επιπλέον, μελετώντας σημασιολογικά και θεωρητικά ζητήματα θα σχολιασθεί, όπου κρίνεται απαραίτητο, και η πρακτική σπουδαιότητα των υπό μελέτη θεμάτων.

Θα γνωρίσουμε τη γλώσσα της Μαθηματικής Λογικής - Προτασιακής και Κατηγορηματικής Λογικής α’ τάξης - ως μιας γλώσσας που μπορεί να χρησιμοποιηθεί γιά την επίλυση προβλημάτων.

Στη συνέχεια θα δούμε πως η λογική και κατ’ επέκταση ο λογικός προγραμματισμός δεν είναι απλά μια ακόμη γλώσσα για την επίλυση συνηθισμένων υπολογιστικών προβλημάτων.

Το κύριο σημείο εδώ είναι ότι, η λογική μπορεί να χρησιμοποιηθεί για την αναπαράσταση γνώσης που είναι συναφής με διάφορους τομείς, και μπορεί να διακινηθεί ή να αυτοματοποιηθεί με πολλούς τρόπους.

Γιά το σκοπό αυτό θα δούμε διάφορες πρακτικές εφαρμογές που θα βασίζονται στην χρήση του λογικού προγραμματισμού και ιδιαίτερα στη χρήση της γλώσσας Prolog.

Ο λογικός προγραμματισμός έχει τη δυνατότητα να εξυπηρετήσει κάθε υπολογιστικό πλαίσιο που δίνει περιθώρια για αυτοματοποιημένη συμπερασματολογία (mechanized reasoning).

Οι λόγοι για τους οποίους θα ασχοληθούμε με την κλασσική λογική είναι :

- η κλασσική λογική είναι πιο απλή και κατανοητή από τα άλλα είδη λογικής.
- μόνο αυτή έχει διερευνηθεί επαρκώς και με πληρότητα

- ως τώρα, μόνο η κλασσική λογική έχει ανταποκριθεί σε υπολογιστικούς σκοπούς με μεγάλο βαθμό επιτυχίας, και αυτό βέβαια λόγω της γλώσσας Prolog.

Βέβαια, η Κλασσική Λογική, και κατ'επέκταση ο λογικός προγραμματισμός, δεν καλύπτει τις ανάγκες αναπαράστασης της γνώσης.

Όμως αποτελεί ίσως την πιο κατάλληλη 'θύρα' για την ξενάγηση του αναγνώστη στον τεράστιο από πλευράς θεωριών και εφαρμογών κόσμο της Τεχνητής Νοημοσύνης.

Υπάρχουν δύο κύριοι τρόποι προσέγγισης του μαθηματικού περιεχομένου της λογικής :

- η θεωρία μοντέλου (model theory) και
- η θεωρία απόδειξης (proof theory).

Η θεωρία μοντέλου εξετάζει την σχέση μεταξύ λογικών προτάσεων αφού έχουν ερμηνευθεί, με την βοήθεια εξωτερικών στοιχείων, με την βοήθεια δηλαδή της απόδοσης τιμών αλήθειας.

Το λεξιλόγιο της βασικής θεωρίας μοντέλου περιλαμβάνει όρους όπως:

- αλήθεια (true),
- ψεύδος (false),
- ερμηνεία (interpretation),
- ικανοποίηση (satisfaction),
- μοντέλο (model),
- λογικό συμπέρασμα (logical consequence) ή σημασιολογική συνέπεια (semantic consequence).

Η θεωρία απόδειξης μελετά τις σχέσεις μεταξύ προτάσεων, όσον αφορά την δυνατότητα παραγωγής τους από άλλες προτάσεις, μέσω κανόνων που ενεργούν μόνο πάνω στη σύνταξη των προτάσεων αυτών.

Το λεξιλόγιο της θεωρίας απόδειξης χρησιμοποιεί όρους όπως:

- αξίωμα (axiom) ,
- κανόνας συμπερασμού (inference rule),
- θεώρημα (theorem) ,
- απόδειξη (proof),
- συνέπεια (consistency) και
- λογικό συμπέρασμα ή
- συντακτική συνέπεια (syntactic consequence).

Και οι δύο προσεγγίσεις είναι σημαντικές για την κατανόηση του λογικού προγραμματισμού, και θα δούμε ότι στην κλασσική λογική υπάρχουν απλές αλλά ουσιώδεις σχέσεις μεταξύ τους.

Η πιο ισχυρή σχέση είναι ότι τα στοιχεία που θέλουμε να αληθεύουν θα πρέπει να συμπίπτουν με εκείνα που μπορούμε να αποδείξουμε, δηλαδή, οι απαντήσεις που υπονοούνται από ένα πρόγραμμα να συμπίπτουν με τις απαντήσεις που υπολογίζονται απ' αυτό.

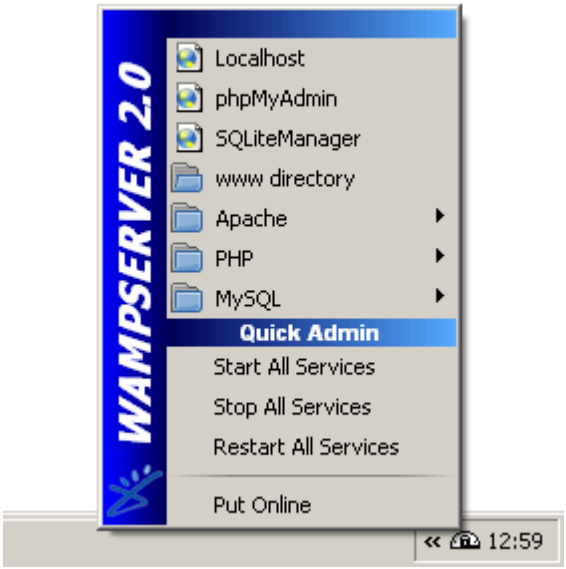
Η απαίτηση της εφαρμογής επιτάσσει σαν είδος προγραμματισμού τον διαδικτυακό αντικειμενοστραφή προγραμματισμό. Για αυτό χρειάζονται τα παρακάτω εργαλεία

## 3.6 Περιγραφή των εργαλείων που χρησιμοποιήθηκαν.

### 3.6.1 WAMP 2.0

Το wamp είναι ένα ολοκληρωμένο πακέτο Server που περιλαμβάνει apache, php, mysql, ftp server, υποστήριξη SSL και φυσικά όλα αυτά με αυτοματοποιημένη εγκατάσταση και ρύθμιση.

Στην εφαρμογή μας χρησιμοποιήσαμε την έκδοση Wamp 2.0 (Apache 2.2.11-MySQL 5.1.36 - PHP 5.3.0)




### 3.6.2 PhpMyAdmin

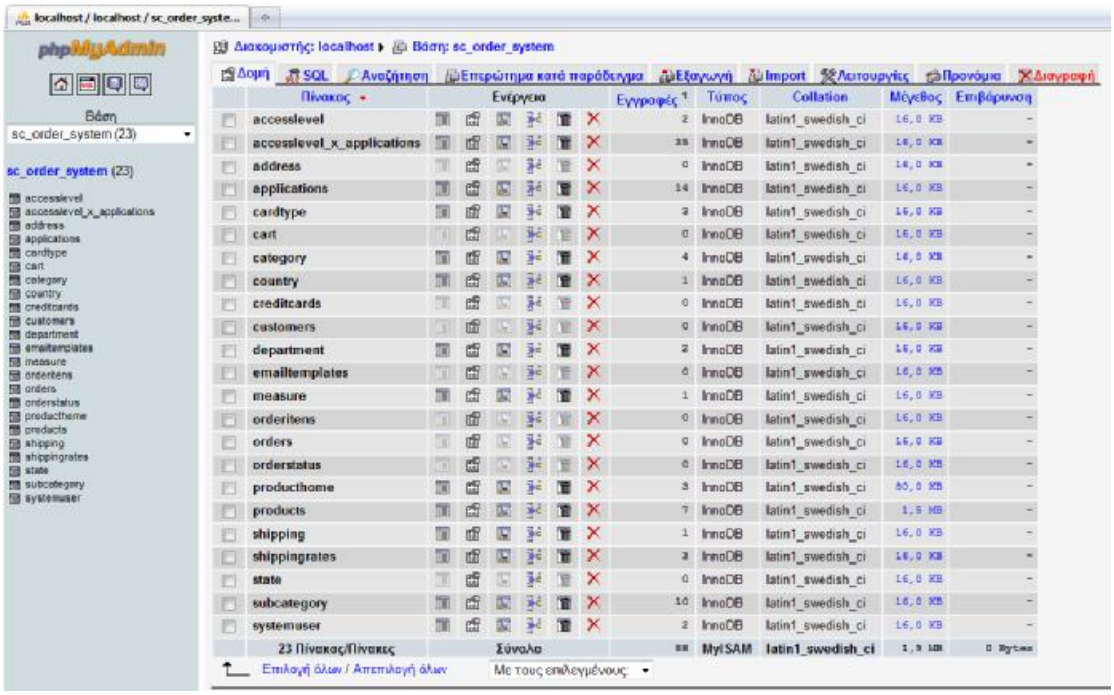
Το PhpMyAdmin είναι ένα εργαλείο γραμμένο σε Php το οποίο διαχειρίζεται την MySQL στο δίκτυο. Μπορεί να χειρίζεται πλήρως βάσεις δεδομένων, πίνακες, πεδία πινάκων αλλά και ολόκληρο τον MySQL Server. Υποστηρίζει 47 γλώσσες μεταξύ των οποίων και τα Ελληνικά και είναι λογισμικό ανοιχτού κώδικα. [9]

#### *3.6.2.1 Δυνατότητες του Php MyAdmin*

Το PhpMyAdmin μπορεί να:

[9] (Melony)

1. Δημιουργεί και να διαγράφει βάσεις δεδομένων
2. Δημιουργεί, τροποποιεί, διαγράφει, αντιγράφει και μετονομάζει πίνακες
3. Κάνει συντήρηση της βάσης
4. Προσθέτει, διαγράφει και τροποποιεί πεδία πινάκων
5. Εκτελεί Sql ερωτήματα, ακόμα και ομαδικά (batch)
6. Διαχειρίζεται κλειδιά σε πεδία
7. “Φορτώνει” αρχεία κειμένου σε πίνακες
8. Δημιουργεί και διαβάζει πίνακες (που προέρχονται από dump βάσης)
9. Εξάγει δεδομένα σε μορφή CVS, Latex, XML
10. Διαχειρίζεται πολλούς διακομιστές
11. Διαχειρίζεται τους χρήστες MySQL και τα δικαιώματά τους
12. Ελέγχει την αναφορική ακεραιότητα των δεδομένων των MyISAM πινάκων
13. Δημιουργεί PDF γραφικών του layout της βάσης δεδομένων
14. Εκτελεί αναζητήσεις σε όλη τη βάση ή μέρος αυτής
15. Υποστηρίζει πίνακες InnoDB και ξένα κλειδιά
16. Υποστηρίζει MySQLi, μια βελτιωμένη επέκταση της MySQL [10]




[10] (Melony)

### 3.6.3 JavaScript

JavaScript είναι μια νέα scripting γλώσσα, η οποία αναπτύσεται από την Netscape. Με το JavaScript μπορείς εύκολα να αναπτύξεις μια interactive σελίδα (μια σελίδα δηλαδή που αλληλεπιδρά στον εαυτό της). Αυτό το tutorial δείχνει τι μπορεί να γίνει μέσω της JavaScript - και πιο σημαντικό *πώς* μπορεί να γίνει.

#### *3.6.3.1 Τρέχοντας JavaScript*

Τι χρειάζόμαστε ώστε να τρέξουν scripts γραμμένα σε JavaScript; Χρειάζόμαστε ένα browser που υποστηρίζει JavaScript. Εφόσον αυτοί οι δύο browsers είναι πολύ διαδεδομένοι, πολλοί χρήστες έχουν την δυνατότητα να τρέξουν scripts γραμμένα σε JavaScript. Βεβαίως χρειάζεται πρώτα να κατανοήσουμε βασικές λειτουργίες της HTML..

#### *3.6.3.2 Εισάγοντας JavaScript σε μια HTML σελίδα*

Ο κώδικας JavaScript εισάγεται απευθείας στην HTML σελίδα. Για παράδειγμα:

```
<html>
<body>
<br>
Αυτό είναι ένα απλό κανονικό HTML έγγραφο.
<br>
  <script language="JavaScript">
    document.write("Αυτό είναι JavaScript!")
  </script>
<br>
Ξανά σε HTML.
</body>
</html>
```

Από πρώτη ματιά φαίνεται σαν ένα κανονικό HTML έγγραφο. Το μόνο καινούργιο μέρος είναι το:

```
  <script language="JavaScript">
    document.write("Αυτό είναι JavaScript!")
  </script>
```

Αυτό λέγεται JavaScript. Για να δούμε αυτό το script να δουλέυει, σώζουμε τον πάνω κώδικα σαν ένα απλό HTML έγγραφο και ανοίξτε το σε ένα JavaScript-enabled browser.
Εδώ είναι το αποτέλεσμα που δημιουργείται από το αρχείο (αν χρησιμοποιείτε έναν JavaScript-enabled browser πρέπει να δούμε 3 γραμμές αμέσως πιο κάτω σαν αποτέλεσμα:

*Αυτό είναι ένα απλό κανονικό HTML έγγραφο.*
*Αυτό είναι JavaScript!*
*Ξανα σε HTML.*
Πρέπει να παραδεχτούμε ότι αυτό το script δεν είναι πολύ χρήσιμο - θα μπορούσε να είχε γραφτεί καθαρή HTML. θέλαμε μόνο να σας δείξουμε την HTML εντολή <script>. Οτιδήποτε μεταξύ <script> και </script> μεταφράζεται σαν JavaScript κώδικας. Εδώ βλέπετε την χρήση του *document.write()* - μία από τις πιο σημαντικές εντολές στο JavaScript προγραμματισμό. *document.write()* χρησιμοποιείται για να γράψουμε κάτι στο πραγματικό έγγραφο (σε αυτή τη περίπτωση, το HTML έγγραφο). Έτσι το μικρό μας JavaScript πρόγραμμα γράφει *Αυτό είναι JavaScript!* στο HTML έγγραφο.

#### *3.6.3.3 Non-JavaScript browsers*

Ένας non-JavaScript browser δεν ξέρει τι σημάινει <script>. Αγνοεί το <script> και βγάζει όλο τον ακόλουθο κώδικα λες και είναι κανονικό κείμενο. Αυτό σημαίναι ότι ο χρήστης θα δει τον Javascript κώδικα μέσα στο HTML έγγραφο. Αυτό βεβαίως δεν ήταν ο σκοπός μας. Υπάρχει τρόπος για να κρύψουμε τον JavaScript κώδικα από παλιούς browsers. Θα χρησιμοποιήσουμε τα HTML comments <!-- -->. Ο νέος μας κώδικας θα δείχνει ως εξής:

```
<html>
<body>
<br>
Αυτό είναι ένα απλό κανονικό HTML έγγραφο.
<br>
 <script language="JavaScript">
 <!-- Κρύβουμε τον κώδικα
```

```
   document.write("Αυτό είναι JavaScript!")

 // -->
 </script>
<br>
Ξανά σε HTML.
</body>
</html>
```

Το αποτέλεσμα σε ένα non-JavaScript browser θα είναι το εξής:

Αυτό είναι ένα απλό κανονικό HTML έγγραφο.

Ξανά σε HTML.

Χωρίς τα HTML-comments το αποτέλεσμα σε ένα non-JavaScript browser θα ήταν:

Αυτό είναι ένα απλό κανονικό HTML έγγραφο.

document.write("Αυτό είναι JavaScript!")

Ξανά σε HTML.

Σημειώστε ότι δεν μπορείτε να κρύψετε τον JavaScript κώδικα τελείως. Αυτό που κάνουμε εδώ είναι να εμποδίσουμε το λάθος αποτέλεσμα του κώδικα σε παλιούς browsers - αλλά ο χρήστης μπορεί να δει τον κώδικα επιλέγοντας την λειτουργία 'View document source' του browser που χρησιμοποιεί. ΔΕΝ υπάρχει τρόπος να εμποδίσουμε κάποιον από το να δει τον πηγαίο κώδικα (με σκοπό να μη δει πώς γίνεται κάτι).

#### *3.6.3.4 Events - Γεγονότα*

Τα γεγονότα (Events) και οι χειριστές γεγονότων (event handlers) είναι ένα πολύ σημαντικό μέρος στον JavaScript προγραμματισμό. Τα Events προκαλούνται από τις πράξεις του χρήστη. Αν ο χρήστης πατήσει ένα κουμπί, τότε συμβαίνει ένα *Click*-event. Αν ο δείκτης του mouse κινηθεί πάνω από μια διεύθυνση (link), τότε συμβαίνει ένα *MouseOver*-event.

Υπάρχουν πολλά διαφορετικά events. Θέλουμε το javaScript πρόγραμμά μας να αντιδρά σε συγκεκριμένα events. Αυτό μπορεί να γίνει με την βοήθεια των event-handlers (χειριστές γεγονότων). Ένα κουμπί μπορεί να εμφανίζει ένα pop-up παράθυρο όταν πατιέται. Αυτό σημαίνει ότι το pop-up παράθυρο πρέπει να

εμφανιστεί σαν απάντηση στο *Click*-event. Ο event-handler που χρειαζόμαστε λέγεται *onClick*. Αυτός λέει στον υπολογιστή τί να γίνει όταν συμβεί το ανάλογο event. Ο επόμενος κώδικας δείχνει ένα απλό παράδειγμα του event-handler *onClick*:

```
<form>
<input type="button" value="Πάτησε με" onClick="alert('Καλημέρα')">
</form>
```

Υπάρχουν πολλά νέα πράγματα σε αυτόν τον κώδικα - γι' αυτό ας τα δούμε βήμα-βήμα. Μπορείτε να δείτε ότι σχεδιάζουμε μία φόρμα με ένα κουμπί (αυτό γίνεται με HTML και γι'αυτό δεν μπαίνω σε λεπτομέρειες). Το νέο μέρος είναι onClick="alert('Καλημέρα')" μέσα στο <input>. Όπως είπαμε ήδη, αυτό καθορίζει τι θα συμβεί όταν το κουμπί πατηθεί. Έτσι αν ένα *Click*-event συμβεί, ο υπολογιστής θα εκτελέσει το alert('Καλημέρα'). Αυτό είναι JavaScript κώδικας (Σημειώστε ότι δεν χρησιμοποιούμε το <script> σε αυτή τη περίπτωση).

Η εντολή *alert()* σου επιτρέπει να δημιουργείς popup παράθυρα. Μέσα στη παρένθεση πρέπει να βάλετε το κείμενο του pop-up παραθύρου. Σε αυτή τη περίπτωση είναι *'Καλημέρα'*. Έτσι το script δημιουργεί ένα παράθυρο με το κείμενο *'Καλημέρα'* όταν ο χρήστης πατήσει το κουμπί.
Ένα πράγμα που μπερδεύει λίγο: Στην εντολή document.write() χρησιμοποιήσαμε δύο άνω τελείες " και στο συνδυασμό alert() χρησιμοποιήσαμε μία άνω τελεία ' - γιατί; Βασικά μπορούμε να χρησιμοποιήσουμε και τα δύο. Όμως στο τελευταίο παράδειγμα γράψαμε onClick="alert('Καλημέρα')" - βλέπετε ότι χρησιμοποιούνται καί τα δύο σύμβολα. ’ν γράφαμε onClick="alert("Καλημέρα")" ο υπολογιστής θα μπερδευόταν μιας και δεν είναι εμφανές ποιό μέρος ανήκει στο onClick event-handler και ποιό όχι. Γι'αυτό πρέπει να εναλλαχθούν οι δύο με μία άνω τελεία σε αυτή τη περίπτωση. Δεν έχει σημασία με ποιά σειρά θα χρησιμοποιηθούν τα δύο σύμβολα. Αυτό σημαίνει ότι θα μπορούσα επίσης να γράψω onClick='alert("Καλημέρα")' και να έχω το ίδιο αποτέλεσμα.

Υπάρχουν πολλοί διαφορετικοί event-handlers που μπορούν να χρησιμοποιηθούν. Θα μάθετε μερικούς καθώς προχωράτε σε αυτό το tutorial - μα όχι όλους. Απευθύνεται σε συγκεκριμένα έγγραφα αν θέλετε να ξέρετε όλους τους event-handlers που υπάρχουν.

Αν χρησιμοποιείτε το Netscape Navigator το popup παράθυρο θα περιέχει το κείμενο *JavaScript alert* στο caption Bar (στο πάνω πάνω αριστερά μέρος του παραθύρου). Αυτός είναι περιορισμός ασφαλείας. Μπορείτε να δημιουργήσετε ένα παρόμοιο popup παράθυρο με την εντολή prompt(). Αυτό το παράθυρο δέχεται και τιμή που καθορίζει ο χρήστης για να χρησιμοποιηθεί αργότερα απ'το script. Π.χ. μπορεί αν χρησιμοποιηθεί σε ένα script που μιμείται ένα σύστημα προστασίας και να ρωτά για ένα συγκεκριμένο password. Το κείμενο στο popup παράθυρο δείχνει ότι το παράθυρο έρχεται από τον browser και όχι από το Operating system. Μιας και το Caption Bar είναι περιορισμός ασφαλείας, δεν μπορεί να αλλάξει.

### *3.6.3.5 Functions - Λειτουργίες*

Χρησιμοποιούμε λειτουργίες (functions) στα περισσότερα scripts. Γι' αυτό θα μιλήσω γι'αυτά από τώρα. Βασικά, οι λειτουργίες (fuctions) είναι ένας τρόπος για να ενσωματώνουμε πολλές εντολές μαζί. Ας γράψουμε ένα script που βγάζει τρείς φορές το ίδιο κείμενο:

```
<html>
<script language="JavaScript">
<!-- Κρύβουμε το πηγαίο κώδικα

document.write("Καλωσήρθατε στη σελίδα!<br>");
document.write("Αυτό είναι JavaScript!<br>");

document.write("Καλωσήρθατε στη σελίδα!<br>");
document.write("Αυτό είναι JavaScript!<br>");

document.write("Καλωσήρθατε στη σελίδα!<br>");
document.write("Αυτό είναι JavaScript!<br>");

// -->
</script>
</html>
```

Αυτό θα γράψει το κείμενο

Καλωσήρθατε στη σελίδα!

Αυτό είναι JavaScript!

τρείς φορές. Κοιτάχτε τον πηγιαίο κώδικα - γράφοντας τον κώδικα τρεις φορές βγάζει το σωστό αποτέλεσμα. Αλλά είναι αυτό πολύ αποτελεσματικό; Όχι, μπορούμε να το κάνουμε καλύτερα. Τι θα λέγατε γι'αυτό τον κώδικα που φέρνει το ίδιο αποτέλεσμα:

```
<html>
<script language="JavaScript">
<!-- hide

function myFunction() {
  document.write("Καλωσήρθατε στη σελίδα!<br>");
  document.write("Αυτό είναι JavaScript!<br>");
}

myFunction();
myFunction();
myFunction();

// -->
</script>
</html>
```

Σε αυτό το script προσδιορίζουμε μία function. Αυτό γίνεται από τις γραμμές:

```
function myFunction() {
  document.write("καλωσήρθατε στη σελίδα!<br>");
  document.write("Αυτό είναι JavaScript!<br>");
}
```

Οι εντολές μέσα στα { } ανήκουν στην λειτουργία *myFunction()* που δημιούργησα. Αυτό σημαίνει ότι τα δύο document.write() ενοποιούνται, γίνονται ο ορισμός της function και μπορούν να εκτελεστούν αν καλέσουμε την ανάλογη function. Στο παράδειγμά μας, έχουμε τρείς κλίσεις του function *myFuction*. Βλέπετε ότι γράφουμε τρείς φορές *myFunction()* ακριβώς κάτω από τον ορισμό της function. Αυτό σημαίνει ότι το περιεχόμενο της function εκτελείται τρείς φορές.

Αυτό είναι ένα πολύ εύκολο παράδειγμα μιας function. Ίσως αναρωτιέστε γιατί οι functions είναι τόσο σημαντικές. Καθώς διαβάζετε αυτό το tutorial θα

συνειδητοποιήσετε τα πλεονεκτήματα μιας function. Ειδικά η ρύθμιση μεταβλητών κάνει τα scripts πιο ένυκαμπτα - θα καταλάβετε τι είναι αυτό αργότερα.

Οι function μπορούν επίσης να χρησιμοποιηθούν σε συνδυασμό με event-handlers. Κοιτάξτε το παρακάτω παράδειγμα:

```
<html>
<head>

<script language="JavaScript">
<!-- Κρύβουμε τον κώδικα

function calculation() {
  var x= 12;
  var y= 5;

  var result= x + y;

  alert(result);
}

// -->
</script>

</head>
<body>

<form>
<input type="button" value="Calculate" onClick="calculation()">
</form>

</body>
</html>
```

Εδώ μπορείτε να δοκιμάσετε το παράδειγμα:

Το κουμπί καλεί την function *calculation()*. Βλέπετε ότι η function αυτή κάνει συγκεκριμένους υπολογισμούς. Για αυτό χρησιμοποιούμε τις μεταβλητές *x*, *y* και *result*. Μπορούμε να καθορίσουμε την τιμή της μεταβλητής χρησιμοποιώντας την εντολή *var*. Οι μεταβλητές μπορούν να χρησιμοποιηθούν για να αποθηκεύσουν διαφορετικού είδους τιμές - όπως νούμερα, κείμενο, κλπ. Η γραμμή var result= x + y; λέει στον browser να δημιουργήσει μια μεταβλητή *result* και να αποθηκέυσει σ'αυτή το αποτέλεσμα της πράξης *x* + *y* (δηλαδή 5 + 12). Μετά από αυτό, η μεταβλητή *result* παίρνει την τιμή *17*. Η εντολή alert (result) είναι σε αυτή την περίπτωση ίδια με την εντολή alert (17). Αυτό σημαίνει ότι το αποτέλεσμα είναι ένα popup παράθυρο με την τιμή 17 πάνω του. (www.users.hol.gr)

### 3.6.4 Scriptcase 4

Το ScriptCase είναι μια ολοκληρωμένη γεννήτρια κώδικα. Με φιλικό περιβάλλον, το ScriptCase δημιουργεί τις εφαρμογές Ιστού εξαιρετικά γρήγορα με ποιότητα και ολοκληρωμένες, κερδίζοντας χρόνο, χαμηλώνοντας τις δαπάνες και αυξάνοντας την παραγωγικότητα. Το Scriptcase υποστηρίζει τις περισσότερες χρησιμοποιημένες βάσεις δεδομένων, όπως τη Oracle, DB2, MS SQLServer, MySQL, PostgreSQL, Sybase, MS Access καθώς και άλλες.

Ο πηγαίος κώδικας της εφαρμογής είναι PHP και JavaScript και χρησιμοποιεί την τεχνολογία AJAX. Οι εφαρμογές τρέχουν απολύτως ανεξάρτητα από το εργαλείο και είναι συμβατές με τα Windows, τη Unix, AS/400 και άλλα συστήματα. (Scriptcase)[11]


### 3.6.5 AJAX

Η AJAX δεν είναι μια καινούρια γλώσσα, είναι μια διαφορετική χρησιμοποίηση τεχνολογιών που υπάρχουν εδώ και καιρό. Με τη τεχνολογία AJAX μπορείτε να δημιουργήσετε καλύτερο, γρηγορότερο και πιο φιλικές διαδικτυακές εφαρμογές για το τελικό χρήστη. Η τεχνολογία AJAX βασίζεται στη JavaScript και σε κλήσεις

[11] (Scriptcase)

HTTP. Το πρώτο λοιπόν πράμα που θα μάθουμε είναι από πού έχει προκύψει ο όρος AJAX (προφέρεται ΑΓΙΑΞ), έχουμε και λέμε λοιπόν στα αγγλικά τα αρχικά AJAX = Asynchronous JavaScript And XML η στα ελληνικά ***AJAX*** = *Ασύγχρονη* ***JavaScript*** *και* ***XML*** Τι είναι το Ajax;

Η τεχνολογία AJAX δεν είναι μια νέα γλώσσα προγραμματισμού, αλλά απλά μία νέα τεχνική για τη δημιουργία πιο γρήγορων και πιο φιλικών για το χρήστη διαδικτυακών εφαρμογών. Η τεχνολογία AJAX χρησιμοποιεί JavaScript για την αποστολή και λήψη δεδομένων μεταξύ ενός web browser (φυλλομετρητή, πλοηγό διαδικτύου) και τον εξυπηρετητή (web server). Η τεχνική AJAX κάνει της ιστοσελίδες πιο διαδραστικές επιτρέποντας την αποστολή δεδομένων πίσω από τις σκηνές χωρίς να χρειαστεί ο χρήστης να μεταφορτώνει μία ιστοσελίδα κάθε φορά που ο χρήστης έκανε κάτι στην σελίδα.


### *3.6.5.1 Η τεχνολογία AJAX είναι μια τεχνολογία που τρέχει στους υπολογιστές μας*

Η τεχνολογία AJAX τρέχει στους υπολογιστές που υπάρχει ο φυλλομετρητής. Χρησιμοποιεί ασύγχρονη μεταφορά δεδομένων (HTTP κλήσεις) μεταξύ του φυλλομετρητή και του κεντρικού εξυπηρετητή, επιτρέποντας της σελίδες web να ζητούν μικρές πληροφορίες από τον εξυπηρετητή αντί για πλήρες σελίδες. Η τεχνολογία AJAX είναι πολύ χρήσιμη μιας και επιτρέπει της διαδικτυακές εφαρμογές να είναι μικρές, να φορτώνονται γρήγορα και να είναι πολύ φιλικές για το τελικό χρήστη. Θα πρέπει να σημειωθεί πως η τεχνολογία AJAX είναι κυρίως τεχνολογία που υποστηρίζεται στο φυλλομετρητή (web browser) και όχι τεχνολογία του εξυπηρετητή web server.

### *3.6.5.2 Η τεχνολογία AJAX βασίζεται σε ανοικτά πρότυπα*

Η τεχνολογία AJAX βασίζεται σε ανοικτά πρότυπα και αυτό το κάνει πολύ εύχρηστο και ελκυστικό για εταιρείες ανάπτυξης λογισμικού. Μερικές από τις τεχνολογίες του AJAX είναι:

- JavaScript

- XML
- HTML
- CSS

Τα πρότυπα που προαναφέρουμε πως χρησιμοποιούνται από τη τεχνολογία AJAX είναι πλήρως καθορισμένα, και υποστηρίζονται πλήρως από τους πιο γνωστούς φυλλομετρητές. Η εφαρμογές AJAX είναι επίσης γνωστές ως Cross-Platform και Cross-Browser δηλαδή τεχνολογία που τρέχει σε όλες τις πλατφόρμες (λειτουργικά συστήματα) και σε όλους τους φυλλομετρητές. (www.coder.gr)

### 3.6.6 Macromedia Dreamweaver 8

Ο Dreamweaver MX 2004 αποτελεί το κορυφαίο πρόγραμμα δημιουργίας website και διαδικτυακών εφαρμογών, προσφέρει δυνατότητες δημιουργίας προηγμένων γραφικών, οπτικά εργαλεία περιγράμματος, χαρακτηριστικά ανάπτυξης εφαρμογών και υποστήριξη επεξεργασίας κώδικα μέσω ενός περιβάλλοντος τεχνολογίας ανεξαρτήτου πλατφόρμας.
Με το εργαλείο αυτό δημιουργήσαμε τη homepage της σελίδας μας καθώς και ρυθμίσαμε τα μεγέθη των διαφόρων πεδίων ώστε να πετύχουμε ένα πιο ομοιόμορφο οπτικό αποτέλεσμα. (Dreamweaver)

# 4 Μοντελοποίηση

## 4.1 Εισαγωγικά

Η ανάπτυξη ενός μοντέλου για ένα σύστημα λογισμικού που θα μπορέσει να χρησιμοποιηθεί στη βιομηχανία, προτού δομηθεί ή επανασχεδιαστεί το σύστημα, είναι τόσο απαραίτητο όσο το να υπάρχουν τα αρχιτεκτονικά σχέδια για ένα μεγάλο κτίσμα. Τα καλά μοντέλα είναι απαραίτητα για την επικοινωνία ανάμεσα στις ομάδες ενός έργου και επίσης για να υπάρχει μία ορθή αρχιτεκτονική του συστήματος. Δημιουργούμε μοντέλα των σύνθετων συστημάτων γιατί δεν μπορούμε να τα κατανοήσουμε στην ολότητά τους. Όσο αυξάνει η πολυπλοκότητα των συστημάτων, αυξάνει και η σημασία της ύπαρξης καλών τεχνικών μοντελοποίησης. Υπάρχουν και άλλοι παράγοντες για την επιτυχία ενός έργου, αλλά η ύπαρξη μίας πρότυπης, πλούσιας γλώσσας μοντελοποίησης είναι απαραίτητος παράγοντας. Μία γλώσσα μοντελοποίησης πρέπει να περιλαμβάνει:

- Στοιχεία μοντέλων – στοιχειώδεις έννοιες μοντελοποίησης και τη σημασιολογία τους.
- Συμβολισμό – οπτικές αναπαραστάσεις των στοιχείων μοντελοποίησης.
- Οδηγίες – τρόπους χρήσης στην πράξη.

Στην περίπτωση των όλο και πιο πολύπλοκων συστημάτων η οπτική αναπαράσταση και η μοντελοποίηση γίνονται απαραίτητες.

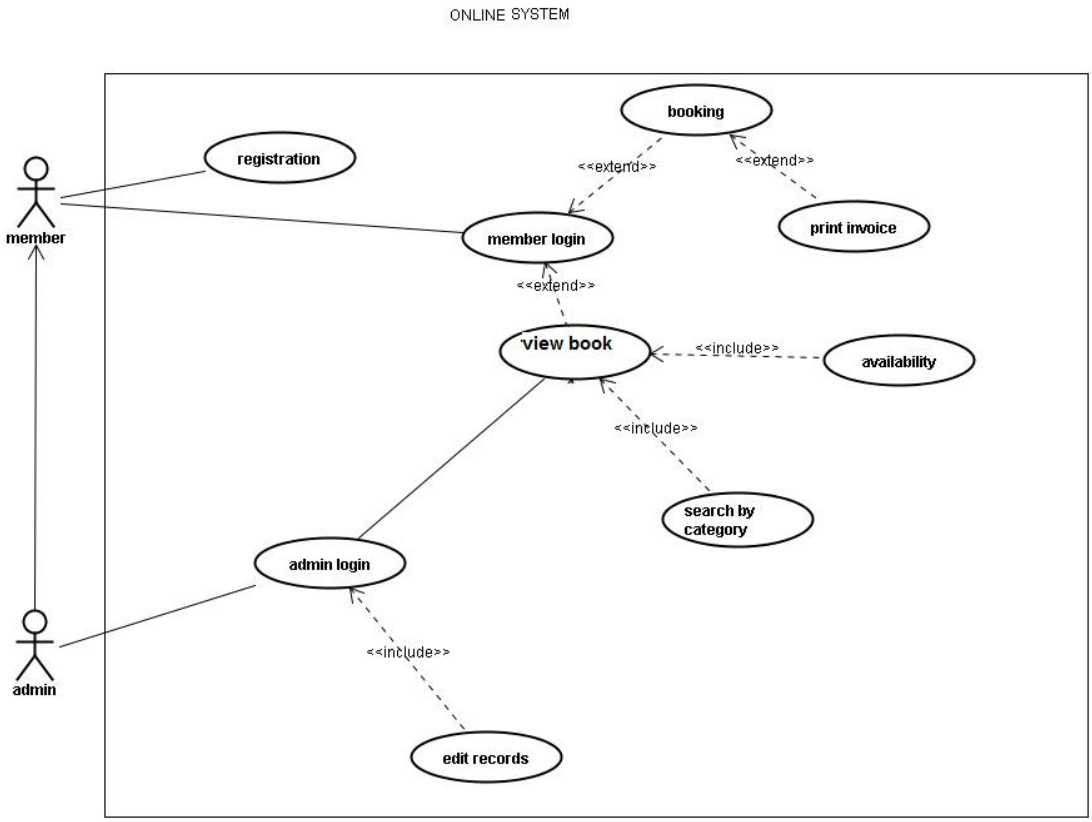
## 4.2 Use Case Diagram

Τα διαγράμματα περιπτώσεων χρήσης παρουσιάζουν τους actors και τις περιπτώσεις χρήσης ενός συστήματος μαζί με τις σχέσεις μεταξύ τους. Οι περιπτώσεις χρήσης αναπαριστούν λειτουργικότητα ενός συστήματος ή ενός classifier, όπως ένα υποσύστημα ή μία κλάση, όπως παρουσιάζεται σε όσους αλληλεπιδρούν με το σύστημα ή τον classifier, ενώ βρίσκονται εκτός του.

Μία *περίπτωση χρήσης* (use case) είναι ένα είδος classifier που αναπαριστά μία συνεπή μονάδα λειτουργικότητας που παρέχεται από το σύστημα, ένα υποσύστημα, ή μία κλάση, όπως παρουσιάζεται από ακολουθίες μηνυμάτων που ανταλλάσσονται ανάμεσα στο σύστημα και έναν ή περισσότερους εξωτερικούς χρήστες (οι οποίοι ονομάζονται *actors*) μαζί με τις πράξεις που θα πραγματοποιηθούν από το σύστημα.

Ένα *σημείο επέκτασης* (extension point) είναι μία αναφορά σε μία περιοχή μέσα στην περίπτωση χρήσης στην οποία μπορούν να εισαχθούν ακολουθίες πράξεων από άλλες περιπτώσεις χρήσης. Κάθε σημείο επέκτασης έχει ένα μοναδικό όνομα μέσα στην περίπτωση χρήσης και μία περιγραφή της περιοχής μέσα στη συμπεριφορά της περίπτωσης χρήσης.

### 4.2.1 Διάγραμμα



# 5 Βάσεις Δεδομένων

Πριν προχωρήσουμε στην έννοια της Βάσης Δεδομένων πρέπει να διακρίνουμε μεταξύ των εννοιών δεδομένα (data) και πληροφορία (information). Από τα πρώτα χρόνια της χρήσης των Ηλεκτρονικών Υπολογιστών οι έννοιες δεδομένα και πληροφορία συγχέονταν μεταξύ τους. Έτσι τα δεδομένα μπορούν να είναι λέξεις π.χ. {“Νίκος”, “Μιχάλης”, “Μαρία”, “Θάλασσα”, “Αυτοκίνητο”}, αριθμοί, π.χ. {1, 2, 3, 5, 78}, συμβολοσειρές (strings) π.χ. {“Κώστας”, “5621”, “TP–882”, “6&5 #1”, “<>+–/*”}, ή σύμβολα που έχουν νόημα σε συγκεκριμένο πλαίσιο αναφοράς. Ένα τέτοιο παράδειγμα, δεδομένων είναι ψηφιοποιημένες εικόνες με κάποια καθορισμένη τυποποίηση. Αυτές οι εικόνες θα έχουν νόημα μόνο με τη χρήση ενός συγκεκριμένου προγράμματος απεικόνισης, δηλαδή μέσα στο πλαίσιο το προγράμματος απεικόνισης, ενώ έξω από αυτό είναι ακατανόητες συλλογές συμβόλων.

Έτσι όπως ορίσαμε τα δεδομένα είναι φανερό ότι είναι κατάλληλα για αποθήκευση σε ηλεκτρονικό υπολογιστή. Οι άνθρωποι, όμως, χρησιμοποιούσαν δεδομένα, πολύ πριν από την ανακάλυψη των ηλεκτρονικών υπολογιστών, στα πλαίσια ενός πεδίου αναφοράς για την εξαγωγή συμπερασμάτων ή την λήψη αποφάσεων. Για παράδειγμα, από τα δεδομένα {85, 210, 515} και τα δεδομένα {“Κόρινθος”, “Πάτρα”, “Θεσσαλονίκη”}, αλλά και από τη γνώση (πεδίο αναφοράς) ότι τα πρώτα δεδομένα περιγράφουν αποστάσεις από την Αθήνα για τις πόλεις που ορίζονται στα δεύτερα δεδομένα, κάποιος μπορεί να καταλάβει ότι «η Πάτρα είναι πιο μακριά από την Αθήνα από ότι η Κόρινθος, αλλά πολύ κοντύτερα στην Αθήνα από ότι η Θεσσαλονίκη». Αυτό το συμπέρασμα είναι πληροφορία.

## 5.1 Βάση δεδομένων

Ένα βασικό χαρακτηριστικό των δεδομένων, έτσι όπως τα παρουσιάσαμε είναι ότι είναι κατάλληλα για να αποθηκευτούν σε ηλεκτρονικό υπολογιστή. Αυτό οδηγεί στο επόμενο βήμα που είναι η δημιουργία και χρήση Βάσεων Δεδομένων (databases).

Η σχετικότητα των δεδομένων είναι σημαντική για τον ορισμό μιας Βάσης Δεδομένων. Δεδομένα που δε σχετίζονται μεταξύ τους και απλά έχουν αποθηκευτεί σε έναν ηλεκτρονικό υπολογιστή δεν αποτελούν μια Βάση Δεδομένων. Μια Βάση Δεδομένων πρέπει να αντικατοπτρίζει ένα περιβάλλον του πραγματικού κόσμου. Τα δεδομένα που αποθηκεύονται στη Βάση Δεδομένων πρέπει να έχουν λογική συνέχεια

και νόημα. Επίσης οι λειτουργίες που παρέχει η Βάση Δεδομένων είναι σημαντικές για τον ορισμό μιας Βάσης Δεδομένων. Συλλογές δεδομένων χωρίς τη δυνατότητα ενός αυτόματου, κοινού και κεντρικού τρόπου χειρισμού των δεδομένων αυτών, δεν αποτελούν Βάση Δεδομένων. Απλά είναι δεδομένα αποθηκευμένα στον ηλεκτρονικό υπολογιστή.

## 5.2 Σύστημα Διαχείρισης Βάσεις Δεδομένων.

Η εξέλιξη των Βάσεων Δεδομένων και οι ανάγκες για δημιουργία όλο και περισσότερων Βάσεων Δεδομένων, οδήγησαν στη δημιουργία των Συστημάτων Διαχείρισης Βάσεων Δεδομένων (Database Management Systems ή DBMS). Το Σύστημα Διαχείρισης Βάσεων Δεδομένων είναι ένα εργαλείο το οποίο διευκολύνει τους χρήστες να εργάζονται με Βάσεις Δεδομένων. Με τη χρήση του Συστήματος Διαχείρισης Βάσεων Δεδομένων οι χρήστες μπορούν να κατασκευάσουν και να χρησιμοποιήσουν Βάσεις Δεδομένων. Ακολουθεί ο ορισμός του Συστήματος Διαχείρισης Βάσεων Δεδομένων.

Ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων κατά κανόνα «φιλοξενεί» πολλές Βάσεις Δεδομένων που έχουν κατασκευαστεί από διαφορετικούς χρήστες. Οι δυνατότητες που παρέχει ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων στους χρήστες συνοψίζονται στις παρακάτω:

• Ορισμός της Βάσης Δεδομένων
• Κατασκευή της Βάσης Δεδομένων
• Διαγραφή της Βάσης Δεδομένων
• Χρήση της Βάσης Δεδομένων

Στον **ορισμό** της Βάσης Δεδομένων ο χρήστης μπορεί να καθορίσει το μοντέλο της Βάσης Δεδομένων, να ορίσει τους τύπους δεδομένων που θα χρησιμοποιήσει και να ελέγξει τη Βάση Δεδομένων χωρίς να προχωρήσει σε κατασκευή της. Στην **κατασκευή** της Βάσης Δεδομένων οι τύποι των δεδομένων και τα δεδομένα αποθηκεύονται στο υλικό (hardware) του ηλεκτρονικού υπολογιστή με διαδικασίες που ελέγχονται από το Σύστημα Διαχείρισης Βάσεων Δεδομένων και δεν απασχολούν τον χρήστη. Στη **διαγραφή** της Βάσης Δεδομένων ο χρήστης αποφασίζει τον τερματισμό μιας Βάσης Δεδομένων και την απομάκρυνση των δεδομένων από το υλικό. Τέλος, κατά τη **χρήση** της Βάσης Δεδομένων ο χρήστης, είτε χειρίζεται τα

δεδομένα (εισάγει νέα δεδομένα, τροποποιεί δεδομένα, ή διαγράφει δεδομένα), είτε υποβάλει ερωτήσεις στη Βάση Δεδομένων με στόχο την εξαγωγή πληροφοριών.

## 5.3 Διαφορά βάσης δεδομένων με σύστημα διαχείρισης βάσεων δεδομένων

Σε αυτό το σημείο πρέπει να είναι ξεκάθαρη σε εσάς η διαφορά μεταξύ ενός Συστήματος Διαχείρισης Βάσεων Δεδομένων και μίας Βάσης Δεδομένων. Το Σύστημα Διαχείρισης Βάσεων Δεδομένων είναι λογισμικό το οποίο διευκολύνει τους χρήστες να υλοποιήσουν Βάσεις Δεδομένων. Αντίθετα η Βάση Δεδομένων υλοποιείται με τη βοήθεια ενός Συστήματος Διαχείρισης Βάσεων Δεδομένων που χειρίζεται δεδομένα τα οποία αποθηκεύονται στο υλικό του υπολογιστή. Η όλη διαδικασία χειρισμού και αποθήκευσης των δεδομένων διευκολύνεται από το Σύστημα Διαχείρισης Βάσεων Δεδομένων, το οποίο αναλαμβάνει τη μετατροπή των εντολών του χρήστη σε εντολές προς τον υπολογιστή και λειτουργίες που σχετίζονται με το χειρισμό των δεδομένων στο υλικό του υπολογιστή.

Ο χρήστης μιας Βάσης Δεδομένων επιτελεί λειτουργίες που μπορούν να ομαδοποιηθούν στις παρακάτω: εισαγωγή δεδομένων, διαγραφή δεδομένων, αλλαγή δεδομένων και ανάκληση δεδομένων. Όλες οι παραπάνω λειτουργίες προϋποθέτουν επικοινωνία με το υλικό του υπολογιστή (π.χ. το μαγνητικό δίσκο του υπολογιστή). Αυτή η επικοινωνία γίνεται μέσω του λογισμικού του Συστήματος Διαχείρισης Βάσεων Δεδομένων. Η επικοινωνία αυτή συνήθως δεν είναι ορατή από τον χρήστη της Βάσης Δεδομένων, ο οποίος απλά βλέπει τα αποτελέσματα από τις λειτουργίες που ζήτησε να επιτελούνται. Αυτό είναι και το σημείο που κάνει αρκετούς χρήστες να συγχέουν τις έννοιες Βάση Δεδομένων και Σύστημα Διαχείρισης Βάσεων Δεδομένων.

## 5.4 Γνωστά Συστήματα βάσεων Δεδομένων

**Εμπορικά**

- Oracle
- IBM/DB2
- MS SQL-server
- Sybase
- Informix
- (MS Access, ...)

**Ελεύθερο Λογισμικό- Open Source**

- Postgres (UCB)
- mySQL, mSQL
- miniBase (Wisc)
- Predator (Cornell)

Εμείς θα χρησιμοποιήσουμε την mysql

## 5.5 Ιστορία της SQL

Ο Dr. E. F. Codd δημοσίευσε την εργασία, «*Ένα Σχεσιακό Μοντέλο Δεδομένων για Μεγάλες Κοινές Τράπεζες Δεδομένων*», τον Ιούνιο του 1970 στο περιοδικό της Ένωσης Υπολογιστών και Περιφερειακών Συστημάτων ( Association of Computer Machinery-ACM ), *Communications of the ACM*. Το μοντέλο αυτό αποτελεί σήμερα το οριστικό πρότυπο για τα Relational DBMS. Η γλώσσα, **Structured English Query Language** ( "SEQUEL" ) (Δομημένη Αγγλική Γλώσσα Διατύπωσης Ερωτήσεων ) αναπτύχθηκε από την IBM A.E., για να χρησιμοποιήσει το μοντέλο του Codd. Η SEQUEL αργότερα μετονομάστηκε σε **Structured Query Language** ( SQL ) ( Δομημένη Γλώσσα Διατύπωσης Ερωτήσεων ). Το 1979, η Relational Software A.E., ( γνωστή ως Oracle ) εισήγαγε την πρώτη εμπορικά διαθέσιμη εφαρμογή της SQL. Σήμερα, η SQL είναι αποδεκτή ως η τυποποιημένη γλώσσα για DBMS. Ο Dr. E. F. Codd καθόρισε 13 κανόνες, που παραδόξως αναφέρθηκαν ως οι 12 κανόνες του Codd, για το σχεσιακό μοντέλο :

Ένα RDBMS πρέπει να είναι ικανό να διαχειρίζεται βάσεις δεδομένων εξ ολοκλήρου μέσω των σχεσιακών ικανοτήτων του.

1. Κανόνας πληροφοριών : Όλες οι πληροφορίες σε μια βάση δεδομένων ( συμπεριλαμβανομένου τα ονόματα των πινάκων και των στηλών ) αντιπροσωπεύονται ρητά ως τιμές σε πίνακες.
2. Εγγυημένη πρόσβαση : Κάθε τιμή σε μια σχεσιακή βάση δεδομένων είναι εγγυημένο ότι θα είναι προσβάσιμη, χρησιμοποιώντας ένα συνδυασμό του ονόματος του πίνακα, της τιμής του κυρίους κλειδιού και του ονόματος της στήλης.

3. Συστηματική υποστήριξη της τιμής null : Τα DBMS παρέχουν συστηματική υποστήριξη για την επεξεργασία null τιμών ( άγνωστα ή μη εφαρμόσιμα δεδομένα ), ευδιάκριτη από τις προκαθορισμένες τιμές και ανεξάρτητα από οποιαδήποτε περιοχή.
4. Ενεργός, online σχεσιακός κατάλογος : Η περιγραφή της βάσεως και των περιεχομένων της αντιπροσωπεύονται στο λογικό επίπεδο ως πίνακες και για αυτό μπορούν να ανακτηθούν χρησιμοποιώντας τη γλώσσα των βάσεων δεδομένων.
5. Περιεκτική υπογλώσσα δεδομένων : Μια υποστηριζόμενη γλώσσα πρέπει τουλάχιστο, να έχει ένα καλά καθορισμένο συντακτικό και να είναι περιεκτική. Πρέπει να υποστηρίζει καθορισμό δεδομένων, χειρισμό, κανόνες ακεραιότητας, έγκριση και συναλλαγές.
6. Κανόνας ανανέωσης όψης : Όλες οι όψεις που θεωρητικά είναι ανανεώσιμες μπορούν να ανανεωθούν μέσω του συστήματος.
7. Επίπεδο συνόλου εισαγωγή, ανανέωση και διαγραφή : Τα DBMS δεν υποστηρίζουν μόνο ανάκτηση σε επίπεδο συνόλου αλλά και εισαγωγή, ανανέωση και διαγραφή σε επίπεδο συνόλου.
8. Φυσική ανεξαρτησία δεδομένων : Προγράμματα εφαρμογής και ειδικά προγράμματα είναι λογικά απρόσβλητα όταν αλλάζουν οι φυσικές μέθοδοι προσπέλασης ή οι δομές αποθήκευσης.
9. Λογική ανεξαρτησία δεδομένων : Τα προγράμματα εφαρμογής και τα ειδικά προγράμματα είναι λογικά απρόσβλητα, στο μέτρο του δυνατού, όταν γίνονται αλλαγές στη δομή των πινάκων.
10. Ανεξαρτησία ακεραιότητας : Η γλώσσα της βάσεως πρέπει να είναι ικανή να καθορίζει κανόνες ακεραιότητας.
11. Ανεξαρτησία διανομής : Τα προγράμματα εφαρμογής και τα ειδικά προγράμματα είναι λογικά απρόσβλητα όταν αρχικά διανέμονται τα δεδομένα ή όταν ανακατανέμονται.
12. Καμιά ανατροπή : Δεν πρέπει αν είναι δυνατό να παρακαμφθούν οι κανόνες ακεραιότητας που ορίζονται από τη γλώσσα της βάσεως ακόμη και όταν χρησιμοποιηθεί χαμηλού επιπέδου γλώσσα. [12]

[12] (Sql)

## 5.6 Πώς λειτουργεί η SQL

Οι δυνάμεις στης SQL, παρέχουν οφέλη προς όλους τους τύπους των χρηστών, συμπεριλαμβανομένων των προγραμματιστών εφαρμογών, των διαχειριστών βάσεων δεδομένων, των διευθυντών και των τελικών χρηστών. Από τεχνικής απόψεως, η SQL είναι μια υπογλώσσα δεδομένων και είναι η δεδομένη γλώσσα που χρησιμοποιείτε για να χειριστεί και να ανακτήσει δεδομένα από τις σχεσιακές βάσεις δεδομένων. Σκοπός της είναι να παρέχει μια διεπαφή σε μια σχεσιακή βάση δεδομένων όπως η MySQL και όλες οι SQL δηλώσεις είναι οδηγίες προς τη βάση δεδομένων, κάτι που τη διαφοροποιεί από τις γενικής χρήσεως γλώσσες προγραμματισμού, όπως η C και η Basic. Μεταξύ των χαρακτηριστικών γνωρισμάτων της SQL είναι και τα εξής :

- Επεξεργάζεται τα σύνολα των δεδομένων ως ομάδες παρά ως μεμονωμένες μονάδες.
- Παρέχει αυτόματη περιήγηση στα δεδομένα.
- Χρησιμοποιεί δηλώσεις που είναι πολύπλοκες και ισχυρές χωριστά και για αυτό και στέκονται από μόνα τους.

Ουσιαστικά η SQL, σας επιτρέπει να εργαστείτε με τα δεδομένα στο λογικό επίπεδο. Χρειάζεται να ενδιαφερθείτε με τις λεπτομέρειες της εκτέλεσης μόνο όταν χρειάζεται να χειριστείτε τα δεδομένα. Για παράδειγμα, για να ανακτήσετε ένα σύνολο γραμμών από ένα πίνακα, απλά ορίζεται έναν κανόνα που χρησιμοποιείτε για να φιλτράρει το σύνολο των γραμμών του πίνακα. Όλες οι γραμμές που ικανοποιούν τον κανόνα ανακτώνται σε ένα και μόνο βήμα και μπορούν να περάσουν ως μονάδα στο χρήστη, σε μια άλλη SQL δήλωση ή σε μια εφαρμογή. Δεν χρειάζεται να εξετάσετε τις γραμμές μια προς μια, ούτε να ανησυχήσετε για το πώς αποθηκεύονται φυσικά ή ανακτώνται. Η SQL παρέχει δηλώσεις για μια ποικιλία από εργασίες, όπως :

- Διατύπωση ερωτήσεων δεδομένων.
- Εισαγωγή, ανανέωση και διαγραφή γραμμών σε έναν πίνακα.
- Δημιουργία, αντικατάσταση, αλλαγή και διαγραφή αντικειμένων.
- Να προκαλέσει ενέργειες βασισμένες σε αλλαγές στη βάση.

Επίσης επιτρέπει σε έναν προγραμματιστή ή σε ένα διαχειριστή της βάσεως να κάνει τα παρακάτω :

- Να τροποποιήσει τη δομή της βάσης δεδομένων.

- Να αλλάξει τις ρυθμίσεις ασφάλειας του συστήματος.
- Να προσθέσει δικαιώματα χρηστών στις βάσεις ή στους πίνακες.
- Να διατυπώσει ερωτήματα για να ανάκτηση πληροφοριών.
- Να ανανεώσει τα περιεχόμενα της βάσης.[13]

## 5.7 MySQL

Η αποθήκευση και ανάκτηση δεδομένων είναι ένα βασικό στοιχείο των περισσότερων εφαρμογών σήμερα. Στις παλιότερες ημέρες ανάπτυξης λογισμικού, οι προγραμματιστές έγραφαν το δικό τους κώδικα χαμηλού επιπέδου για να το καταφέρουν. Εντούτοις, γρήγορα αντιλήφθηκαν ότι σε κάθε εφαρμογή εφεύρισκαν πάλι ουσιαστικά, την ρόδα. Μέσω του συνηθισμένου κύκλου της δοκιμής, του λάθους και του επόμενου καθορισμού, μια λύση αναπτύχθηκε : η μηχανή αποθήκευσης και ανάκτησης των δεδομένων ενσωματώθηκε σε έναν αυτόνομο server βάσεως δεδομένων με τους πελάτες να συνδέονται σε αυτό και στέλνοντας αιτήματα σε μια γλώσσα την SQL. Σήμερα οι προγραμματιστές μπορούν να επιλέξουν μέσα από πολλά προϊόντα αποθήκευσης και ανάκτησης δεδομένων που χρησιμοποιούν τη γλώσσα SQL. Αυτά τα προϊόντα αναφέρονται συνήθως ως server βάσεων δεδομένων SQL ή μερικές φορές ως σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων. Μια σχεσιακή βάση δεδομένων αποθηκεύει τα δεδομένα σε ξεχωριστούς πίνακες αντί να τα βάζει σε μια μεγάλη αποθήκη.

Αυτό προσθέτει ταχύτητα. Οι πίνακες συνδέονται μεταξύ τους κάνοντας εφικτό το συνδυασμό δεδομένων από πολλού πίνακες κατόπιν αιτήσεως. Αντιθέτως με τους περισσότερους server βάσεων δεδομένων, το MySQL είναι ένα προϊόν ανοικτού κώδικα : ο πηγαίος κώδικας του είναι ελεύθερα διαθέσιμος για κατέβασμα στο καθένα. Οι προγραμματιστές μπορούν να αλλάξουν τον πηγαίο κώδικα έτσι ώστε να προσαρμόσουν το MySQL στις ανάγκες τους. Ένα από τα καλά των προϊόντων ανοικτού κώδικα είναι ότι ένα ευρύ φάσμα προγραμματιστών και χρηστών συμβάλλουν την εμπειρία τους στο λογισμικό, κάνοντας το καλύτερο. Το MySQL έχει μια μεγάλη κοινότητα πιστών υποστηρικτών.

Η πιο κοινή ανησυχία για τα προϊόντα ανοιχτού κώδικα είναι ότι δεν έχουν μια εμπορική οντότητα από πίσω τους που θα έχει την ευθύνη για το λογισμικό. Από

[13] (why-mysql)

αυτή την άποψη, το MySQL είναι ένα από τα λίγα προϊόντα ανοιχτού λογισμικού. Η MySQL AB είναι μια ολοκληρωμένη επιχείρηση, που απασχολεί 50 άτομα σε όλο τον κόσμο τα οποία είναι υπεύθυνα για την ανάπτυξη, την υποστήριξη, τις πωλήσεις, την κατάρτιση, την τεκμηρίωση και τις επιχειρησιακές λειτουργίες. Το σύστημα βάσεων δεδομένων του MySQL χρησιμοποιεί την αρχιτεκτονική πελάτη-εξυπηρετητή που συγκεντρώνεται γύρο από τον server, *mysqld*, υποστηρίζει πολλά διαφορετικά προγράμματα και βιβλιοθήκες χρηστών, εργαλεία διαχείρισης και πολλές προγραμματιστικές διεπαφές. Ο server είναι στη πραγματικότητα το πρόγραμμα που διαχειρίζεται τις βάσεις δεδομένων. Τα προγράμματα των πελατών δεν το κάνουν αυτό άμεσα, μάλλον, διαβιβάζουν την πρόθεση τους στο server με την βοήθεια ερωτήσεων γραμμένων σε SQL. Το πρόγραμμα ή τα προγράμματα των χρηστών μπορούν να αποθηκευτούν τοπικά στη μηχανή, από την οποία θα έχουν πρόσβαση στο MySQL. Αντιθέτως server μπορεί να εγκατασταθεί οπουδήποτε αρκεί οι χρήστες να μπορούν να συνδεθούν σε αυτό. Το MySQL είναι ένα εγγενώς δικτυωμένο σύστημα βάσεων δεδομένων, έτσι οι χρήστες μπορούν να επικοινωνήσουν με τον server είτε αυτός είναι εγκατεστημένος τοπικά στο μηχάνημα τους είτε σε κάποιο άλλο μηχάνημα, ακόμη και στην άλλη μεριά του πλανήτη. Όταν χρησιμοποιείτε αμφίδρομα, το MySQL εμφανίζεται για ένα ερώτημα, το αποστέλλει στο MySQL server για εκτέλεση και έπειτα εμφανίζει τα αποτελέσματα. Το MySQL μπορεί να χρησιμοποιηθεί και μη-αμφίδρομα παραδείγματος χάριν, να διαβάσει τις ερωτήσεις από ένα αρχείο ή από άλλα προγράμματα. [14]

## 5.8 Κύρια χαρακτηριστικά του MySQL

Η παρακάτω λίστα περιγράφει μερικά από τα πιο σημαντικά χαρακτηριστικά του MySQL :

- Φορητότητα.
- Καμία διαρροή μνήμης.
- Δουλεύει σε πολλές διαφορετικές πλατφόρμες.
- API για C, C++, Java, Perl, PHP, Python και TCL.
- Μπορεί εύκολα να χρησιμοποιήσει πολλαπλές CPU αν είναι διαθέσιμες.
- Πολύ γρήγορα B-tree πίνακες με συμπίεση δεικτών.
- Πολύ γρήγορο σύστημα κατανομής μνήμης.

[14] (1keydata.com)

- Οι SQL λειτουργίες εφαρμόζονται μέσω μιας ιδιαίτερα βελτιστοποιημένης βιβλιοθήκης και πρέπει να είναι όσο το δυνατό γρήγορη.

Τύποι στηλών :

1. Πολλοί τύποι στηλών : υπογεγραμμένοι / ανυπόγραφοι ακέραιοι αριθμοί FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET και ENUM τύποι.
2. Καθορισμένου και μεταβλητού μήκους εγγραφές.
3. Όλες οι στήλες έχουν προκαθορισμένες τιμές. Μπορείτε να χρησιμοποιήσετε το INSERT για να εισάγετε ένα υποσύνολο στηλών ενός πίνακα, αυτές οι στήλες που δεν δίνονται ρητά τιμές, τίθενται στις προκαθορισμένες τιμές.

Εντολές και λειτουργίες :

1. Πλήρης υποστήριξη στα τμήματα SELECT και WHERE των ερωτήσεων.
2. Πλήρης υποστήριξη στις SQL προτάσεις GROUP BY και ORDER BY.

Υποστήριξη λειτουργιών ομαδοποίησης COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX() και MIN().

3. Υποστήριξη για LEFT OUTER JOIN και RIGHT OUTER JOIN με ANSI SQL και ODBC σύνταξη.
4. Τα ψευδώνυμα στους πίνακες και στις στήλες επιτρέπονται στο πρότυπο SQL92.
5. Τα DELETE, INSERT, REPLACE και UPDATE επιστρέφουν τον αριθμό των γραμμών που άλλαξαν.
6. Η εντολή SHOW μπορεί να χρησιμοποιηθεί για να ανακτηθούν πληροφορίες σχετικές με τη βάση δεδομένων, τους πίνακες και τους δείκτες.
7. Η εντολή EXPLAIN μπορεί αν χρησιμοποιηθεί για να καθορίσει πως επιλύεται μια ερώτηση.
8. Τα ονόματα των λειτουργιών δεν έρχονται σε σύγκρουση με τα ονόματα των πινάκων ή των στηλών. Για παράδειγμα το ABS είναι ένα έγκυρο όνομα στήλης. Ο μόνος περιορισμός είναι το ότι σε μια κλήση λειτουργίας δεν πρέπει να υπάρχει κενό μεταξύ του ονόματος της λειτουργίας και του ΄(΄ που ακολουθεί.
9. Μπορείτε να αναμίξετε πίνακες από διαφορετικές βάσεις δεδομένων στο ίδιο ερώτημα.

- Εξελιξιμότητα και όρια :

1. Διαχειρίζεται μεγάλες βάσεις δεδομένων. Το MySQL χρησιμοποιείτε σε βάσεις που έχουν 50.000.000 εγγραφές και υπάρχουν χρήστες με βάσεις που έχουν 60.000 πίνακες και 5.000.000.000 γραμμές.
2. Μέχρι 32 δείκτες ανά πίνακα επιτρέπονται. Κάθε δείκτες μπορεί να αποτελείτε από 1 ως 6 στήλες ή μέρος των στηλών. Το μέγιστο μέγεθος των δεικτών είναι 500 bytes.

- Συνδετικότητα :

1. Οι πελάτες μπορούν συνδεθούν στο MySQL server χρησιμοποιώντας το TCP/IP.
2. Υποστήριξη ODBC για Win32.

- Εντοπισμός :

1. Ο server μπορεί να παρέχει μηνύματα λάθους στους πελάτες σε πολλές γλώσσες.
2. Πλήρης υποστήριξη για πολλά διαφορετικά σύνολα χαρακτήρων, συμπεριλαμβανομένου το ISO-8859-1.
3. Όλα τα δεδομένα σώζονται στο επιλεγμένο σύνολο χαρακτήρων.
4. Η ταξινόμηση γίνεται σύμφωνα με το επιλεγμένο σύνολο χαρακτήρων ( εξ ορισμού είναι ο Σουηδικός ).

Είναι δυνατό να το αλλάξετε αυτό όταν τίθεται σε λειτουργία ο MySQL server.

- Πελάτες και εργαλεία :

1. Περιλαμβάνει την myisamchk, μια πολύ γρήγορη λειτουργία για τον έλεγχο, βελτιστοποίηση και επισκευή πινάκων. [15]

## 5.9 Πόσο σταθερό είναι το MySQL;

Μέχρι τα μέσα του 1996 το MySQL είχε λειτουργήσει σε προγράμματα χωρίς κάποιο πρόβλημα. Όταν το MySQL απελευθερώθηκε σε μεγαλύτερο κοινό, έγινε αντιληπτό ότι υπήρχαν κομμάτια κώδικα που δεν είχαν περάσει κάποια δοκιμασία και τα οποία βρέθηκαν πολύ γρήγορα από τους διάφορους χρήστες. Κάθε νέα έκδοση είχε λιγότερα προβλήματα φορητότητας.

Η σχεδίαση του MySQL είναι πολυστρωματική με ανεξάρτητες ενότητες. Κάποιες από τις ενότητες παρατίθενται παρακάτω με μια ένδειξη του πόσο καλά δοκιμασμένες είναι η κάθε μια :

- Αντένσταση : οι μεγάλες συστάδες server που χρησιμοποιούν την αντένσταση

[15] (why-mysql)

είναι σε χρήση παραγωγής, με καλά αποτελέσματα.

- InnoDB πίνακες : ενώ ο InnoDB διαχειριστής πινάκων είναι μια αρκετά πρόσφατη

προσθήκη στο MySQL, φαίνεται να δουλεύει καλά και ήδη έχει χρησιμοποιηθεί σε μεγάλα συστήματα παραγωγής με βαρύ φορτίο.

- DBD πίνακες : ο κώδικας DB του Berkeley είναι πολύ σταθερός, αλλά ακόμη βελτιώνουμε τη διεπαφή στο MySQL του DBD διαχειριστή πινάκων.
- FULLTEXT : η αναζήτηση πλήρους κειμένου λειτουργεί, αλλά δεν χρησιμοποιείτε ευρέως.
- MyODBC 2.50 ( χρησιμοποιεί ODBC SDK 2.5 ) : όλο και περισσότερο χρησιμοποιείτε. Μερικά ζητήματα που τίθενται φαίνεται να είναι εφαρμογές σχετιζόμενες και ανεξάρτητες από τον οδηγό ODBC ή τον ελλοχεύοντα server βάσεων δεδομένων.
- Αυτόματη αποκατάσταση των MyISAM πινάκων : αυτή η θέση αφορά μόνο τον νέο κώδικα στο διαχειριστή πινάκων MyISAM που ελέγχει μόνο αν ο πίνακας έχει κλείσει κατάλληλα σε ένα άνοιγμα και εκτελεί ένα αυτόματο έλεγχο / επισκευή του πίνακα αν δεν έχει κλείσει κατάλληλα.
- Bulk-insert : νέο χαρακτηριστικό στους MyISAM πίνακες για γρηγορότερη εισαγωγή πολλών γραμμών.
- Locking : σε μερικά συστήματα υπάρχουν πολλά προβλήματα στη χρήση του OS locking ( fcntl() ). Σε αυτές τις περιπτώσεις, πρέπει να τρέχετε το mysqld παραλείποντας το κλείδωμα.

Η MySQL AB παρέχει υψηλής ποιότητας υποστήριξη για πελάτες που πληρώνουν, αλλά ο κατάλογος διευθύνσεων του MySQL, δίνει συνήθως απαντήσεις στις συνηθισμένες ερωτήσεις-προβλήματα. Τα σφάλματα διορθώνονται συνήθως αμέσως και για κάθε σοβαρό σφάλμα υπάρχει σχεδόν πάντα μια νέα έκδοση. [16] [17]

## 5.10 Πόσο μεγάλοι μπορεί να είναι οι πίνακες στο MySQL;

Στην έκδοση 3.22 του MySQL οι πίνακες είχαν ένα όριο των 4G. Με τον νέο τύπο πινάκων MyISAM στην 3.23 έκδοση του MySQL, το μέγιστο μέγεθος των πινάκων ωθήθηκε μέχρι τα 8 εκατομμύρια terabytes ( 2^63 ).

[16] (Sql)
[17] (why-mysql)

## 5.11 Πλεονεκτήματα και μειονεκτήματα του MySQL

### 5.11.1 Πλεονεκτήματα :

- Ταχύτητα : Ο κώδικας του πυρήνα της MySQL γράφτηκε από την αρχή μέχρι το

τέλος με άριστη απόδοση ως αρχικός στόχος.

- Αξιοπιστία : Το MySQL έχει κερδίσει τη φήμη του ότι είναι σε θέση να τρέχει αφύλακτο για μέρες ακόμη και μήνες μετά από την αρχική οργάνωση.
- Χαμηλές απαιτήσεις σε πόρους συστήματος : Το MySQL είναι ικανό να κάνει το καλύτερο με τους πόρους που του δίνεται. Φυσικά, όσο περισσότεροι πόροι, τόσο καλύτερη απόδοση να περιμένετε, αλλά ελάχιστοι πόροι δεν θα βγάλουν το MySQL εκτός λειτουργίας.
- Εξελιξιμότητα : Η εμπειρία έδειξε ότι το MySQL αποδίδει καλά σε συστήματα με μέχρι 4 επεξεργαστές και μνήμη μέχρι 4GB εκμεταλλευόμενο πλήρως τους πόρους του συστήματος.
- Ποικιλομορφία πλατφόρμων : Τρέχει σε μια ευρεία ποικιλία υπολογιστικών συστημάτων. Ανάμεσα στα οποία τα πιο δημοφιλή είναι τα Linux, Windows, Solaris και FreeBSD.
- Υποστήριξη για έναν μεγάλο αριθμό γλωσσών : Όταν αναπτύσσετε μια εφαρμογή για τη βάση δεδομένων, μια από τις αρχικές ανησυχίες είναι η διασύνδεση με τον server βάσεων δεδομένων χρησιμοποιώντας κάποια γλώσσα προγραμματισμού.

Αυτός είναι ακόμη ένας τομέας της δύναμης του MySQL. Οι προγραμματιστές μπορούν να επικοινωνήσουν με το MySQL χρησιμοποιώντας C/C++, PHP, Perl, Java, Python, TCL, Ruby και Eiffel.

- Υποστήριξη ODBC : Περιλαμβάνει οδηγό ODBC. Αυτό δίνει τη δυνατότητα στους προγραμματιστές να γράψουν εφαρμογές χρησιμοποιώντας Open Database Connectivity. Το ODBC επιτρέπει στο MySQL να χρησιμοποιηθεί στο Microsoft Access, Excel και άλλα. Επίσης επιτρέπει στο MySQL να χρησιμοποιηθεί σε εφαρμογές σε Visual Basic και Delphi, στην ASP και σε άλλα εργαλεία και περιβάλλοντα ανάπτυξης εφαρμογών.
- Δωρεάν ή χαμηλού κόστους χορήγηση αδειών : Το MySQL διανέμεται κάτω από την άδεια GPL ( General Public License ). Αυτή η άδεια σας επιτρέπει να

το χρησιμοποιείτε και για εμπορικό και για μη εμπορικό σκοπό υπό την προϋπόθεση ότι οποιοδήποτε παραγόμενο προϊόν πρέπει να διανεμηθεί με ολόκληρο τον πηγαίο κώδικα κάτω από τους όρους της ίδιας άδειας.

- Ανέξοδη εμπορική υποστήριξη : Για όσους σκέφτονται να τρέξουν το MySQL σε ένα περιβάλλον, το ζήτημα υψηλής ποιότητας εμπορικής υποστήριξης είναι πολύ σημαντικό. Το MySQL AB παρέχει ένα ευρύ φάσμα εμπορικής υποστήριξης σε λογική τιμή, συμπεριλαμβανομένου 24x7 τηλεφωνική υποστήριξη.
- Ισχυρή υποστήριξη κοινοτήτων χρηστών : Όπως αναφέρθηκε πιο πάνω το MySQL δεν είναι απλά μια βάση δεδομένων. Οι ιδρυτές του MySQL έχουν εστιάσει τη προσοχή τους στο να δίνουν στη κοινότητα περισσότερα από αυτά που παίρνουν από αυτήν. Η κοινότητα ανταποκρίθηκε με αφοσίωση, σκληρή δουλειά και συναδελφοσύνη. Έτσι μπορεί κάποιος χρήστης να βρει τις απαντήσεις στα προβλήματα που δεν μπορεί να επιλύσει λαμβάνοντας βοήθεια από τη κοινότητα, πέρα από την βοήθεια που υποστηρίζει το MySQL. Μια άλλη πτυχή της ισχυρής κοινότητας είναι το ότι μπορεί να βρει κανείς ένα ειδικό στο MySQL για να εργαστεί για εσάς.
- Διαθεσιμότητα του πηγαίου κώδικα : Η πρόσβαση στον πηγαίο κώδικα είναι κάτι πολύ σημαντικό για τις επιχειρήσεις που απασχολούν πεπειραμένους C/C++ προγραμματιστές. Παρέχει την ευκαιρία να γίνουν διάφορες προσαρμογές, βελτιώσεις, επεκτάσεις και διόρθωση λαθών, χωρίς να περιμένεις των προμηθευτή να το κάνει.

### 5.11.2 Αδυναμίες :

- Έλλειψη ορισμένων χαρακτηριστικών γνωρισμάτων της SQL : Η πιο σοβαρή αδυναμία του MySQL είναι ότι αυτή τη περίοδο δεν υποστηρίζει εμφωλευμένα ερωτήματα, όψεις, αποθηκευμένες διαδικασίες, εναύσματα και επιβολή ξένου κλειδιού.
- Έλλειψη λεπτομερούς δοκιμής σε συγκεκριμένες πλατφόρμες : Το MySQL AB μπορεί να εξασφαλίσει ως ένα επίπεδο την ποιότητα του MySQL σε μια πλατφόρμα από εκεί και μετά, η ποιότητα αυτή θα αυξάνεται όσοι περισσότεροι χρησιμοποιούν το MySQL στη πλατφόρμα αυτή και αποστέλλουν τα σφάλματα που βρίσκουν. Επομένως η εξασφάλιση της ποιότητας εξαρτάτε κατά ένα μεγάλο μέρος από τις δοκιμές που κάνουν οι

χρήστες, οπότε όσοι περισσότεροι χρήστες τόσο λιγότερα σφάλματα θα περάσουν απαρατήρητα, για μεγάλο χρονικό διάστημα. Βέβαια αυτό δεν σημαίνει ότι σε πλατφόρμες που δεν χρησιμοποιούν πολλοί χρήστες το MySQL δεν θα τρέχει και μάλιστα καλά, αλλά ίσως να υπάρξουν κάποια προβλήματα.

- Δυσκολία εργασίας με τον πηγαίο κώδικα του server : Το δύσκολο του να δουλεύει κανείς με τον πηγαίο κώδικα που έγραψε κάποιος άλλος είναι ότι πρέπει να τον καταλάβει για να υπάρχει κάποια πρακτική αξία. Ο κώδικας του MySQL server είναι πολύ δύσκολο να το καταλάβει ακόμη και ένας ειδικευμένος πεπειραμένος προγραμματιστής της C/C++. Υπάρχουν δυο κύριοι λόγοι της δυσκολίας αυτής. Πρώτα, είναι ένας server βάσεων δεδομένων, το οποίο σημαίνει ότι πρέπει αν έχει το κώδικα για να οργανώσει τα δεδομένα στο σκληρό δίσκο και να επιλέξει μια στρατηγική για την επίλυση ενός συγκεκριμένου ερωτήματος.

# 6 Η αναγκαιότητα αυτοματοποίησης των βιβλιοπωλείων

Η επίδραση των τεχνολογιών Πληροφορίας και Επικοινωνίας καλύπτει σταδιακά ολόκληρο το φάσμα των επιχειρηματικών ή διοικητικών συναλλαγών που λαμβάνουν χώρα μέσα στο αστικό σώμα και προσφέρει νέες καινοτομίες και επιχειρησιακές δραστηριότητες

Με την ανάπτυξη του διαδικτύου, της ψηφιακής τεχνολογίας και των Τηλεπικοινωνιών:

- Οι περισσότερες επιχειρηματικές λειτουργίες σήμερα μετατρέπονται σε ηλεκτρονικές.
- τα γεωγραφικά και τα χρονικά όρια λειτουργίας των επιχειρήσεων εκμηδενίζονται,
- καινούργιες μορφές αγορών δημιουργούνται,
- νέα υβριδικά προϊόντα δημιουργούνται,

ενώ ταυτόχρονα:

- γίνεται ολοένα και πιο δυσδιάκριτη η φύση των αγαθών από τις υπηρεσίες.
- Οι ιεραρχίες της χρηστικής σημασίας των προϊόντων και υπηρεσιών αλλάζουν.
- νέες δυναμικές σχέσεις αναπτύσσονται μεταξύ πελατών και εταιρειών.

Η βάση της ηλεκτρονικοποίησης των δραστηριοτήτων στηρίζεται στη δυνατότητα

- Εξ αποστάσεως αναγνώρισης και παραγγελίας των προϊόντων,
- Οργανωμένης μαζικής κεντρικής αποστολής τους στους καταναλωτές

υπηρεσιών. Τούτο έχει άμεση επίπτωση στη μείωση των κυκλοφοριακών φόρτων και των χώρων στάθμευσης.

- διαφοροποίησης της φύσης των προϊόντων σε υλικά και άυλα
- Επέκτασης των δυνατοτήτων εξαϋλωμένων διαχειριστικών λύσεων των υλικών προϊόντων.

Διαπιστώνουμε ήδη ευρείας κλίμακας ηλεκτρονικοποίηση, και χρήση του διαδικτύου στις παρακάτω δραστηριότητες.

**A) e-business**

Ένας ριζικός μετασχηματισμός πραγματοποιείται στον επιχειρηματικό κόσμο.

- οι ηλεκτρονικές συναλλαγές,
- η δημιουργία των ηλεκτρονικών αγορών (e-markets), και
- οι ηλεκτρονικές επιχειρηματικές κοινότητες (networked business communities),

καταρρίπτουν τα παραδοσιακά επιχειρηματικά μοντέλα.

**B) e-commerce**

Πρέπει να επισημανθεί, ότι το πραγματικό ηλεκτρονικό εμπόριο είναι κάτι περισσότερο από απλή πώληση προϊόντων στο web. Μπορεί να διακρίνει κανείς δύο βασικές κατηγορίες δραστηριοποίησης, οι οποίες δημιουργούν και διαφορετικές επιχειρηματικές διαδικασίες.

**α) Το ηλεκτρονικό εμπόριο που απευθύνεται στον καταναλωτή**

**β) Το ηλεκτρονικό εμπόριο που επεκτείνεται σε επιχειρηματικούς πόρους και διαδικασίες.**

**Γ ) e-Procurement**

Η διαχείριση προμηθειών μέσω Internet, είναι ένα από τα βασικά συστατικά του ηλεκτρονικού «επιχειρείν».

Η ηλεκτρονική διαχείριση προμηθειών παρέχει σημαντικές δυνατότητες μείωσης των δαπανών, και προσφέρει νέες ευκαιρίες κερδοφορίας, στοιχεία που την καθιστούν βασικό κομμάτι του στρατηγικού σχεδιασμού των επιχειρήσεων στη **Νέα Οικονομία.**

**Δ ) e-Government**

Αλλάζει τον τρόπο που οι Κυβερνήσεις και οι Οργανισμοί Τοπικής Αυτοδιοίκησης εξυπηρετούν τους πολίτες. Η εποχή της ηλεκτρονικής διακυβέρνησης έχει ήδη ξεκινήσει. Στην απλή της έκφραση αφορά στη χρήση τεχνολογίας ούτως ώστε να πλουτιστούν τα μέσα πρόσβασης και παροχής υπηρεσιών στους πολίτες, και στις επιχειρήσεις με στόχο μία «**πελατο-κεντρική**» προσέγγιση.

**E) e-Marketplaces.**

Με τον όρο e-Marketplaces εννοούμε ηλεκτρονικές αγορές δομημένες για να εξυπηρετήσουν διεπιχειρησιακές συναλλαγές (B2B e-Marketplaces).

**Το ηλεκτρονικό βιβλιοπωλείο**

Αποτελεί μια web εφαρμογή η οποία μπορεί να λειτουργήσει αυτόνομα ή ως συνάρτημα ενός ιστοχώρου που σχετίζεται εν γένει με ενός βιβλιοπωλείου τις δραστηριότητες. Η εφαρμογή υποστηρίζει δύο επίπεδα λειτουργιών, του χρήστη και του διαχειριστή του συστήματος, παρέχοντας τη δυνατότητα κατά επίπεδο:

### 6.1.1 Επίπεδο Διαχειριστή

- Διαχείριση κωδικών
- Τροποποίηση (διαγραφή, ενημέρωση, προσθήκη) στοιχείων βιβλίων
- Τροποποίηση (διαγραφή, ενημέρωση, προσθήκη) στοιχείων κατηγοριών π.χ Ιστορικά, πολιτικά κλπ.
- Δυνατότητα ακύρωσης ή ενημέρωσης αγοράς κατ΄ επιθυμία του πελάτη
- Δυνατότητα έκδοσης συγκεντρωτικών ή μη αναφορών των πωλήσεων.

### 6.1.2 Επίπεδο Χρήστη

- Δυνατότητα επιλογής βιβλίων με παράλληλη ενημέρωση για τη διαθεσιμότητα αυτών.
- Εισαγωγή στοιχείων του χρήστη προκειμένου να τεκμηριωθεί η συναλλαγή
- Έκδοση απόδειξης

Όλες αυτές οι υπηρεσίες εξυπηρετούν στην εύκολη και ομαλή λειτουργία της ηλεκτρονικής αγοράς βιβλίων.

### 6.1.3 Σε επίπεδο Διαχειριστή

Διαχείριση κωδικών: Η συγκεκριμένη υπηρεσία παρέχεται στον διαχειριστή για την ασφαλή εισαγωγή του στο σύστημα και στην αποφυγή κακόβουλων επιθέσεων.

Τροποποίηση (διαγραφή, ενημέρωση, προσθήκη) στοιχείων βιβλίων: Ο διαχειριστής μπορεί να τροποποιήσει τα στοιχεία των βιβλίων κατά βούληση και να ενημερώσει το σύστημα με τα χαρακτηριστικά αυτών.

Τροποποίηση (διαγραφή, ενημέρωση, προσθήκη) στοιχείων κατηγοριών : Καλείται ο διαχειριστής να τροποποιήσει τις κατηγορίες των βιβλίων (ιστορικά, πολιτικά κλπ) και να ενημερώσει σχετικώς το σύστημα.

Δυνατότητα ακύρωσης ή ενημέρωσης πωλήσεων κατ΄ επιθυμία του πελάτη: Ο διαχειριστής έπειτα από απαίτηση του πελάτη έχει την δυνατότητα να ακυρώσει ή να διαφοροποιήσει μία πώληση.

Δυνατότητα έκδοσης συγκεντρωτικών ή μη αναφορών των πωλήσεων: Ο διαχειριστής μπορεί να εκδώσει συγκεντρωτικές ή μη αναφορές των πωλήσεων της εφαρμογής.

### 6.1.4 Σε επίπεδο Χρήστη

Δυνατότητα επιλογής βιβλίων και χρονικής διάρκειας με παράλληλη ενημέρωση για τη διαθεσιμότητα αυτών: Εδώ υπάρχει μία φόρμα σύμφωνα με την οποία ο χρήστης θα επιλέγει τι ψάχνει και από πού. Δηλαδή να διαλέγει το βιβλίο βάση του συγγραφέα, της κατηγορίας, της τιμής κλπ.

Εισαγωγή στοιχείων του χρήστη προκειμένου να τεκμηριωθεί η συναλλαγή: Εδώ θα υπάρχει μία φόρμα σύμφωνα με την οποία ο χρήστης θα εισάγει τα στοιχεία του όπως ονοματεπώνυμο, διεύθυνση, email κλπ.

Έκδοση απόδειξης: Εδώ ο χρήστης μπορεί να πάρει το παραστατικό της κράτησης που έκανε, με τα στοιχεία της κράτησης.

### 6.1.5 Μέθοδος έρευνας:

Βιβλιογραφική έρευνα σε διεθνείς και ελληνικές πηγές (μελέτες, έρευνες, εξειδικευμένα δημοσιεύματα), καταγραφή απαιτήσεων, ανάλυση απαιτήσεων, σχεδιασμός, υλοποίηση, έλεγχος σχεδίαση και κατασκευή εφαρμογής,

### 6.1.6 Σκοπός της έρευνας:

Είναι γνωστό ότι σήμερα τα διάφορα βιβλιοπωλεία δουλεύουν σε μικρή κλίμακα, τοπικού χαρακτήρα και στηρίζονται στις πελατειακές σχέσεις που δημιουργούνται τόσο σε τοπικό γεωγραφικό επίπεδο όσο και στην προσωπική σχέση.

Λαμβάνοντας υπ' όψιν τα παραπάνω αντιλαμβανόμαστε ότι η κατασκευή μιας διαδικτυακής εφαρμογής λογικά να βοηθούσε στην κάλυψη των παραπάνω αναγκών και με στόχο να βελτιωθούν τα ήδη εφαρμοζόμενα και να σχεδιασθούν εκ νέου κοινά αποδεκτά. Για αυτό η παρούσα εργασία προσπαθεί:

- να καταγράψει και ν' αξιολογήσει συστηματικά τη υπάρχουσα εμπειρία στον τομέα αυτό
- να εκτιμήσει την έκταση, το είδος των ήδη υπαρχουσών ηλεκτρονικών βιβλιοπωλείων
- να μελετήσει, αναλύσει και σχεδιάσει μία εφαρμογή που θα μπορούσε να καλύψει ως ένα σημείο την κάλυψη της ανάγκης.
- Να κατασκευάσει την εφαρμογή
- Τέλος να την αξιολογήσει και να αναφέρει κέρδη και μελλοντικές επεκτάσεις

.

### 6.1.7 Εισαγωγή / Σύνοψη υπάρχουσας κατάστασης

Ο τομέας των βιβλιοπωλείων μπορεί να αποκομίσει πολύπλευρα οφέλη από την ενσωμάτωση των Τεχνολογιών Πληροφορικής στη λειτουργία του. Τα οφέλη αυτά αντικατοπτρίζονται στους ειδικούς στόχους που περιλαμβάνουν τη μείωση του χρόνου πραγματοποίησης μιας κράτησης, τον περιορισμό της απώλειας παραγωγικού χρόνου, στην προβολή της επιχείρησης, καθώς και στη βελτίωση της επιχειρηματικότητα και παραγωγικότητα**.**

Η συλλογή, συστηματοποίηση και εκμετάλλευση πληροφορίας και η χρήση τηλεπικοινωνιακών δικτύων μεταφοράς, αυξάνουν σημαντικά την αποδοτικότητα και τη λειτουργικότητα των υποδομών των βιβλιοπωλείων. Χωρίς να υποκαθιστά τη δημιουργία νέων υποδομών, η προσέγγιση αυτή εστιάζεται στη βελτιστοποίηση της εκμετάλλευσης των υφισταμένων. Η εκμετάλλευση των σύγχρονων τεχνολογιών στον

τομέα των βιβλιοπωλείων προϋποθέτει υποδομή και τεχνογνωσία τόσο σε επίπεδο συστήματος όσο και σε επίπεδο χρήστη.

### 6.1.8 Ειδικοί στόχοι της εφαρμογής είναι:

- η πραγματοποίηση κρατήσεων μέσω διαδικτύου για όλο το εύρος των υπηρεσιών και προϊόντων των βιβλιοπωλείων,
- στην αναβάθμιση της ποιότητας των υπηρεσιών και της ανταγωνιστικότητας των βιβλιοπωλείων
- η ποιοτική και ολοκληρωμένη παρουσίαση των βιβλιοπωλείων καθώς και της τοποθεσίας στην οποία βρίσκονται,
- στην μείωση του κόστους
- η προσέλκυση νέων επισκεπτών και η διατήρηση πελατών μέσω της παροχής ολοκληρωμένων ηλεκτρονικών και μη υπηρεσιών από τα βιβλιοπωλείων
- την εισαγωγή και προώθηση νέων, καινοτομικών τρόπων κράτησης και πώλησης των υπηρεσιών των βιβλιοπωλείων, με γνώμονα την διευκόλυνση της πρόσβασης στο κοινό στην πληροφορία και τα εισιτήρια και την ενίσχυση των επιχειρήσεων

Έχοντας πλέον το θεωρητικό υπόβαθρό πρέπει να ορίσουμε το λογισμικό μας και τις απαιτήσεις του.

# 7 Απαιτήσεις από το Λογισμικό

## 7.1 Ορισμός:

Μια λειτουργία που θα πρέπει το λογισμικό να επιτελεί ή μια συνθήκη που θα πρέπει να ικανοποιεί όταν θα έχει ολοκληρωθεί η κατασκευή του αφορούν τη συμπεριφορά του λογισμικού προς το εξωτερικό του περιβάλλον (χρήστης, άλλες εφαρμογές, λογισμικού) και όχι εσωτερικά του στοιχεία

Λειτουργικές απαιτήσεις: περιγράφουν τις εργασίες (λειτουργίες) που θα πρέπει να εκτελεί το λογισμικό, καθορίζουν τη συμπεριφορά του συστήματος, δηλ. την απόκριση που πρέπει να εμφανίζει στο περιβάλλον του όταν ισχύουν συγκεκριμένες συνθήκες

Μη λειτουργικές απαιτήσεις: περιγράφουν χαρακτηριστικά που πρέπει να έχει το λογισμικό τα οποία δεν αφορούν την εκτέλεση κάποιας λειτουργίας από αυτό. Καθορίζουν ιδιώματα εμφάνισης (αισθητική, επικοινωνία με το χρήστη), επιδόσεων (αξιοπιστία, χρόνος εκτέλεσης, χρήση πόρων), υλοποίησης, κ.τ.λ.

## 7.2 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις αφορούν στην ανάπτυξη μίας εφαρμογής η οποία θα ικανοποιεί τις ακόλουθες απαιτήσεις :

Τα προγράμματα καταχώρησης που θα αφορούν στα ακόλουθα αντικείμενα :

### 7.2.1 Τι περιλαμβάνει η διαδικασία κράτησης

Περιλαμβάνει τρία βήματα:

Εισαγωγή Επιθυμίας

Ο πελάτης θέτει την επιθυμία του για την κράτηση. Η επιθυμία του πελάτη αφορά αγορά ενός βιβλίου.

Πρόταση Κράτησης

Το σύστημα αποκρίνεται στην επιθυμία με επιλογή αγοράς, οπότε υπάρχουν οι εξής περιπτώσεις:

- είναι δυνατόν να ικανοποιηθεί το αίτημα του πελάτη όπως είναι, άρα υπάρχει η

επιλογή της αγοράς, η οποία στη συνέχεια μπορεί να επιλεγεί ως έχει,

- δεν ικανοποιείται το αίτημα του πελάτη (π.χ. δεν υπάρχει διαθέσιμο βιβλίο) και δεν υπάρχει δυνατότητα κράτησης.

## 7.3 Μη Λειτουργικές Απαιτήσεις

### 7.3.1 Απαιτήσεις χρήσης

Το λογισμικό θα πρέπει να περιέχει φιλικό περιβάλλον προς τον χρήστη με υποστήριξη γραφικών και συνδυασμό χρήσης ποντικιού και πληκτρολογίου. Θα υπάρχουν φόρμες καταχώρησης **στοιχείων με αυτοματοποίηση πεδίων όπου αυτό επιτρέπεται.**

### 7.3.2 Απαιτήσεις αξιοπιστίας

Η πρόσβαση από πολλούς χρήστες ταυτόχρονα προστατεύεται με δυνατότητα να χρησιμοποιούν την εφαρμογή με δικό τους κωδικό χρήστη και password. Το λογισμικό θα υποστηρίζει «κλείδωμα» των εγγραφών κάθε χρήστη ώστε να μην υπάρχουν προβλήματα λάθους κατά τη διάρκεια ενημέρωσης από άλλο χρήστη.

### 7.3.3 Απαιτήσεις επιδόσεων

Θα υπάρχει αξιοποίηση των πόρων του συστήματος, όποιων κι αν είναι αυτοί.. Επιπλέον, η απόκριση από το σύστημα βάσεων δεδομένων δεν θα ξεπερνά τα 2 δευτερόλεπτα.

### 7.3.4 Απαιτήσεις υποστήριξης

Το λογισμικό θα διατίθεται σε περιβάλλον Windows και θα εγκαθίσταται ιδιαίτερα εύκολα σε οποιοδήποτε μηχάνημα με υποστήριξη και αρχείου README το οποίο θα επεξηγεί και θα διευκρινίζει τα θέματα εγκατάστασης και λειτουργίας της εφαρμογής.

### 7.3.5 Απαιτήσεις υλοποίησης

Θα χρησιμοποιηθεί η γλώσσα PHP λόγω της δυνατότητας ανάπτυξης δια-πλατφορμικών εφαρμογών με αντικειμενοστραφή προγραμματισμό σε συνδυασμό με χρήση SQL για υποβολή ερωτημάτων προς τη Βάση Δεδομένων.

### 7.3.6 Απαιτήσεις Βάσεων Δεδομένων

Θα χρησιμοποιηθεί σύστημα Βάσεων Δεδομένων με τα ακόλουθα βασικά αρχεία/πίνακες : Αρχείο βιβλίων ( με τα στοιχεία των κάθε βιβλίων), Αρχείο κρατήσεων (με τα στοιχεία της κάθε κράτησης), Αρχείο θέσεων (με τα στοιχεία των υπαρχόντων κρατήσεων και των κατειλημμένων θέσεων)

### 7.3.7 Φυσικές απαιτήσεις

Το λογισμικό αρχικά θα εγκατασταθεί σε έναν server με ελάχιστη απαίτηση Pentium IV, 128 MB RAM και θα έχει την δυνατότητα εγκατάστασης σε πολλά τερματικά PCs (Pentium II) με λειτουργικό Windows NT στα οποία θα έχει εγκατασταθεί MySQL και Php.[18]

[18] (Ian Sommerville, 2008)

# 8 Περιγραφή της βάσης δεδομένων

Η πτυχιακή εργασία βασίζεται πάνω σε php και MySql. Περιλαμβάνει μια βάση δεδομένων με το όνομα ¨ sc_order_system¨ η οποία περιέχει πίνακες δεδομένων.

Μέσα σε κάθε πίνακα δηλώνουμε τον τύπο κάθε πεδίου (π.χ. Char, int, date) και το πρωτεύων του κλειδί (primary key), το οποίο μπορεί να αποτελείται από ένα πεδίο ή και από συνδυασμό περισσότερων πεδίων με την προϋπόθεση να είναι μοναδικό (unique). Οι πίνακες μέσα στη βάση συνδέονται μεταξύ τους με τα πρωτεύοντα κλειδιά, έτσι ώστε εάν ένας πίνακας αλλάξει δεδομένα, να ενημερώνονται και οι υπόλοιποι πίνακες που είναι συνδεδεμένοι με αυτόν. Επίσης, μέσω των ξένων κλειδιών (foreign keys), τα οποία αποτελούνται από πεδία-ορόσημα για την περάτωση του σκοπού μας, καταφέρνουμε να συνδέσουμε δυναμικά ή στατικά τους πίνακες, κυρίως οπτικά για να εμφανίζουμε δηλαδή τα πεδία ή τους πίνακες που επιθυμούμε κατά την περιήγηση μας στο site.

Παρακάτω παραθέτουμε την βάση μας:

## 8.1 **Πίνακας** accesslevel

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| ☐ | **accessLevelId** | int(11) | | | Όχι | *None* | auto_increment |
| ☐ | **description** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση του επίπεδου πρόσβασης του κάθε χρήστη με την αντίστοιχη περιγραφή.

## 8.2 accesslevel_x_applications

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| ☐ | **accessLevelId** | int(11) | | | Όχι | *None* | |
| ☐ | **applicationId** | varchar(255) | latin1_swedish_ci | | Όχι | | |

Ο πίνακας αυτός χρησιμοποιείται για την διασύνδεση του επίπεδου πρόσβασης του κάθε χρήστη με την αντίστοιχη εφαρμογή και το αντίστοιχο προνόμιο.

## 8.3 address

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | addressId | int(11) | | | Όχι | None | auto_increment |
| | customersId | int(11) | | | Όχι | 0 | |
| | description | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | address1 | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | address2 | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | city | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | state | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | country | char(2) | latin1_swedish_ci | | Ναι | NULL | |
| | postalCode | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | phone1 | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | phone2 | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των διευθύνσεων, των στοιχείων αυτών καθώς και την αντιστοίχηση με τους πελάτες.

## 8.4 applications

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | applicationId | varchar(255) | latin1_swedish_ci | | Όχι | None | |
| | description | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείται για τις εφαρμογές και την περιγραφή αυτών.

## 8.5 cardtype

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | cardTypeId | int(11) | | | Όχι | None | auto_increment |
| | description | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των τύπων καρτών.

## 8.6 cart

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | cartId | int(11) | | | Όχι | None | auto_increment |
| | session_id | varchar(32) | latin1_swedish_ci | | Ναι | NULL | |
| | productId | int(11) | | | Ναι | NULL | |
| | quantity | int(11) | | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείται για το καλάθι αγορών, την διασύνδεση των προιόντων και την ποσότητα αυτών.

## 8.7 category

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | categoryId | int(11) | | | Όχι | None | auto_increment |
| | description | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | list_order | int(11) | | | Ναι | 0 | |

Ο πίνακας αυτός χρησιμοποιείται για τις κατηγορίες, την περιγραφή αυτών καθώς και την λίστα παραγγελιών.

### 8.8 country

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | countryCode | char(2) | latin1_swedish_ci | | Όχι | *None* | |
| | name | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | tax | char(1) | latin1_swedish_ci | | Ναι | *NULL* | |
| | taxPercent | decimal(10,2) | | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείται για τα στοιχεία της χώρας, όπως το ονομά και ο φορολογικός συντελεστής.

### 8.9 creditcards

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | creditCardsId | int(11) | | | Όχι | *None* | auto_increment |
| | cardTypeId | int(11) | | | Όχι | 0 | |
| | customersId | int(11) | | | Όχι | 0 | |
| | expirationDate | date | | | Ναι | *NULL* | |
| | cardNumber | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | cardHolder | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | securityCode | varchar(10) | latin1_swedish_ci | | Ναι | *NULL* | |
| | address1 | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | address2 | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | city | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | state | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | country | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | postalCode | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την αποθήκευση των στοιχείων των πιστωτικών καρτών.

### 8.10 customers

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | customersId | int(11) | | | Όχι | *None* | auto_increment |
| | name | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | email | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | passwd | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την αποθήκευση των στοιχείων των πελατών και την διασύνδεση τους με τον πίνακα.

### 8.11 department

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | departmentId | int(11) | | | Όχι | *None* | auto_increment |
| | description | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την περιγραφή των τμημάτων της εταιρίας.

### 8.12 emailtemplates

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **emailTemplatesId** | int(11) | | | Όχι | *None* | auto_increment |
| | **status** | int(11) | | | Όχι | *None* | |
| | **body** | text | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την περιγραφή της εμφάνιση του email και της κατάστασης αυτού.

### 8.13 measure

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **measureId** | int(11) | | | Όχι | *None* | auto_increment |
| | **description** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την αποθήκευση των μετρικών.

### 8.14 orderitens

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **orderId** | int(11) | | | Όχι | *None* | auto_increment |
| | **productId** | int(11) | | | Όχι | *None* | |
| | **quantity** | int(11) | | | Ναι | *NULL* | |
| | **unitPrice** | decimal(10,2) | | | Ναι | *NULL* | |
| | **total** | decimal(10,2) | | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την διασύνδεση των παραγγελιών με τα αντικείμενα, την ποσότητα και την τιμή αυτών.

### 8.15 orders

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **orderId** | int(11) | | | Όχι | *None* | auto_increment |
| | **customerId** | int(11) | | | Όχι | 0 | |
| | **status** | int(11) | | | Όχι | *None* | |
| | **date** | datetime | | | Ναι | *NULL* | |
| | **total** | decimal(10,2) | | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την διασύνδεση των παραγγελιών με τους πελάτες, την ημερομηνία, την κατάσταση και την τιμή αυτών.

### 8.16 orderstatus

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **statusId** | int(11) | | | Όχι | *None* | auto_increment |
| | **description** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείτε για την περιγραφή της κατάσταση της παραγγελίας.

## 8.17 producthome

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | productHomeId | int(11) | | | Όχι | None | auto_increment |
| | image | blob | | BINARY | Όχι | None | |
| | list_order | int(11) | | | Ναι | NULL | |
| | description | text | latin1_swedish_ci | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείτε για την διασύνδεση των προιόντων με την φωτογραφία (αντικείμενο blob) και την περιγραφή.

## 8.18 products

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | productId | int(11) | | | Όχι | None | auto_increment |
| | shippingId | int(11) | | | Όχι | None | |
| | measure | int(11) | | | Όχι | None | |
| | categoryId | int(11) | | | Όχι | None | |
| | subcategoryId | int(11) | | | Όχι | 0 | |
| | name | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | description | text | latin1_swedish_ci | | Όχι | None | |
| | picture | blob | | BINARY | Ναι | NULL | |
| | price | decimal(10,2) | | | Ναι | NULL | |
| | taxable | char(1) | latin1_swedish_ci | | Ναι | NULL | |
| | onHandQuantity | int(11) | | | Ναι | NULL | |
| | active | char(1) | latin1_swedish_ci | | Ναι | NULL | |
| | overview | text | latin1_swedish_ci | | Ναι | NULL | |
| | specifications | text | latin1_swedish_ci | | Ναι | NULL | |
| | acessories | text | latin1_swedish_ci | | Ναι | NULL | |

Ο πίνακας χρησιμοποιείτε για τα προιόντα, περιέχει τα στοιχεία τους, την τιμή καθώς και άλλα και τα διασυνδέει με τις κατηγορίες.

## 8.19 shipping

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | shippingId | int(11) | | | Όχι | None | auto_increment |
| | description | varchar(255) | latin1_swedish_ci | | Ναι | NULL | |
| | fixedShipRate | char(1) | latin1_swedish_ci | | Ναι | NULL | |
| | fixedWeightRate | decimal(10,2) | | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείτε για τις παραγγελίες.

## 8.20 shippingrates

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | shippingRatesId | int(11) | | | Όχι | None | auto_increment |
| | shippingId | int(11) | | | Όχι | 0 | |
| | weightFrom | decimal(10,2) | | | Ναι | NULL | |
| | weightTo | decimal(10,2) | | | Ναι | NULL | |
| | price | decimal(10,2) | | | Ναι | NULL | |

Ο πίνακας αυτός χρησιμοποιείτε για τις παραγγελίες και τα χαρακτηριστικά αυτών.

### 8.21 state

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **stateId** | char(2) | latin1_swedish_ci | | Όχι | *None* | |
| | **countryCode** | char(2) | latin1_swedish_ci | | Όχι | | |
| | **name** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείται για τις περιοχές των κρατών.

### 8.22 subcategory

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **subcategoryId** | int(11) | | | Όχι | *None* | auto_increment |
| | **categoryId** | int(11) | | | Όχι | *None* | |
| | **description** | varchar(255) | latin1_swedish_ci | | Όχι | *None* | |
| | **list_order** | int(11) | | | Ναι | 0 | |

Ο πίνακας αυτός χρησιμοποιείται για τις υποκατηγορίες, την περιγραφή αυτων καθώς και την λίστα παραγγελιών.

### 8.23 systemuser

| | Πεδίο | Τύπος | Collation | Χαρακτηριστικά | Κενό | Προκαθορισμένο | Πρόσθετα |
|---|---|---|---|---|---|---|---|
| | **systemUserId** | int(11) | | | Όχι | *None* | auto_increment |
| | **accessLevelId** | int(11) | | | Όχι | 0 | |
| | **departmentId** | int(11) | | | Όχι | 0 | |
| | **login** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | **passwd** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |
| | **name** | varchar(255) | latin1_swedish_ci | | Ναι | *NULL* | |

Ο πίνακας αυτός χρησιμοποιείται για τους χρήστες, τα επίπεδα πρόσβασης και τα στοιχεία διασύνδεσης με το σύστημα.

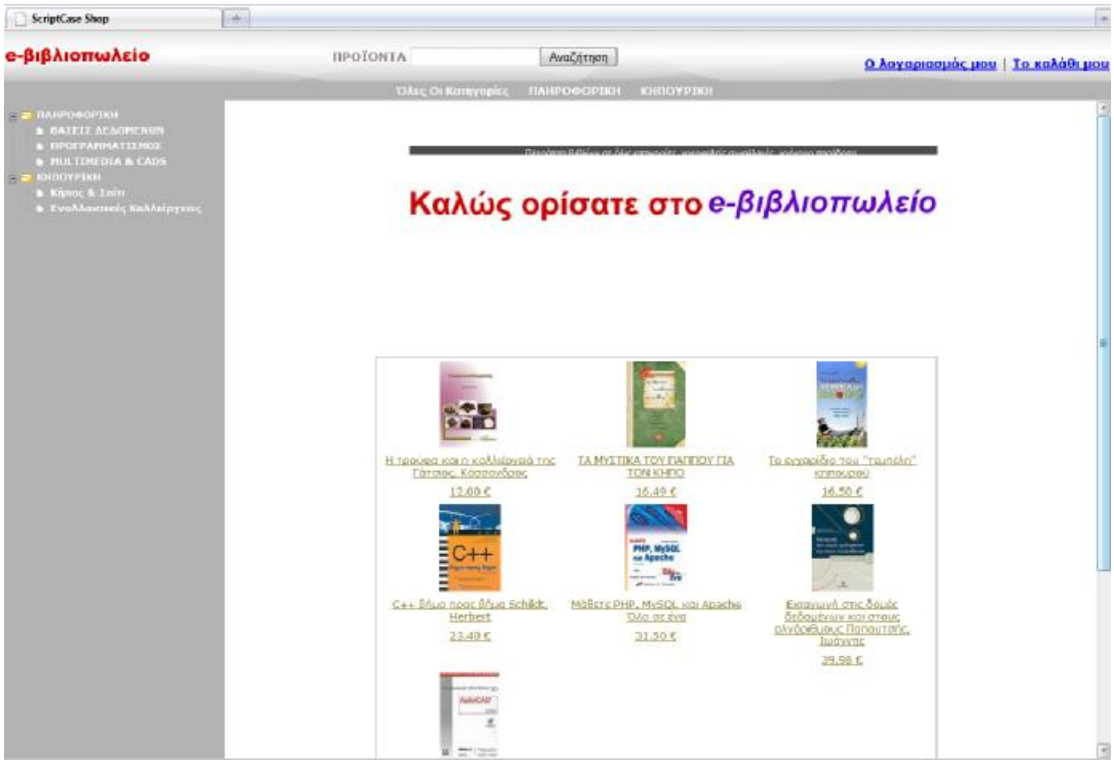
# 9 Εγχειρίδια χρήσης

Σε αυτήν την ενότητα ακολουθεί η περιγραφή του εγχειριδίου χρήσης της εφαρμογής, το οποίο χωρίζεται σε δύο μέρη όσα και τα επίπεδα των χρηστών, πελάτης, και διαχειριστής.

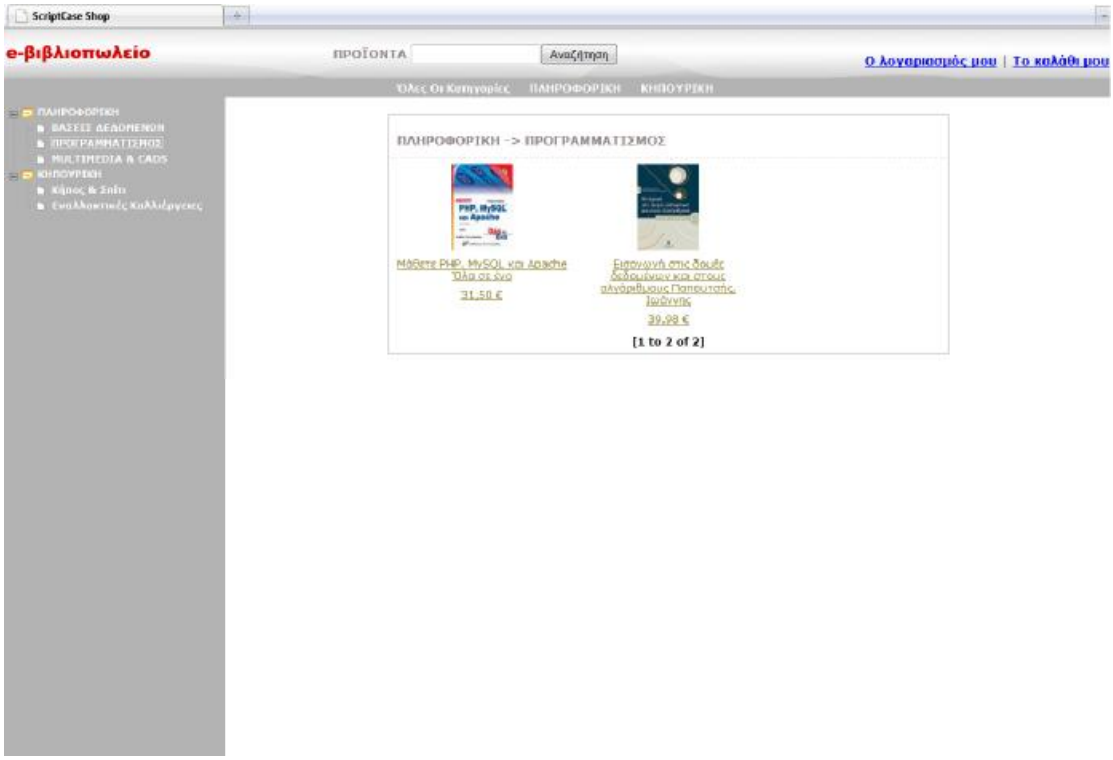
## 9.1 Εγχειρίδιο για τον απλό Χρήστη

### 9.1.1 Αρχική οθόνη

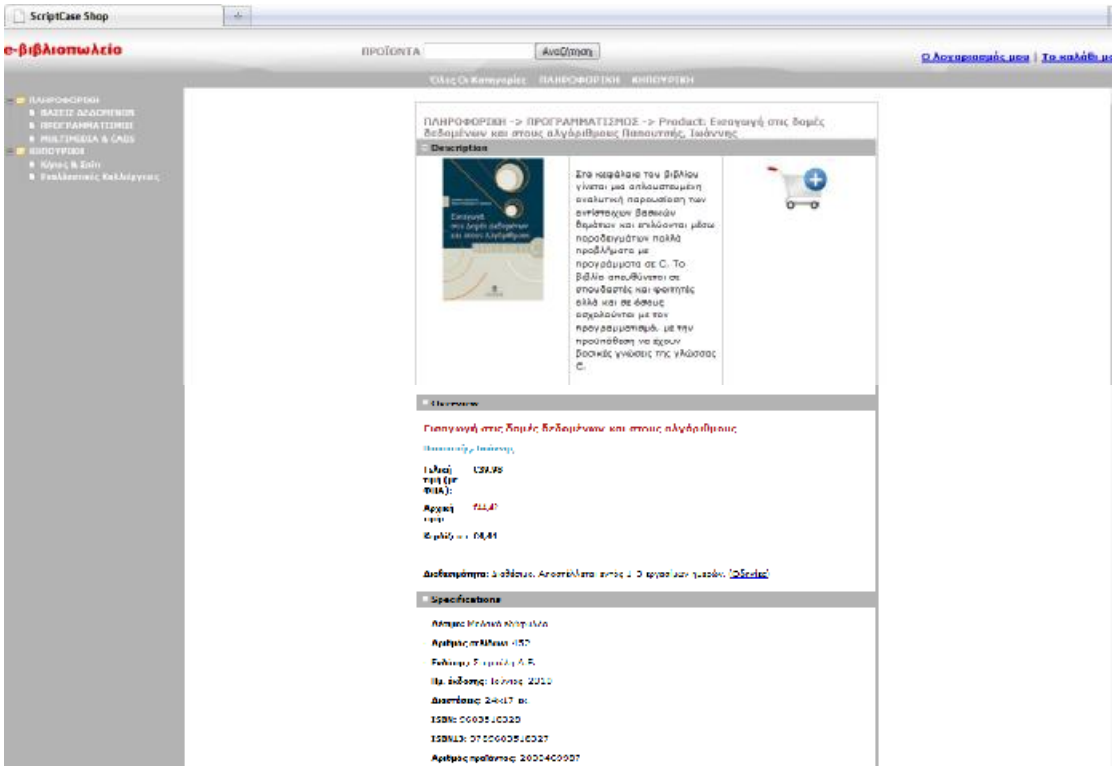
Η αρχική οθόνη εμφανίζεται με την έναρξη της εφαρμογής και φαίνεται στην παρακάτω εικόνα.



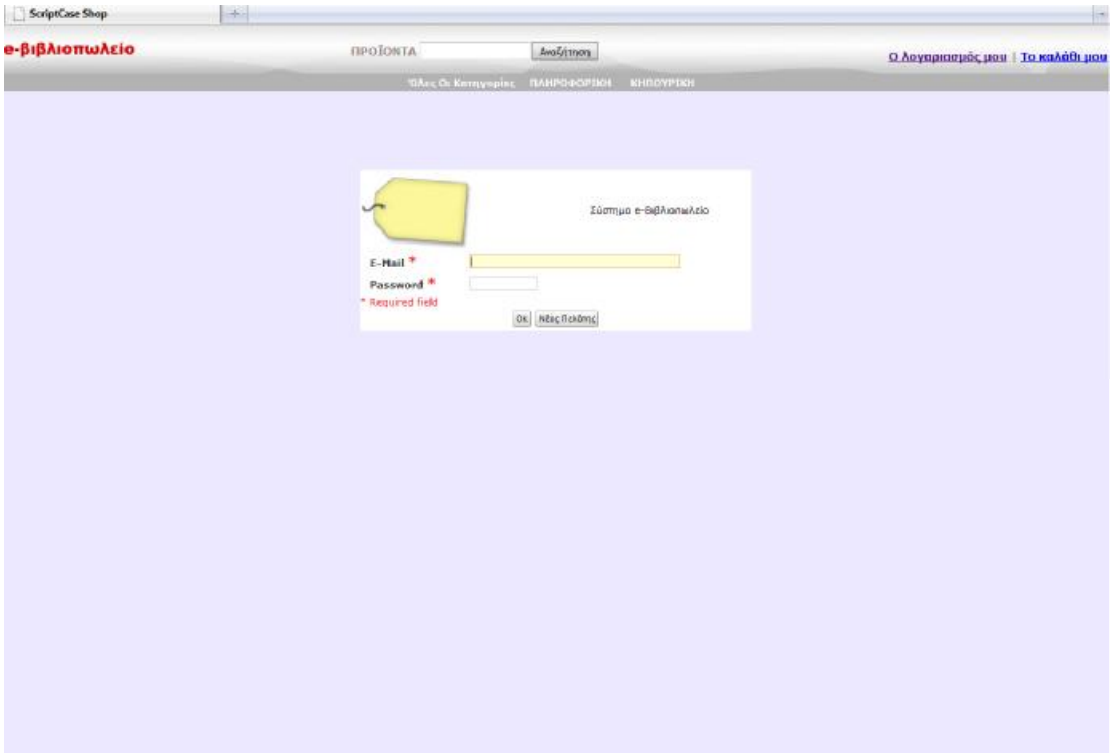

Στην αριστερή πλευρά φαίνονται οι κατηγορίες και οι υποκατηγορίες, στην κεντρική διάφορα τυχαία βιβλία και στην πάνω υπάρχει δυνατότητα αναζήτησης. Επίσης ο χρήστης έχει δυνατότητα να μπεί στον λογαριασμό του ή στο καλάθι.



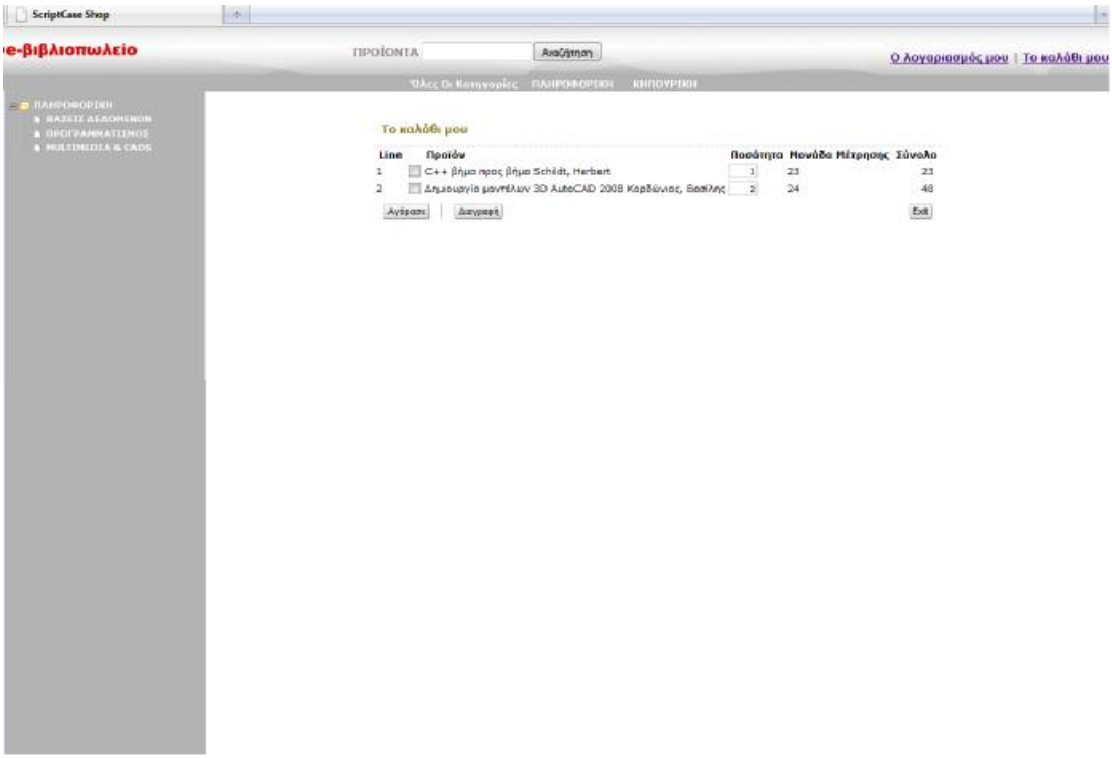

Ό χρήστης αφου διαλέξει κατηγορία ή υποκατηγορία, μπορεί να δει τα διαθέσιμα βιβλία.



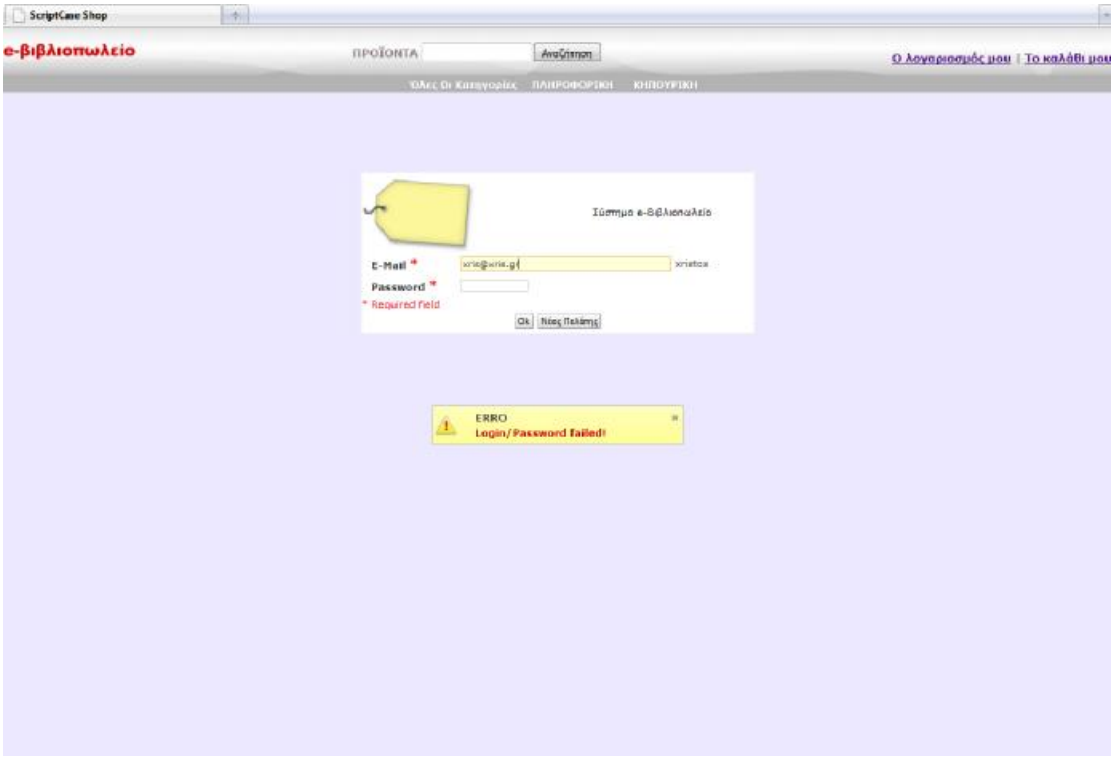

Αφού διαλέξει βιβλίο μπορεί να διαβάσει καποιά πράγματα για αυτό και να δει τα διάφορα χαρακτηριστικά του όπως τιμή, κωδικό κλπ.



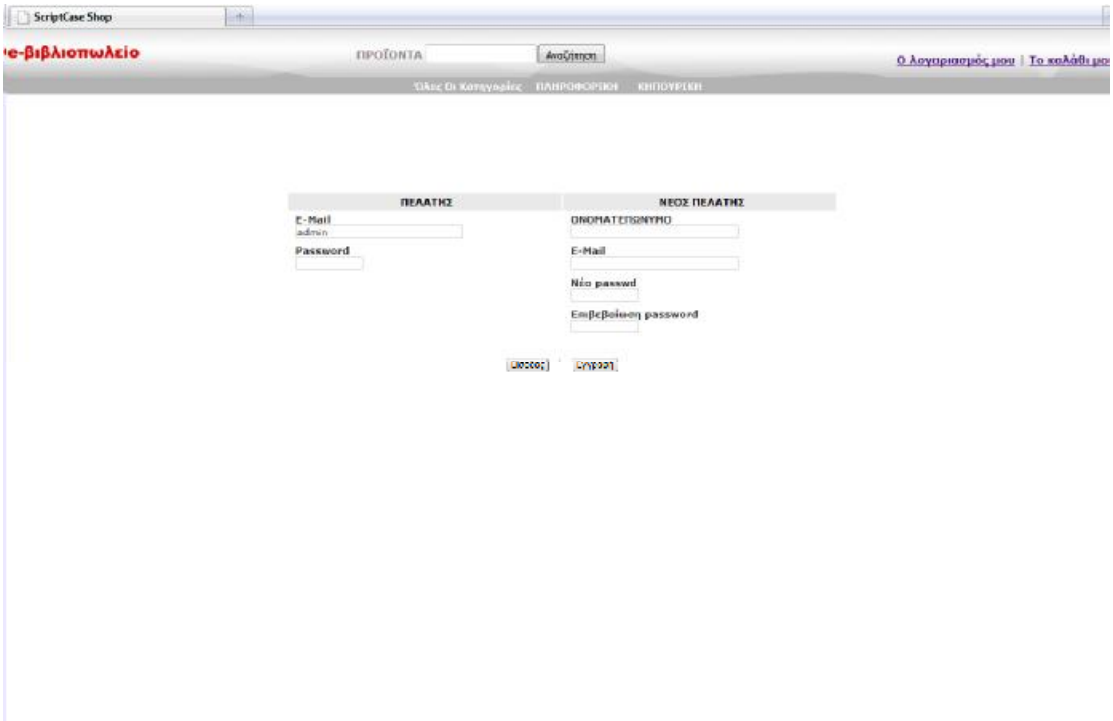

Αφού αποφασίσει να το αγοράσει μπορεί να συνδεθεί στο σύστημα από το κουμπί «ο λογαριασμός μου»




Στο καλάθι ο χρήστης μπορεί να δεί τα προιόντα που έχει διαλέξει και να διαλέξει την ποσότητα που επιθυμεί (εφοσόν υπάρχουν διαθέσιμα).



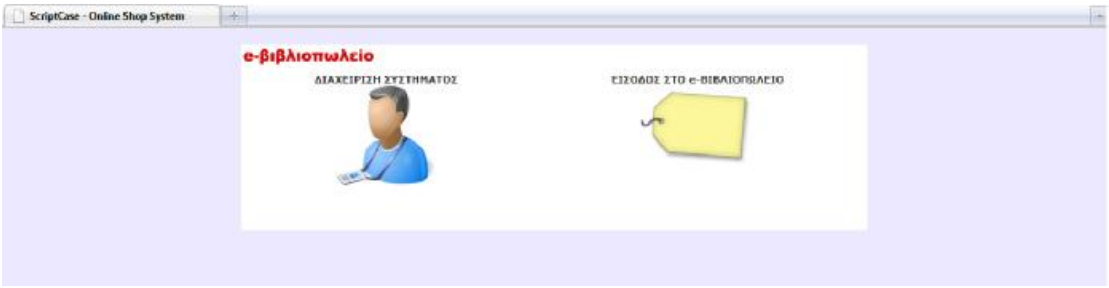

Το σύστημά αναγνωρίζει το χρήστη και ειδοποιεί σε περίπτωση ανεπιτυχούς πρόσβασης



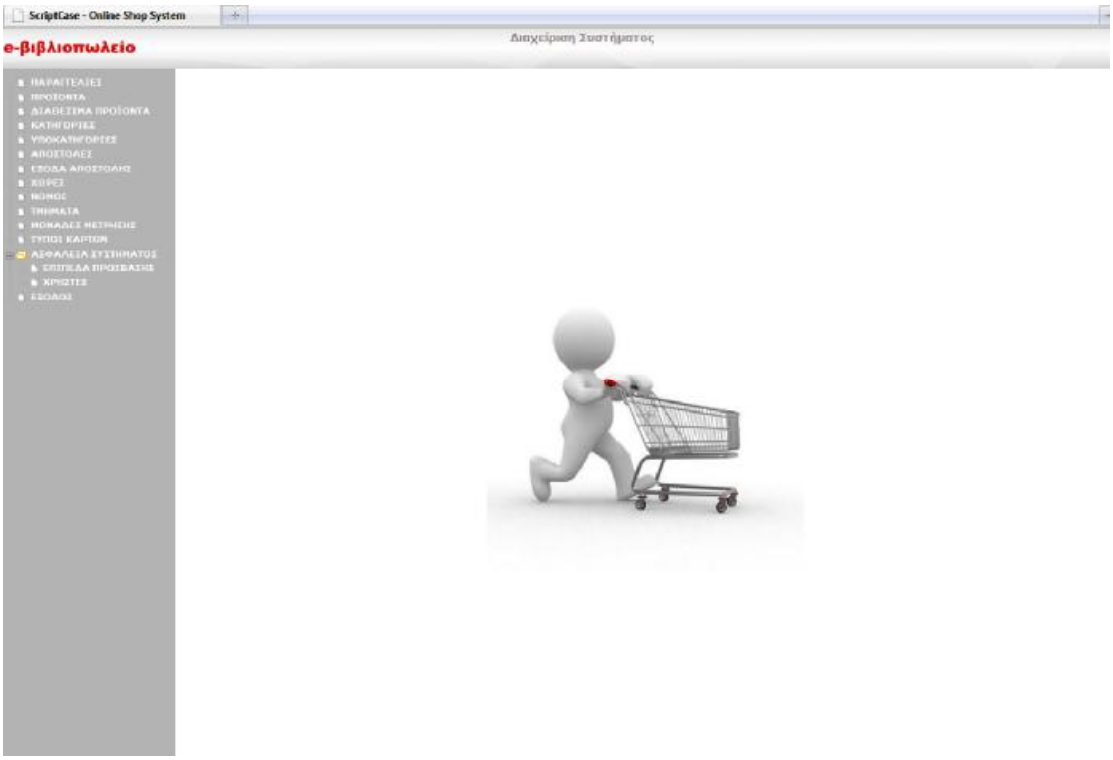

Επίσης ο χρήστης – επισκέπτης μπορει να γραφτεί (αν δεν είναι γραμμένος ) ή να συνθεδεί να δει την κατάσταση της παραγγελίας του.

## 9.2 Εγχειρίδιο Διαχειριστή

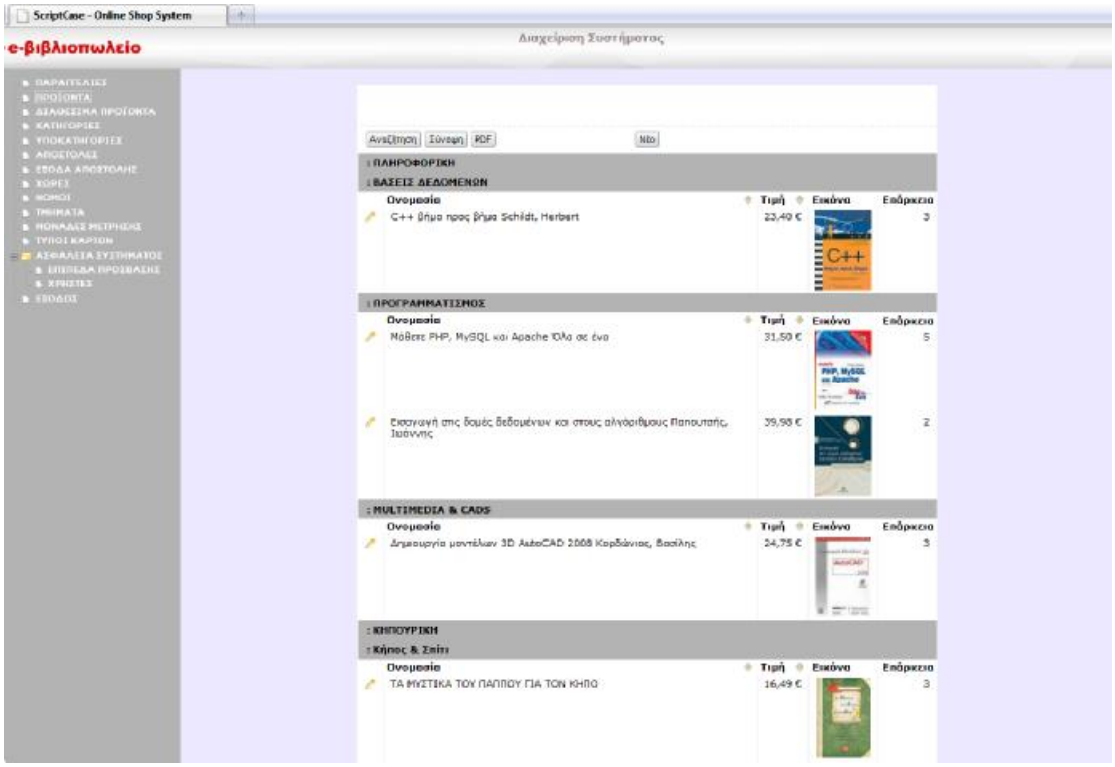
Ο διαχειριστής αφού πληκτρολογήσει έγκυρα, τα στοιχεία πρόσβασης του Όνομα Χρήστη και Κωδικό, στην φόρμα εισόδου, μεταβαίνει στην κεντρική σελίδα ελέγχου τις εφαρμογής. Σε αυτή την σελίδα της εφαρμογής, πρόσβαση έχει μόνο ο διαχειριστής.



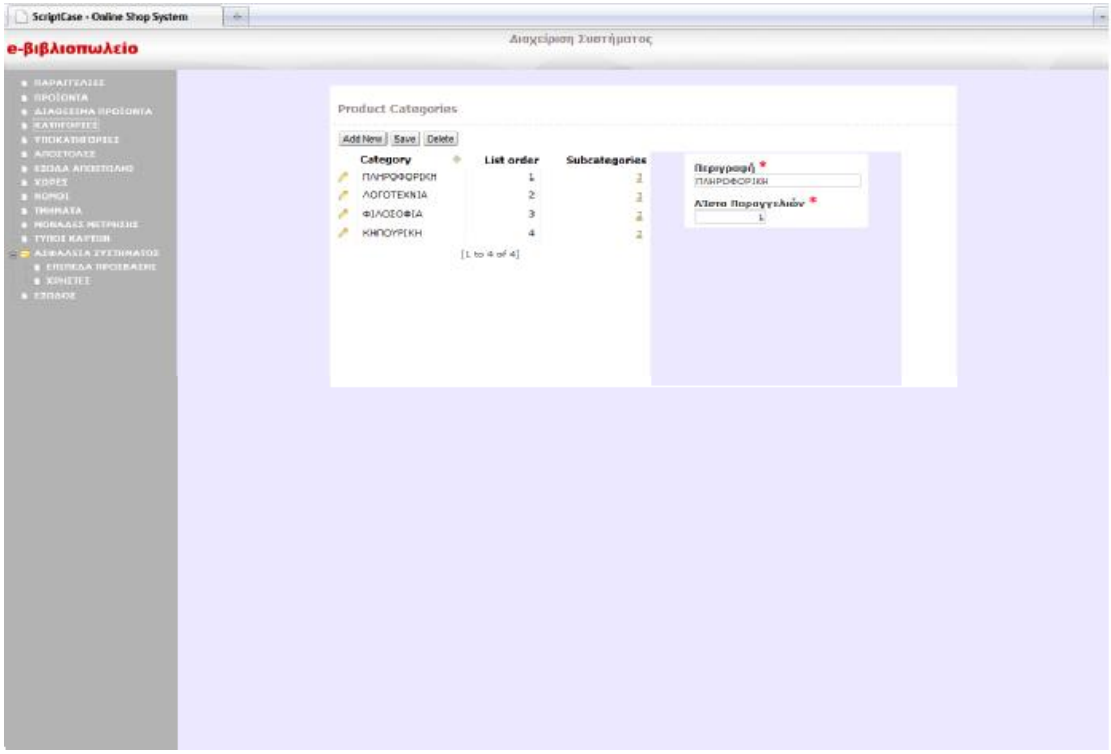

Ο διαχειριστής διαλέγει με ποιά ιδιότητα θα μπεί στο σύστημά.



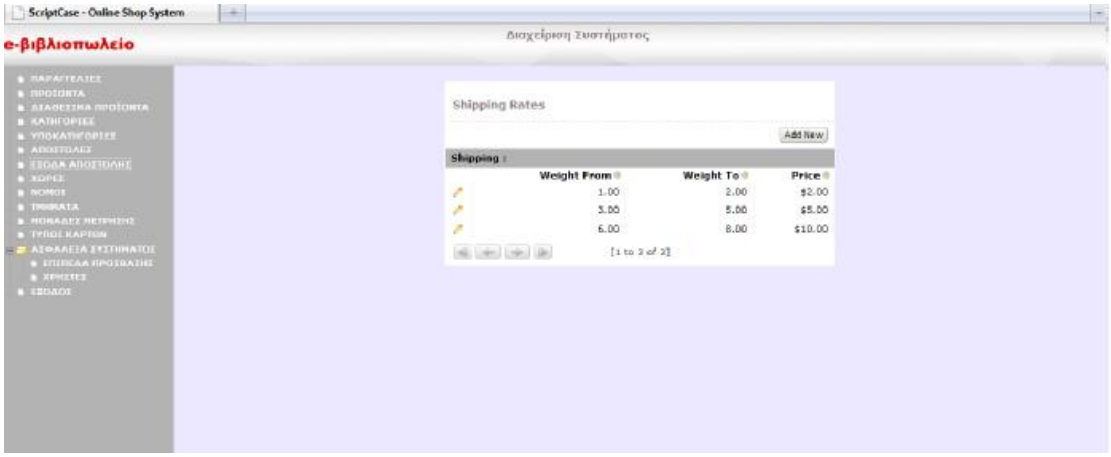

Ο διαχειριστής του συστήματος έχει την δυνατότητα να διαλέξει από μια πληθώρα επιλογών όπως να δει τις παραγγελίες, να τροποποιήσει τα προιόντα (διαθέσιμά και μη), τις κατηγορίες, τις αποστολές, τα έξοδα αποστολής, την χώρα, τις μονάδες μέτρησης κι άλλα.



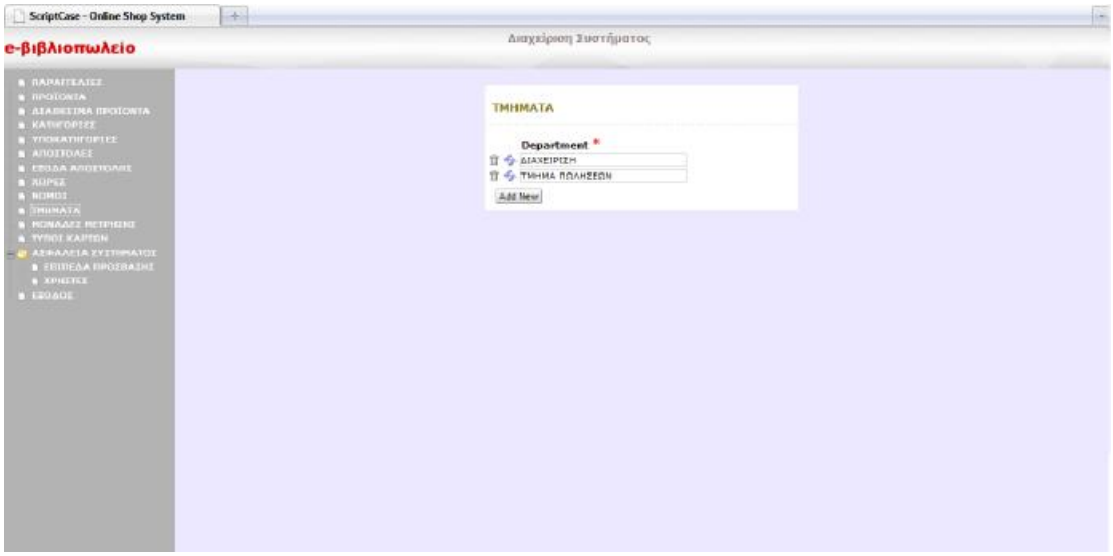

Στα προιόντα μπορεί να προσθαφαιρέσει βιβλία και να ορίσει την διαθεσιμότητα τους.



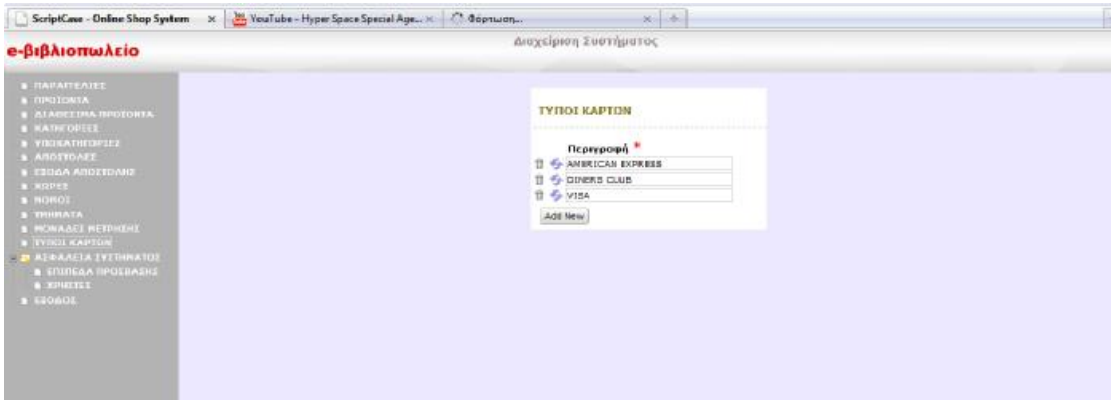

Στις κατηγορίες μπορεί να προσθαφαιρέσει κι άλλες, όπως και στις υποκατηγορίες.



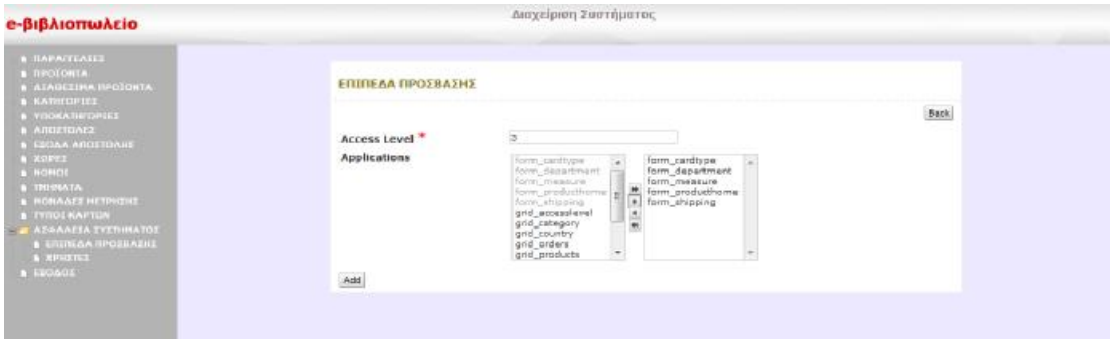

Στα έξοδα αποστολής να τα παραμετροποιήσει με βάση το βάρος.




Επίσης να ορίσει τα τμήματα της εταιρείας.




Μπορεί να δηλώσει τις διαθέσιμες πιστωτικές κάρτες.




Μπορεί να φτιάξει χρήστες με αντίστοιχα προνόμια και επίπεδα πρόσβασης.


Με βάση τα προηγούμενα οι χρήστες έχουν προσβασή στις φόρμες ανάλογα με τα προνόμια που τους έχουν παραχωρηθεί.

# 10 Συμπεράσματα

## 10.1 Εισαγωγικά

Με την παρούσα πτυχιακή πιστεύουμε ότι πετύχαμε τον αρχικό στόχο, δηλαδή την μελέτη όλων των μοντέλων ανάπτυξης λογισμικού με σκοπό την ανάλυση και υλοποίηση διαδικτυακής εφαρμογής ηλεκτρονικών Βιβλιοπωλείων.

Η εφαρμογή αυτή θα καλύπτει τις βασικές λειτουργίες που γίνονται για την αγορά ενός βιβλίου. Βέβαια διευκολύνθηκε η διαδικασία αγοράς που μέχρι στιγμής γινόταν με συμβατά μέσα (τηλέφωνο, κατ' ιδίαν κλπ). Αυτό έχει ως αποτέλεσμά την αύξηση της παραγωγικότητας, της επιχειρηματικότητας καθώς και την διευκόλυνση και επιτάχυνσή των διαδικασιών.

## 10.2 Μελλοντικές δυνατότητες

Η εφαρμογή προσφέρει (σε μελλοντικό επίπεδο) την δυνατότητα για:

- Συλλογή πληροφοριών όπως για παράδειγμα ιστορικό αγορών και προτιμήσεων, δημογραφικά στοιχεία κ.λ.π. και αξιοποίησή τους για την παροχή όσο το δυνατόν καλύτερων υπηρεσιών προς τους πελάτες.
- Διατήρηση πελατών μέσω καλύτερης εξυπηρέτησης
- Μετατροπή των επισκεπτών σε αγοραστές.
- Αξιολόγηση της μακροπρόθεσμης αξίας του πελάτη για την επιχείρηση (customer analysis).
- Συνεργασία με άλλες, μη ανταγωνιστικές εταιρείες και αγορά ή ανταλλαγή δεδομένων
- Ανάλυση εκστρατείας (ποιοι πελάτες ανταποκρίνονται, πόσο συχνά)
- Αξιοποίηση του «ιογενούς» (viral) marketing η οποία δίνει τη δυνατότητα στους πελάτες να στέλνουν σε γνωστούς τους e-mails,

Επίσης θα μπορούσε να προστεθεί μελλοντικά δυνατότητές όπως:

- Newsletters
- Διαφημιστικές εκστρατείες βασισμένες σε συγκεκριμένες εκδηλώσεις ή ημερομηνίες

- Response Management ήτοι αυτόματες εφαρμογές αποστολής απαντητικών e-mail για συγκεκριμένες ερωτήσεις ή περιστάσεις

## 10.3 Οφέλη

Όλα τα παραπάνω με σωστό συντονισμό και οργάνωση θα μπορούσαν να αποφέρουν 3 κύρια οφέλη:

1. Αύξηση των πωλήσεων
2. Μείωση κόστους
3. Αύξηση ευελιξίας στις αλλαγές της αγοράς.

### 10.3.1 Αύξηση πωλήσεων

Η αύξηση των πωλήσεων προκύπτει από:

- Νέους πελάτες
- Πελάτες που ξοδεύουν περισσότερο σε υπάρχοντα προϊόντα ή αγοράζουν νέα (βλέπε Cross Selling, Up Selling)
- Δημιουργία πιστών καταναλωτών που διατηρούνται για μεγαλύτερο χρονικό διάστημα.
- Επιτάχυνση της διαδικασίας πώλησης αφού μπορεί να απαιτούνται λιγότερες επαφές για την πώληση του προϊόντος – υπηρεσίας.
- Αύξηση πωλήσεων προϊόντων με μεγαλύτερο συντελεστή κέρδους
- Αύξηση πωλήσεων λόγω της αύξησης της ικανοποίησης του πελάτη από την:

- Βελτιωμένη ανταπόκριση σε απαιτήσεις για πληροφόρηση
- Καλύτερη ικανοποίηση των αναγκών του
- Άμεση παρακολούθηση της προόδου της παραγγελίας του
- Μεγαλύτερη ποικιλία προσφερόμενων υπηρεσιών
- Καλύτερη υποστήριξη

### 10.3.2 Μείωση κόστους

Η μείωση κόστους προκύπτει από:

- Στη λειτουργία ενός call center – κέντρου εξυπηρέτησης πελατών, μέσω της αυτοματοποίησης των διαδικασιών. Το μεγαλύτερο κόστος της λειτουργίας ενός call

center είναι το στελεχιακό δυναμικό. Το κόστος λειτουργίας ενός τέτοιου κέντρου μπορεί να μειωθεί μέχρι και 70%.

- Στη βελτίωση της αποτελεσματικότητας του direct mail.
- Στη βελτίωση της αποτελεσματικότητας της διαφήμισης.
- Στην αποτελεσματικότερη υποστήριξη των πωλήσεων μέσω της αυτοματοποίησης των προσφορών, της καλύτερης πληροφόρησης για τον πελάτη, της δημιουργίας προβλέψεων πωλήσεων κλπ.
- Στην αυξημένη ικανοποίηση της ομάδας πωλήσεων εφόσον αυτή απολαμβάνει αυξημένη υποστήριξη από την εταιρεία και μπορεί ευκολότερα και γρηγορότερα να υποστηρίξει τους πελάτες της.

### 10.3.3 Άμεση ανταπόκριση στις αλλαγές της αγοράς

Η άμεση ανταπόκριση στις αλλαγές της αγοράς προκύπτει:

Υπάρχουν συστήματα CRM, τα οποία μπορούν να ενσωματωθούν στην εφαρμογή και τα οποία βοήθησαν σημαντικά τις επιχειρήσεις να προσαρμοστούν σε σημαντικές επιχειρησιακές αλλαγές. Με την εγκατάσταση των συστημάτων CRM η επιχείρηση δεν ακολουθεί απλά τον ανταγωνισμό. Έχοντας τη δυνατότητα να γνωρίζει τους πελάτες και τις ανάγκες τους ανά πάσα στιγμή μπορεί να προσαρμόζει τα προϊόντα, την τιμολογιακή πολιτική της, να στοχεύει καλύτερα τη διαφήμιση της και να χρησιμοποιεί τα αποδοτικότερα κανάλια διανομής. Έτσι δημιουργεί τις τάσεις στον κλάδο της, κρατώντας τους πελάτες της πιστούς σ’ αυτήν.

### 10.3.4 Άλλα οφέλη

Άλλα οφέλη που προκύπτουν είναι:

- Παροχή καλύτερης εξυπηρέτησης, σύμφωνα με τις ανάγκες των πελατών.
- Αύξηση της συνολικής αποδοτικότητας μέσω της υλοποίησης διαδικασιών αυτοματοποίησης.
- Αποτελεσματική λειτουργία Κέντρων Παροχής Βοήθειας και Τμήματος Πωλήσεων.
- Διασταυρούμενες πωλήσεις και δυνατότητα υλοποίησης ενεργειών προσωποποιημένου marketing («1 προς 1» marketing).
- Απλοποίηση διαδικασιών marketing και πωλήσεων.

- Προσδιορισμός νέων πελατών και ανάπτυξη βελτιωμένων προϊόντων / υπηρεσιών.
- Αύξηση εσόδων ανά πελάτη.
- Να μειώσει τα έξοδα marketing, καθώς και το κόστος συναλλαγής
- Να αυξήσει τις συναλλαγές ανά πελάτη
- Να βελτιώσει τη φήμη της εταιρείας (word-of-mouth).

## 10.4Σύνοψη

Μελλοντικά πιστεύουμε ότι υπάρχει η δυνατότητα ανάπτυξης περαιτέρω της εφαρμογής καθώς η εμπορευματοποίηση της με ανάλογα οφέλη τόσο για την εταιρεία – επιχείρηση που θα το αναλάβει όσο και για το καταναλωτικό κοινό. Με την κατάλληλη οργάνωση, συντονισμό και στήσιμο θα μπορούσε να φέρει σημαντικές αλλαγές στο χώρο των Βιβλιοπωλείων και να δώσει μία άλλη χροιά στο κλάδο αυτό.

# 11 Πηγές-Βιβλιογραφία

## Βιβλιογραφία


- Beck, K. (1999). *Extreme Programming Explained, Embracing Change.* Addison-Wesley Professional.
- Ian Sommerville, M. K. (2008). *Software engineering.* Pearson Education Limited.
- Madachy, R. J. (2008). *Software Process Dynamics.* Wiley-IEEE Press.
- Madnick, T. K.-H. (1991). *Software Project Dynamics: An Integrated Approach.* Prentice-Hall.
- Multithreading. (2007 ). *Schildt, Herbert.* ΓΚΙΟΥΡΔΑΣ Μ.
- Per Kroll, P. K. (2003). *The rational unified process made easy: a practitioner's guide to the RUP*. Addison-Wesley.
- Βεσκούκης, Β. (2000). *Τεχνολογία Λογισμικού Ι.* ΠΑΤΡΑ: ΕΑΠ.
- Χρυσογιός. (2006). *Βιβλιοπωλείο και τεχνολογία.* Αθήνα: Ελληνικά Γράμματα
- Melony, J. C.(2000) *PHP, MySql και Apache.* Μ.Γκιούρδας.


## Πηγές από διαδίκτυο


- *1keydata.com*. (n.d.). Retrieved from http://www.1keydata.com/sql/sql.html : http://www.1keydata.com/sql/sql.html (last access 13/11/2010)
- *Dreamweaver*. (n.d.). Retrieved from http://www.adobe.com/products/dreamweaver/: http://www.adobe.com/products/dreamweaver/(last access 12/11/2010)

- *en.wikipedia.org*. (n.d.). Retrieved from http://en.wikipedia.org/wiki/Simple_API_for_XML: (last access 23/1/2011)
- *en.wikipedia.org*. (n.d.). Retrieved from http://en.wikipedia.org/wiki/Web_service: http://en.wikipedia.org/wiki/Web_service (last access 4/1/2011)
- *http://www.sql.org* . (n.d.). Retrieved from http://www.sql.org : http://www.sql.org (last access 10/10/2010)
- *Scriptcase*. (n.d.). Retrieved from www.scriptcase.net: www.scriptcase.net (last access 28/12/2010)
- *Sql*. (n.d.). Retrieved from wikipedia: http://www.en.wikipedia.org/wiki/SQL (last access 3/11/2010)
- *why-mysql*. (n.d.). Retrieved from www.mysql.com: http://www.mysql.com/why-mysql/ (last access 13/1/2011)
- *www.coder.gr*. (n.d.). Retrieved from http://www.coder.gr/article.php?story=20060704215418108: http://www.coder.gr/article.php?story=20060704215418108 (last access 14/11/2010)
- *www.it.uom.gr*. (n.d.). Retrieved from http://www.it.uom.gr/project/soap/Theory/architecture.html: http://www.it.uom.gr/project/soap/Theory/architecture.html (last access 13/11/2010)
- *www.users.hol.gr*. (n.d.). Retrieved from http://users.hol.gr/~bat_geor/js/part1/part1.htm: http://users.hol.gr/~bat_geor/js/part1/part1.htm (last access 10/12/2010)
- *www.w3schools.com*. (n.d.). Retrieved from http://www.w3schools.com/webservices/default.asp: http://www.w3schools.com/webservices/default.asp (last access 24/10/2010)