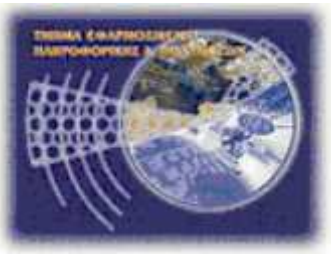
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων



Πτυχιακή εργασία

Ανάπτυξη παιχνιδιού πολλών παικτών

Τζεδάκης Χαράλαμπος (Α.Μ. 1658)

Ροδινός Ιωσήφ (Α.Μ. 1939)

Επιβλέπων Καθηγητής: Παπαδάκης Νικόλαος

Επιτροπή αξιολόγησης:

Ημερομηνία Παρουσίασης:

Περιεχόμενα

Περιεχόμενα 3
Abstract 5
Περίληψη 6
1 Εισαγωγή 7
1.1 Περίληψη 7
1.2 Κίνητρο για την διεξαγωγή της εργασίας 7
1.3 Στόχος εργασίας 7
1.4 Δομή εργασίας 7
2 Τεχνολογίες που χρησιμοποιήθηκαν 8
2.1 Java 8
2.1.1 Servlets 8
2.2 APACHE TOMCAT 16
2.3 Android 17
2.4 Σχεσιακές βάσεις δεδομένων 20
2.4.1 Εισαγωγή στις Βάσεις Δεδομένων 20
2.5 Εφαρμογές ανοικτού κώδικα 36
2.6 Eclipse 38
2.7 Netbeans 39
3 Εργαλεία που χρησιμοποιήθηκαν 40
3.1 WampServer 40
3.1.1 Εγκατάσταση προγραμμάτων 40
3.2 NetBeans 42
3.2.1 Εγκατάσταση προγραμμάτων 42
3.3 Android SDK 47
3.3.1 Εγκατάσταση προγραμμάτων 47
3.3.2 Εγκατάσταση plug-in 53
3.3.3 Δημιουργία project (χωρίς το command line) 53
3.3.4 Εκτέλεση project (Run on a Real Device και Run on the emulator) 54
3.3.5 Τρέχοντας στον Emulator 55
3.4 JSON 56
4 Ανάλυση εφαρμογής 57
4.1 Γενικά 57
4.2 Βάση δεδομένων 57
4.3 Ανάλυση server 58
4.4 Ανάλυση Εφαρμογής Android 66

4.4.1 Κλάση Game 67
4.4.2 Κλάση Player 70
4.4.3 Κλάση AsyncHttpRequest 72
4.4.4 Κλάση TPoint 73
4.4.5 Κλάση TicTacToeGame 74
4.4.6 Κλάση TicTacToeApp 75
4.4.7 Κλάση Symbol 75
4.4.8 Activity Login 75
4.4.9 Activity Register 76
4.4.10 Activity Join 77
4.4.11 Activity Main 78
5 Συμπεράσματα 80
6 Βιβλιογραφία 81

Abstract

This thesis aims to develop an online web application environment in Android. The application allows mobile phone owners with the particular operating system to compete in the game "tic tac toe". The game is played between two players who can compete either through a device or network using each different device.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως στόχο την ανάπτυξη μιας διαδικτυακής ηλεκτρονικής εφαρμογής σε περιβάλλον Android. Η εφαρμογή επιτρέπει στους κατόχους κινητών τηλεφώνων με το συγκεκριμένο λειτουργικό σύστημα να ανταγωνίζονται στο παιχνίδι «τρίλιζα». Το παιχνίδι παίζεται μεταξύ δύο παικτών οι οποίοι μπορούν να ανταγωνίζονται είτε με χρήση μιας συσκευής είτε μέσω δικτύου χρησιμοποιώντας ο καθένας διαφορετική συσκευή.

1 Εισαγωγή

1.1 Περίληψη

Τα τελευταία χρόνια βιώνουμε μια επανάσταση των έξυπνων κινητών τηλεφώνων (Smartphone's), τα όποια ενσωματώνουν πλέον δυνατότητες, οι οποίες δεν υπάρχουν στα συμβατικά κινητά. Περιλαμβάνουν ολοκληρωμένο λειτουργικό σύστημα, ισχυρό επεξεργαστή ικανό για εκτέλεση πολύπλοκων και χρονοβόρων υπολογισμών δυνατότητα σύνδεσης στο διαδίκτυο είτε μέσω ασύρματων δικτύων (Wi-Fi) είτε δικτύων τρίτης γενιάς ,κάμερα υψηλής ανάλυσης , συστήματα εντοπισμού θέσης κ.α. Τα λειτουργικά συστήματα των συσκευών αυτών ποικίλουν , με το Android της Google να περιλαμβάνεται τη στιγμή αυτή στο μεγαλύτερο ποσοστό των Smartphone's της αγοράς. Μάλιστα επίσημα στοιχειά δείχνουν ότι η ιστοσελίδα της Google, Play Store, που περιέχει τις εφαρμογές για android περιλαμβάνει αυτή τη στιγμή περισσότερες από 700,000 εφαρμογές. Στο πλαίσιο αυτό σκοπός της παρούσας εργασίας είναι η ανάπτυξη μιας εφαρμογής που προορίζεται για τα παραπάνω κινητά τηλεφωνά. Βεβαία η εφαρμογή μπορεί να εκτελεστεί όχι μόνο από κινητά αλλά από οποιαδήποτε ηλεκτρονική συσκευή φέρει το συγκεκριμένο λειτουργικό.

1.2 Κίνητρο για την διεξαγωγή της εργασίας

Στην εποχή που ζουλώ το διαδίκτυο έχει γίνει ένα μΈγελο κορεάτη της ζωής μας, όπου απευθυνόσαστε για συλλογή πληροφοριών, ανεύρεση διαφόρων αντικειμένων για αγορά, έγινε χώρος συνάντησης με φίλους και άγνωστους από όλο τον Κάσο και Ακόλα και είναι χώρος διασκέδασης με τις διάφορες εφαρμογές που υπάρχουν.

Έτσι θα ήταν χρήσιμο και την ιδία στιγμή διασκεδαστικό να υπάρχει μια εφαρμογή ενός κλασικού παιχνιδιού σε ψηφιακή μορφή που οι παίκτες θα μπορούν να διαιρούν και να ανταγωνίζονται μεταξύ τους .

1.3 Στόχος εργασίας

Σκοπός της παρούσας εργασίας είναι η εριουργία μώας εφαρμογής για την πρόσβαση από παίκτες. Έχει σχεδιαστεί και δομηθεί με τέτοιο τρόπο έτσι ώστε να δίνει τη δυνατότητα πρόσβασης από παντού εφόσον υπάρχει σύνδεση στον διακομιστεί (server). Στόχος της εργασίας αυτής είναι να μάθολε για το σχεδιασμό και προγραμματισμό μώας εφαρμογής για κινητά τηλέφωνο χρησιμοποιώντας τις πλατφόρμες netbeans και eclipse και η κατανόηση του τρόπου λειτουργίας τους. Επίσης, είναι και η κατανόηση κάποιων βασικών γλωσσών προγραμματισμού που απαιτούνται για την εριουργία εφαρμογών π.χ. Java.

1.4 Δομή εργασίας

2 Τεχνολογίες που χρησιμοποιήθηκαν

2.1 Java

2.1.1 Servlets

Η πρώτη τεχνολογία που θα περιγράψω είναι τα servlets. Είναι ένα είδος προγράμματος που μπορεί να εκτελεστεί όταν ζητηθεί από έναν browser. Τα servlets είναι γραμμένα σε Java, φορτώνονται στον διακομιστή Web και εκτελούνται όταν μία κατάλληλη εντολή HTTP που ζητά την εκτέλεση τους λαμβάνεται από τον διακομιστή. Τα servlets έχουν ορισμένα σημαντικά χαρακτηριστικά:

- Μπορούν να εκτελεστούν χωρίς αλλαγές σε διάφορους τύπους διακομιστών.
- Από την στιγμή που τα servlets είναι γραμμένα σε Java έχουν πρόσβαση σε διάφορα εργαλεία της Java όπως η CORBA, RMI, εργαλεία ασφαλείας της Java και εργαλεία σύνδεσης βάσεων δεδομένων.
- Τα servlets είναι εγκατεστημένα στην μνήμη. Με τον προγραμματισμό CGI σε γλώσσες όπως η Perl, κάθε φορά που ένα καινούργιο αίτημα προωθείται από τον Web browser μια καινούργια επεξεργασία πρέπει να ξεκινήσει και να τερματιστεί με την λήξη της σύνδεσης και το κατάλληλο πρόγραμμα πρέπει να φορτωθεί και να ξεφορτωθεί από τη μνήμη. Αυτό σημαίνει ότι το φορτίο μπορεί να είναι μεγάλο για τον διακομιστή Web.
- Από την στιγμή που τα servlets βασίζονται στην Java περιλαμβάνουν όλα τα χαρακτηριστικά ασφαλείας της γλώσσας.
- Τα servlets μπορούν να διατηρήσουν την κατάσταση τους, να διατηρήσουν κάποιο είδος μνήμης, ανάμεσα στα αιτήματα, κάτι που, στα πρώτα βήματα του Παγκόσμιου Ιστού, ήταν δύσκολο. Αυτό σημαίνει ότι τα servlets μπορούν να θυμηθούν δεδομένα και λεπτομέρειες ενός προηγούμενου αιτήματος.
- Το αντικειμενοστραφές μοντέλο της Java βοηθά τον κώδικα της να είναι πιο κομψός από κώδικα γραμμένο σε άλλες γλώσσες όπως η C και η Perl. Αν εξετάσετε τα προγράμματα Perl CGI θα δείτε, συχνά, ότι είναι τεράστια, μονοκόμματα και δυσνόητα. Η προσανατολισμένη στ' αντικείμενα φύση της Java παρέχει επίσης και την δυνατότητα μεγαλύτερης επαναχρησιμοποίησης κώδικα.
- Τα servlets, επειδή ενσωματώνονται καλά με τις τεχνολογίες ασφάλειας που σχετίζονται με την Java, μπορούν να διαμορφωθούν ώστε να παρέχουν υψηλά επίπεδα ασφαλείας.

Πριν να εξετάσουμε τα servlets πιο λεπτομερώς αξίζει να θυμηθούμε το πως οι διακομιστές Web επεξεργάζονται τα αιτήματα. Ένας browser θα εκδώσει ένα αίτημα σε HTTP, για παράδειγμα ζητώντας μία ιστοσελίδα. Αυτό αποστέλλεται στον διακομιστή Web που ερμηνεύει το αίτημα και εκτελεί κώδικα, ο οποίος επιστρέφει μία σελίδα HTML. Η επεξεργασία που εκτελείται από τον διακομιστή εξαρτάται από την λειτουργικότητα της ιστοσελίδας που ζητήθηκε. Για παράδειγμα, ο κώδικας που εκτελείται μπορεί:

Να επιστρέψει μία ιστοσελίδα χωρίς να την αλλάξει καθόλου.

Να επιστρέψει μία ιστοσελίδα αφού την μορφοποιήσει, για παράδειγμα να εισάγει κάποιο δυναμικό περιεχόμενο όπως τιμές μετοχών και εμπορευμάτων.

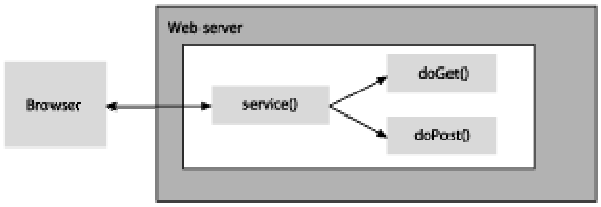
Να επεξεργαστεί μία φόρμα, να συνδεθεί με μία βάση δεδομένων ώστε να ανακτήσει δεδομένα και στην συνέχεια να κατασκευάσει μία σελίδα βασιζόμενη σ' αυτές τις λεπτομέρειες.

2.1.1.1 Μηχανισμοί Servlet

Ένα servlet δεν περιέχει κύρια μέθοδο. Ένα servlet πρέπει να υλοποιεί την διεπαφή javax.servlet.Servlet. Για ένα servlet που προορίζεται για το Web και το οποίο θα χρησιμοποιεί HTTP, αυτό συνήθως γίνεται φτιάχνοντας μια υποκλάση της HttpServlet. Όταν το κατάλληλο αίτημα λαμβάνεται από τον διακομιστή Web εκτελείται ανάλογα με το αίτημα η κατάλληλη μέθοδος της κλάσης. Για να υλοποιήσει ο προγραμματιστής την απαραίτητη λειτουργικότητα πρέπει να γράψει τον κώδικα αυτών των μεθόδων.

Σ' έναν διακομιστή Web η εσωτερική μέθοδος / λειτουργία που εκτελείται, όταν μεταβιβάζεται μία εντολή HTTP, ονομάζεται υπηρεσία. Αυτή καλεί άλλες μεθόδους ανάλογα με την μορφή HTTP της εντολής. Για παράδειγμα, αν ληφθεί μία εντολή GET, τότε εκτελείται η μέθοδος doGet. Οποιαδήποτε επεξεργασία σχετική μ' αυτήν την εντολή εισάγεται στην συνέχεια στην μέθοδο. Αυτό φαίνεται στο Σχήμα 7.2 που περιγράφει επίσης την μέθοδο doPost που αντιστοιχεί στην εντολή HTTP POST. Η εντολή GET διαβιβάζεται στον διακομιστή Web με το πάτημα ενός συνδέσμου, μ' ένα URL της μορφής http://server address/servlets/example, όπου servlets είναι ένας κατάλογος στον οποίο αποθηκεύονται servlets.

Σαν ένα παράδειγμα απλού servlet δείτε τον παρακάτω κώδικα. Αυτό που κάνει είναι να ανταποκρίνεται σε μία εντολή GET με την αποστολή στον browser, που ξεκίνησε την εντολή, μίας πολύ απλής σελίδας HTML. Αυτή η σελίδα απλά θα εμφανίσει το μήνυμα 'Hello there'.



Δημιουργείται μια υποκλάση της κλάσης HttpServlet και προσδιορίζεται η μέθοδος doGet. Η μέθοδος doGet έχει δύο ορίσματα: το πρώτο είναι ένα αντικείμενο που αντιπροσωπεύει το αίτημα που λαμβάνεται από τον browser, το δεύτερο είναι ένα αντικείμενο που αναπαριστά την απάντηση που στέλνεται πίσω στον browser.

Το πρώτο πράγμα που κάνει το servlet είναι να ενημερώσει το αντικείμενο απάντησης rp ότι η απάντηση που θα σταλεί πίσω στον πελάτη θα έχει τη μορφή HTML κειμένου. Αυτό αποτελεί μέρος της συναλλαγής μεταξύ του browser και του διακομιστή Web που αναφέραμε στο προηγούμενο κεφάλαιο. Η επόμενη γραμμή δηλώνει ένα αντικείμενο PrintWriter και αρχικοποιεί τη σχετική μεταβλητή με το σχετικό αντικείμενο που είναι μέρος του rp. Μετά από αυτό, ο κώδικας HTML της σελίδα που πρέπει να σταλεί εγγράφονται στο αντικείμενο PrintWriter. Αυτό ισοδυναμεί με το να σταλεί ο HTML κώδικας στον browser. Εκεί τελικά θα εμφανιστεί ένα απλό μήνυμα χρησιμοποιώντας σε επικεφαλίδα <H3> της γλώσσας.

Για να μπορεί ο διακομιστής να χρησιμοποιήσει τον κώδικα αυτό, θα πρέπει πρώτα να μεταγλωττιστεί και το αρχείο κλάσης που θα δημιουργηθεί να τοποθετηθεί σ' έναν κατάλογο κατάλληλο για servlets στον διακομιστή. Το servlet είναι έπειτα προσβάσιμο μέσω του URL του. Το επόμενο παράδειγμα είναι λίγο πιο περίπλοκο: Αυτή την φορά θα παρουσιάσω τον κώδικα ενός μετρητή επισκέψεων.

2.1.1.2 Μετρητές επισκέψεων

Οι μετρητές επισκέψεων σε μια ιστοσελίδα δίνουν τον αριθμό των επισκεπτών που η σελίδα έχει δεχτεί. Πεπειραμένοι σχεδιαστές Web τους αποδοκιμάζουν. Τους θεωρούν ως την ηλεκτρονική μορφή των γούνινων ζαριών που είναι κρεμασμένα στα παρμπρίζ μερικών αυτοκινήτων. Υπάρχουν διαθέσιμοι εκατοντάδες διαφορετικοί τύποι μετρητών, και οι περισσότεροι από αυτούς είναι CGI scripts.

Εδώ, η μόνη διαφορά είναι η ύπαρξη μιας μεταβλητής μετρητή που αρχικοποιείται στην τιμή μηδέν και έπειτα αυξάνεται κάθε φορά που κάποιος επισκέπτεται την ιστοσελίδα.

Αξίζει να επαναλάβουμε ότι παρόλο που ο κώδικας στα προηγούμενα παραδείγματα είναι σχεδόν τετριμμένος, υπάρχει η πιθανότητα συγγραφής κώδικα περισσότερο περίπλοκου, για παράδειγμα κώδικα που επεξεργάζεται μία φόρμα η οποία περιλαμβάνει κάποιο αίτημα, αντλεί στοιχεία από κάποια βάση δεδομένων, όπως έναν κατάλογο προϊόντων, και στην συνέχεια επιστρέφει ιστοσελίδες που αντιπροσωπεύουν τ' αποτελέσματα του αιτήματος.

2.1.1.3 Ο κύκλος ζωής του servlet

Το servlet έχει έναν καθορισμένο κύκλο ζωής:

- Το servlet αρχικοποιείται και φορτώνεται στην μνήμη του διακομιστή Web.
- Το servlet παραμένει στην μνήμη περιμένοντας τα αιτήματα των πελατών.
- Το servlet καταστρέφεται.

Servlets και αντικατάσταση συνδέσμων: μία εφαρμογή
Ένα από τα κύρια προβλήματα με τις ιστοσελίδες είναι ότι συχνά παραπέμπουν σε σελίδες που δεν υπάρχουν πια. Μία τεχνική για να ξεπεραστεί αυτό το πρόβλημα είναι η αποθήκευση εναλλακτικών URL για τους συνδέσμους μιας σελίδας σε μια βάση δεδομένων. Όταν μία σελίδα ζητείται από έναν browser, το servlet θα εξετάσει κάθε έναν από τους συνδέσμους για να δει αν είναι ακόμη ενεργοί. Αν δεν είναι τότε το URL του συνδέσμου αντικαθίσταται από ένα άλλο το οποίο είναι ενεργό.

Έχουμε ήδη δει με ποιο τρόπο ένα servlet ανταποκρίνεται στις εντολές ενός browser: με την εκτέλεση μεθόδων όπως η doGet. Για την αρχικοποίηση ενός servlet υπάρχει μία αντίστοιχη μέθοδος, η init. Αυτή εκτελείται αμέσως μετά το φόρτωμα του servlet στην μνήμη και θα εκτελέσει τις απαραίτητες ενέργειες αρχικοποίησης. Για να εκτελέσει την αρχικοποίηση η init πρέπει να έχει πρόσβαση στις παραμέτρους αρχικοποίησης. Υπάρχει μια μέθοδος για την προσπέλαση παραμέτρων αρχικοποίησης και είναι η getInitParameter. Αυτή παίρνει ένα string όρισμα το οποίο είναι το όνομα της παραμέτρου που ζητείται. Για παράδειγμα μπορεί να ζητηθεί το όνομα ενός αρχείου αρχικοποίησης ή η αρχική τιμή μιας μεταβλητής. Για να λειτουργήσει, η μέθοδος χρειάζεται να συνδεθεί μ' ένα αντικείμενο που καθορίζεται από από την κλάση ServletConfig, και το οποίο περιέχει πληροφορίες σχετικά με την παρούσα κατάσταση του διακομιστή στον οποίο το servlet έχει στηθεί.

Το όρισμα της μεθόδου init είναι ένα αντικείμενο κλάσης ServletConfig. Η init μπορεί να χρησιμοποιηθεί ώστε με τη μέθοδο getInitParameter να πάρουμε τις παραμέτρους αρχικοποίησης. Ένα παράδειγμα της χρήσης της init και των παραμέτρων ενεργοποίησης δίδεται παρακάτω: αποτελεί την λύση στο πρόβλημα του μετρητή

επισκέψεων που αναφέρθηκε προηγουμένως, ότι δηλαδή το servlet δεν μπορεί να κρατά την τιμή του μετρητή αφού κλείσει το servlet.

Η λύση είναι να τοποθετηθεί κώδικας μέσα στην μέθοδο init, ο οποίος θα φορτώνει κατά την αρχικοποίηση του servlet την τιμή που είχε ο μετρητής επισκέψεων την τελευταία φορά που έκλεισε το servlet.

Η μέθοδος init αρχικά αρχικοποιεί το αντικείμενο ServletConfig, το οποίο περιέχει τα δεδομένα αρχικοποίησης του servlet, καλώντας την αντίστοιχη μέθοδο στην κλάση HttpServlet (η υπερκλάση της κλάσης μέσα στην οποία βρίσκεται η μέθοδος init).

Στην συνέχεια, ο κώδικας εξάγει το όνομα του αρχείου στο οποίο είναι αποθηκευμένη η τιμή του μετρητή. Αυτό προσδιορίζεται από το όρισμα "PersistentFile". Ο κώδικας αυτός, που δεν φαίνεται, θα ανοίξει το αρχείο και θα εξάγει την τιμή του αποθηκευμένου ακέραιου αριθμού.

Αξίζει να τονιστεί ότι οι τιμές των παραμέτρων αρχικοποίησης παρέχονται από τον διαχειριστή του συστήματος κατά την ρύθμιση του διακομιστή Web. Αυτό μπορεί να γίνεται είτε από ένα πρόγραμμα που διαθέτει γραφικό περιβάλλον είτε γράφοντας κείμενο σε κάποιο απλό αρχείο ρυθμίσεων. Έτσι, για παράδειγμα, ο διαχειριστής μπορεί να δώσει την τιμή "Counter.txt" στην παράμετρο με το όνομα PersistentFile, όπου Counter.txt είναι το αρχείο το οποίο αποθηκεύεται ο μετρητής.

Υποστήριξη των servlets από διακομιστές
Όταν πρωτοεμφανίστηκαν τα servlets από την Sun, δεν υπήρχαν πολλοί διακομιστές που να τα υποστηρίζουν: στην αρχή ο μόνος διακομιστής Web που τα υποστήριζε ήταν ο Java Web Server της Sun. Αυτός ο διακομιστής ήταν εξ' ολοκλήρου γραμμένος σε Java αλλά δεν ήταν ιδιαίτερα δημοφιλής. Η κατάσταση βελτιώθηκε σημαντικά καθώς πολλοί διακομιστές αργότερα άρχισαν να υποστηρίζουν servlets άμεσα ή μέσω πρόσθετων πακέτων.

2.1.1.4 Το μόνο πρόβλημα που μένει προς επίλυση είναι η εγγραφή της τιμής του διαρκούς μετρητή στο αρχείο που τον κρατά. Πριν να το εξετάσουμε πρέπει να ρίξουμε μία ματιά στο τελικό μέρος του κύκλου ζωής του servlet: στο κλείσιμο. Ένα servlet κλείνει όταν ένας διακομιστής Web το "ξεφορτώνει". Η μέθοδος κλεισίματος (και πάλι μία μέθοδος που κληρονομείται από μία υπερκλάση της) πρέπει να περιέχει κώδικα που εκτελείται όταν το servlet κλείνει. Κανονικά αυτός ο κώδικας στη μέθοδο κλεισίματος θα απελευθερώσει όποιον πόρο κρατά το servlet έτσι ώστε αυτοί να μπορούν να χρησιμοποιηθούν και πάλι (πιθανώς από το ίδιο ή από άλλο πρόγραμμα). Θα περιέχει επίσης άλλο κώδικα που πρέπει να εκτελεστεί κατά τον τερματισμό του servlet. Στην περίπτωση του servlet που υλοποιεί τον μετρητή επισκέψεων για παράδειγμα, θα γράψει την τιμή του μετρητή στο αρχείο που χρησιμοποιείται για την αποθήκευσή του.

Στο σημείο αυτό έχουμε παρουσιάσει όλες τις σημαντικές μεθόδους του κύκλου ζωής των servlets. Τώρα, πρέπει να εξετάσουμε, με περισσότερες λεπτομέρειες, την λειτουργία των μεθόδων που επεξεργάζονται HTTP αιτήματα όπως τα POST και GET, που προέρχονται από φόρμες ή από συνδέσμους.

2.1.1.5 Μια φόρμα μπορεί να προκαλέσει είτε ένα POST είτε ένα GET αίτημα, ανάλογα με την τιμή της παραμέτρου METHOD που υπάρχει στο tag της

φόρμας. Ένας σύνδεσμος προκαλεί πάντα ένα GET αίτημα. Για να μπορέσετε να καλέσετε servlet από φόρμα, πρέπει στην παράμετρο ACTION του tag της φόρμας να ορίσετε το αρχείο που περιέχει τον κώδικα ενός servlet (θυμίζουμε πως τα servlets μπαίνουν όλα σε ένα συγκεκριμένο κατάλογο). Για παράδειγμα, το tag

<FORM METHOD = "POST" ACTION = "/servlets/formprocessor">

εισάγει μία φόρμα. Τα περιεχόμενα της οποίας θα επεξεργαστεί το servlet formprocessor το οποίο βρίσκεται στον κατάλογο *servlets*. Το αρχείο το οποίο περιέχει το servlet είναι το αρχείο class που προέκυψε από τη μεταγλώττιση του κώδικα του servlet. Όταν το κουμπί submit της φόρμας πατηθεί θα σταλεί στον διακομιστή μία εντολή POST (αφού αυτό έχει δηλωθεί στην παράμετρο METHOD) τότε, με το πάτημα του κουμπιού κατάθεσης με το όνομα ``Darrel' στο πρώτο κουτί και το όνομα 'Ince' στο δεύτερο κουτί, μία εντολή POST θα αποστέλλονταν στον διακομιστή με την στοιχειοσειρά

FirstName=Darrel&SurName=Ince

Όταν το servlet, που περιέχει την μέθοδο doPost, παραλάβει αυτό το μήνυμα πώς προχωρά; η ενέργεια που πρέπει να εκτελεστεί είναι η εξαγωγή των τιμών των αντικειμένων της φόρμας. Αυτό επιτυγχάνεται με την μέθοδο getParameter που περιλαμβάνεται στην κλάση HttpServletRequest.

Εδώ ο κώδικας παίρνει τις τιμές των πεδίων, ελέγχει μήπως είναι μηδενισμένες (αν είναι μηδενισμένες τότε ο χρήστης έχει ξεχάσει να γράψει κάποιο κείμενο μέσα στο κουτί) και στην συνέχεια εμφανίζει τις τιμές του πρώτου ονόματος και επιθέτου στην ιστοσελίδα που επιστρέφεται στον browser. Με δεδομένα τα παραπάνω εργαλεία μπορείτε ν' αναπτύξετε εφαρμογές ηλεκτρονικού εμπορίου που μπορούν να έχουν πρόσβαση σε διακομιστή βάσεων δεδομένων: η μέθοδος doPost, για παράδειγμα, μπορεί να διαβάσει δεδομένα που εισάγονται από έναν χρήστη, για παράδειγμα το όνομα ενός προϊόντος, και στην συνέχεια να εμφανίσει μία ιστοσελίδα που περιέχει λεπτομέρειες του προϊόντος - οι οποίες θα έχουν ανακτηθεί από μία βάση δεδομένων.

Ο παραπάνω κώδικας περιέχει δηλώσεις που επεξεργάζονται ένα λάθος χρήστη. Το servlet API περιλαμβάνει μεθόδους που επιτρέπουν στα λάθη HTTP και σε λάθη όπως το παραπάνω να διαβιβάζονται πίσω στον πελάτη. Για παράδειγμα, αν ο πελάτης ζητήσει έναν πόρο που δεν υπάρχει τότε είναι σχετικά εύκολο να προγραμματιστεί ένα servlet ώστε να επιστρέφεται το σταθερό μήνυμα λάθους 404.

Θα θυμάστε από το προηγούμενο κεφάλαιο ότι η απάντηση ενός διακομιστή Web αποτελείται από μία κεφαλίδα, που δίνει την απάντηση μαζί με ζεύγη στοιχειοσειρών χωρισμένων από στήλες, που παρέχουν περαιτέρω πληροφορίες.

υποδεικνύει ότι ένας πόρος έχει ανακτηθεί επιτυχώς (κώδικας κατάστασης 200),με πληροφορίες για τον πόρο και τον διακομιστή να παρέχονται από σειρές που περιέχουν ζεύγη στοιχειοσειρών χωρισμένων από στήλες.

Με όλα τα παραπάνω παραδείγματα το servlet έχει ρυθμιστεί να στέλνει αυτόματα τον κώδικα 200. Ωστόσο, αυτό μπορεί να παραμεριστεί από την χρήση της μεθόδου setStatus. Η μέθοδος αυτή, που βρίσκεται στην κλάση HttpServletResponse, μπορεί να θέσει την απάντηση σε οποιονδήποτε κώδικα λάθους. η μέθοδος έχει δύο μορφές:

setStatus(int)
setStatus(int, String)

Η πρώτη μεταβλητή απλά επιστρέφει έναν κώδικα λάθους και το σταθερό μήνυμα κειμένου που σχετίζεται με τον κώδικα λάθους που εμφανίζεται. η δεύτερη μεταβλητή επιστρέφει έναν κώδικα λάθους, αλλά παραμερίζει το σταθερό μήνυμα λάθους μ' ένα άλλο, που παρέχεται από τον προγραμματιστή. Η κλάση HttpServletResponseClass περιλαμβάνει στατικές τελικές δημόσιες σταθερές που προσδιορίζουν τα σταθερά HTTP μηνύματα, για παράδειγμα

HttpServletResponseClass.SC_OK

είναι η σταθερά που υποδεικνύει ότι μία HTTP εντολή έχει εκτελεστεί επιτυχώς και

HttpServletResponseClass.SC_MOVED_PERMANENTLY

υποδεικνύει ότι ένας πόρος έχει μεταφερθεί μόνιμα σ' έναν άλλο διακομιστή. Έτσι, η επίκληση της setStatus μεθόδου , που παρουσιάστηκε παραπάνω, θα επιστρέψει ένα μήνυμα 404, ότι ο πόρος - συνήθως μία ιστοσελίδα - δεν βρέθηκε.

2.1.1.6 Διάρκεια στον πελάτη

Στο Κεφάλαιο 6 ανέφερα ότι ένα από τα μεγαλύτερα προβλήματα με το πρωτόκολλο HTTP είναι ότι δεν εγκαθιστά μία διαρκή κατάσταση. Για παράδειγμα, δεν υπάρχει μνήμη παρελθόντων διαδράσεων. Πολλές από τις εφαρμογές ηλεκτρονικού εμπορίου απαιτούν να υπάρχει μνήμη. Ένα σταθερό παράδειγμα αποτελεί το καλάθι αγορών που διατηρεί κομμάτια που έχει αγοράσει ο επισκέπτης ενός ιστοχώρου λιανικού εμπορίου.

2.1.1.7 Μερικές λύσεις

Έχουν γίνει ορισμένα hacks τα οποία υπερνίκησαν αυτό το πρόβλημα μέσα στα πλαίσια των εφαρμογών ηλεκτρονικού εμπορίου.

Hacks και shims

Το hack, μερικές φορές γνωστό και ως shim, είναι ένα μπερδεμένο κομμάτι προγραμματισμού το οποίο υπερνικά τους οργανικούς περιορισμούς μιας τεχνολογίας. Το Internet έχει μεγαλώσει πολύ γρήγορα και, εξαιτίας αυτού, πολλές από τις εφαρμογές και τα εργαλεία που είχαν αναπτυχθεί δεν ήταν σε θέση να εφαρμόσουν άμεσα λειτουργίες που προέκυψαν εν μια νυκτί. Ωστόσο, οι περισσότερες από τις τεχνολογίες που αναπτύχθηκαν μπορούν να "στρωθούν" με προγραμματικά τεχνάσματα ώστε να συμπεριφέρονται όπως πρέπει. Πιθανότατα το καλύτερο παράδειγμα hack εμφανίζεται στις πρώτες εκδόσεις της σημειακής γλώσσας HTML. Η γλώσσα αυτή δεν παρείχε εργαλεία για ελεύθερο σχεδιασμό φόρμας, όπου κομμάτια κειμένου και εικόνες θα μπορούσαν να τοποθετηθούν επακριβώς σε μια σελίδα. Ωστόσο, παρείχε εργαλεία για την εμφάνιση πινάκων. Πολλοί σχεδιαστές HTML χρησιμοποίησαν αυτούς τους πίνακες για την παραγωγή σχεδιαστικών φορμών όπου ενότητες κειμένου εμφανίζονταν με μεγάλα, ενδιάμεσα, διαχωριστικά κενά.

Η πρώτη λύση σ' αυτό το πρόβλημα - που συχνά αναφέρεται ως πρόβλημα εντοπισμού ενότητας - είναι να ζητηθεί από τους χρήστες ενός ιστοχώρου να εγγραφούν

με το όνομά τους η κάποια άλλη ταυτότητα. Κάθε φορά που ο χρήστης εκτελεί μία ενέργεια, όπως την τοποθέτηση ενός αντικειμένου στο καλάθι αγορών, τότε η φόρμα που χρησιμοποιείται για την εφαρμογή αυτή επισυνάπτει την ταυτότητα του χρήστη στα δεδομένα που αποστέλλονται στον διακομιστή όταν εκτελείται η φόρμα. Το κύριο πρόβλημα αυτής της προσέγγισης είναι ότι κάθε φορά που ο χρήστης διαδρά με τον ιστοχώρο πρέπει να συνδέεται. Αυτή είναι η κυριότερη αιτία για το ότι οι χρήστες δεν επιστρέφουν ξανά στον ιστοχώρο: δεν επιθυμούν να εκτελέσουν μία χρονοβόρα διαδικασία μόνο και μόνο για να μπουν σε μια ιστοσελίδα. Επίσης, δεν θέλουν να απομνημονεύουν πολλούς κωδικούς πρόσβασης.

Ένας δεύτερος τρόπος για την προσπέλαση προβλημάτων περιορισμού της HTTP είναι το πέρασμα σελίδων του διακομιστή, οι οποίες περιέχουν κρυφά πεδία όπως κρυφά κουτιά κειμένου που, όταν ενεργοποιηθεί η φόρμα, ανταποκρίνονται με ένα συγκεκριμένο αναγνωριστικό. Ένα παράδειγμα HTML που κατασκευάζει ένα κρυφό πεδίο κειμένου φαίνεται παρακάτω:

<INPUT TYPE = "hidden" NAME = "itemsbought" VALUE ="'209087">

Αρχικά ο διακομιστής Web θα κατασκευάσει μία φόρμα χωρίς κρυμμένα στοιχεία. Σταδιακά ωστόσο, καθώς ο πελάτης διαδρά με τον διακομιστή, δημιουργείται μία σειρά κρυφών στοιχείων τα οποία αντιπροσωπεύουν την συνεργασία χρήστη-διακομιστή. Για παράδειγμα, κάθε στοιχείο μπορεί να περιλαμβάνει μία ταυτότητα του κομματιού συνενωμένη με τον χρόνο κατά τον οποίον αυτό αγοράστηκε (αυτό φαίνεται παρακάτω ως ο αριθμός των δευτερολέπτων που πέρασαν μια συγκεκριμένη στιγμή). Σταδιακά μία φόρμα θ' αρχίσει να έχει την μορφή

<INPUT TYPE = "hidden" NAME = "Hide1" VALUE = "MickyBook236678990">
<INPUT TYPE = "hidden" NAME = "Hide2" VALUE = "DumboDoll236679002">
<INPUT TYPE = "hidden" NAME = "Hide3" VALUE = "DonaldCoffee236679107">
<INPUT TYPE = "hidden" NAME = "Hide4" VALUE = "ArthurBook236679207">

Εδώ μία ομάδα τεσσάρων κρυφών αντικειμένων φόρμας HTML έχουν παραχθεί από τον διακομιστή σε μία σελίδα που επιστρέφεται στον πελάτη. Αυτό συνήθως ακολουθείται από "πραγματικά" οπτικά αντικείμενα όπως κουτιά κειμένου και κουμπιά ραδιοφώνου, που παροτρύνουν τον χρήστη για την επόμενη επιλογή. Όταν ο χρήστης επιλέξει κάτι, ένα μήνυμα

Hide1=MickyBook236678990&
Hide2=DumboDoll236679002&
Hide3=DonaldCoffee236679107&
Hide4=ArthurBook236679207&
ItemSelected=BarbyDoll

θα επιστραφεί από την φόρμα, όπου το κουτί κειμένου ονόματι *ItemSelected* θα είναι ένα από τα πραγματικά οπτικά αντικείμενα που χρησιμοποιούνται για την μεταφορά του αγορασμένου αντικειμένου. Αυτό, στην συνέχεια, επιδέχεται επεξεργασία από τον διακομιστή Web που επιστρέφει μία σελίδα η οποία περιέχει πέντε αόρατα αντικείμενα φόρμας και ούτω καθεξής. Όταν ο χρήστης ελέγχει τα ονόματα των συλλεχθέντων αντικειμένων, υπολογίζεται το κόστος και αποστέλλεται πίσω στον πελάτη μία σελίδα με το συνολικό ποσό ως επιβεβαίωση της αγοράς. Αυτό είναι ένα hack που δεν είναι ικανοποιητικό όταν μεγάλα ποσά διαδραστικότητας πρέπει να παραμένουν στην μνήμη.

Μία άλλη τεχνική είναι η δυναμική εγγραφή Ενιαίων Δεικτών Πόρων έτσι, ώστε επιπλέον πληροφορίες, οι οποίες προσδιορίζουν μία συνεδρία με τον διακομιστή, να περιλαμβάνονται στο URL πριν αυτό να σταλεί στον διακομιστή. Οι λεπτομέρειες αυτού είναι πέρα από τον σκοπό αυτού του βιβλίου. Ωστόσο, αξίζει να πούμε ότι αποτελούν ένα άλλο παράδειγμα hack.

2.1.1.8 Μία λύση servlet

Το servlet API υποστηρίζει μία κομψή και βολική λύση για το πρόβλημα του εντοπισμού διαρκών δεδομένων. Παρέχει μία κλάση HttpSession, που είναι μία φόρμα συσχετιστικής μνήμης, η οποία εντοπίζει τα δεδομένα που σχετίζονται με μία συνεδρία, όπου συνεδρία είναι η σειρά των διαδράσεων ενός browser μ' έναν ιστοχώρο πριν την απομάκρυνση. Ένα servlet μπορεί να δημιουργήσει ένα αντικείμενο το οποίο θα περιγράφεται απ' αυτήν την κλάση δια μέσου της μεθόδου getSession που βρίσκεται στην κλάση HttpServletRequest.

θα επιστρέψει μ' ένα αντικείμενο συνεδρίας για την συνεδρία που εκτελείται αυτήν την στιγμή μεταξύ ενός συγκεκριμένου πελάτη, που τρέχει έναν browser, και του διακομιστή, μέσα στον οποίο βρίσκεται το servlet. Το όρισμα Boolean καθορίζει τί θα συμβεί αν ο χρήστης δεν έχει τρέχουσα συνεδρία. Αν το όρισμα είναι αληθινό τότε δημιουργείται ένα αντικείμενο συνεδρίας. Αν το όρισμα είναι μη αληθινό τότε επιστρέφει ένα κενό.

Ένα αντικείμενο HttpSession δρα ως μία συσχετιστική μνήμη, όπως κι ένα αντικείμενο Hashtable: σχετίζει κλειδιά με τιμές, αν και το κλειδί που χρησιμοποιείται πρέπει να είναι ένα αντικείμενο στοιχειοσειράς. Για την εγγραφή και ανάκτηση δεδομένων χρησιμοποιούνται δύο μέθοδοι. Η πρώτη είναι η μέθοδος putValue της κλάσης HttpSession. Αυτή έχει δύο ορίσματα: μία στοιχειοσειρά που αντιπροσωπεύει ένα κλειδί κι το σχετικό αντικείμενο. έτσι, για παράδειγμα, ο κώδικας

```
HttpSessionhtSess=rq.getSession(true);
..
htSess.putValue("Orders",orderVector);
```

θα συσχετίσει το Vector αντικείμενο orderVector, που περιέχει παραγγελίες αγορών, με την στοιχειοσειρά "Orders". Η δεύτερη μέθοδος getValue επιστρέφει το αντικείμενο που σχετίζεται μ' ένα συγκεκριμένο κλειδί, για παράδειγμα ο κώδικας που παρουσιάζεται παρακάτω θα επιστρέψει το Vector αντικείμενο που σχετίζεται με την στοιχειοσειρά "Credit Cards".

```
HttpSessionsess=rq.getSession(true);
..
sess.putValue("CreditCards",orderVector);
..
Vector shopVect = (Vector) sess.getValue("Credit Cards");
```

Οι δύο μέθοδοι τείνουν να χρησιμοποιούνται αρκετά συχνά όταν ο διαρκής προγραμματισμός χρησιμοποιεί servlet: η μέθοδος getValueNames επιστρέφει με μία στοιχειοσειρά που περιλαμβάνει όλα τα ονόματα των αντικειμένων που σχετίζονται με μία HttpSession, και η removeValue αφαιρεί ένα αντικείμενο από μία συνεδρία, με το όρισμα στοιχεισειράς της μεθόδου να προσδιορίζει το αντικείμενο, για παράδειγμα η δήλωση

```
htSess.removeValue("Credit Cards");
```

θα αφαιρέσει από την συνεδρία το αντικείμενο που σχετίζεται με τις πιστωτικές κάρτες.

Δεδομένων αυτών των εργαλείων μία εφαρμογή που απαιτεί διάρκεια μπορεί να προγραμματιστεί εύκολα, για παράδειγμα ένα καλάθι αγορών μπορεί να συσχετιστεί μ' ένα servlet που κρατά αγορασμένα προϊόντα, που προσθέτει προϊόντα και υπολογίζει την συνολική τιμή της αγοράς όταν ο χρήστης επιθυμεί ν' αποσυνδεθεί.

Αυτή ήταν μία πολλή γρήγορη ματιά στις συνεδρίες. Ωστόσο, σας έδωσε τα περισσότερα εργαλεία που θα σας επιτρέψουν να αναπτύξετε εξεζητημένες εφαρμογές Web που απαιτούν διάρκεια. Οι παραπομπές στο τέλος του κεφαλαίου παρέχουν περισσότερες λεπτομέρειες σε θέματα servlet, όπως η χρήση των servlet σε από κοινού συνεργασία, η ασφάλεια των servlet και η σύνδεση.

Apache Tomcat (server που τρέχει τα servlets)

2.2 APACHE TOMCAT

2.2.1.1 Για να είναι διαθέσιμες οι ιστοσελίδες στο διαδίκτυο πρέπει να «φορτωθούν» σε έναν διακομιστή (server). Στην παρούσα εργασία χρησιμοποιήθηκε ο Apache Tomcat. Ο Apache Tomcat (ή Jakarta Tomcat ή απλά Tomcat) είναι ένα λογισμικό ανοιχτού κώδικα (ελεύθερο λογισμικό) που αναπτύχθηκε από την Apache Software Foundation(ASF). Είναι ένας server ο οποίος μας επιτρέπει να εκτελέσουμε αρχεία τύπου Java και JSP, σε αντίθεση με τον Apache ο οποίος μας επιτρέπει να εκτελούμε αρχεία PHP. Όταν δημιουργούμε μια μεγάλη web εφαρμογή η οποία θα δέχεται requests ταυτόχρονα από διάφορους clients και όταν θα πρέπει να απαντάει σε όλους με ένα μεγάλο όγκο δεδομένων, κρίνεται αναγκαία η χρήση πολλών servers σε clusters που θα εξυπηρετούν αυτά τα requests αλλά και κάποια λειτουργία που θα μοιράζει το βάρος (μνήμη, requests, responses, processes) σε όλα τα clusters. Λόγω του ότι η διαδικασία αυτή προϋποθέτει την αγορά νέου hardware, η οποία δεν συμφέρει από οικονομικής άποψης, επιλέγεται η λύση του software.

Ο Apache Tomcat περιλαμβάνει εργαλεία για τη διαμόρφωση και τη διαχείριση, αλλά μπορεί επίσης να ρυθμιστεί για την επεξεργασία XML αρχείων. Με το εργαλείο παρέχεται η δυνατότητα:

- Ρύθμισης του διαχειριστή ασφάλειας της Java
- Χρησιμοποίησης πηγών δεδομένων μέσω JDBC
- Υποστήριξης του πρωτοκόλλου SSL για κρυπτογραφημένες συνδέσεις
- Χρήσης διαμεσολαβητών (proxy)
- Εικονικής λειτουργίας (virtual hosting)
- Εκτέλεσης σεναρίων CGI

[1] **Tomcat: The Definitive Guide, Jason Brittain, Ian F.Darwin**

2.3 Android

2.3.1.1 Στον κόσμο της τεχνολογίας, όπου τα αμύθητα ποσά και οι τεχνολογικές καινοτομίες διαδέχονται η μία την άλλη με ταχύτατους ρυθμούς, οι κολοσσοί διαρκώς μάχονται μεταξύ τους: Apple vs Microsoft, Google vs Apple vs Samsung, και πάει λέγοντας. Άλλος κερδίζει, άλλος χάνει. Πώς θα μπορούσαμε να ερμηνεύσουμε αλλιώς το γεγονός ότι ο εμπνευστής και δημιουργός του Android, του λειτουργικού συστήματος για κινητά τηλέφωνα που βασίζεται στον πυρήνα του Linux, που έχει τη σφραγίδα της Google, έκανε τα πρώτα του επαγγελματικά του βήματα στην Apple;

2.3.1.2 Η ιστορία

Τα πράγματα ξεκίνησαν όταν ο ευφυής Andy Rubin θέλησε την Άνοιξη του 2005 να χρησιμοποιήσει την Google ως κατ' εξοχήν μηχανή α

ναζήτησης για το T-Mobile Sidekick, μια φερέλπιδα συσκευή κινητού, την οποία είχε αναπτύξει με την ομάδα συνεργατών του. Εν συνεχεία, ζήτησε να συναντηθεί με τον Larry Page, ο οποίος είναι ο ένας από τους δύο ιδρυτές της Google. Σε αυτήν τη συνάντηση ο Rubin παρουσίασε το Android ως ένα εν δυνάμει παγκόσμιο ανοικτό λειτουργικό σύστημα που θα άλλαζε για πάντα τον τρόπο που διαντιδρούνε οι χρήστες με το κινητό σας, τονίζοντας, ταυτόχρονα, τη σταθερή υπεροχή που παρατηρείται στις συνήθειες του αγοραστικού κοινού των κινητών τηλεφώνων, σε αντιδιαστολή με τις πωλήσεις ηλεκτρονικών υπολογιστών.

Την ίδια στιγμή, ο Page δεν ήθελε να γίνει απλώς ο υποστηρικτής του Android, ήθελε να γίνει ο ιδιοκτήτης του. Ο Andy… βρήκε «την καλή», την ώρα που ένας ισχυρός παίκτης εμφανίστηκε στο προσκήνιο και έθεσε έτσι τους όρους του ανταγωνισμού σε άλλο επίπεδο. Ο καινούργιος «παίκτης» δεν είναι άλλος από εκείνον που τελικά λάνσαρε το καλοκαίρι του 2005, το iPhone της Apple.

Ο επιχειρηματικός-τεχνολογικός κόσμος περίμενε πως η Google θα απαντούσε με ένα gPhone, αλλά αυτό δεν έγινε, διότι έγινε κάτι άλλο, πολύ σημαντικότερο. Το Φθινόπωρο του 2005 ανακοινώνεται ότι 34 εταιρίες, όπως η Texas Instruments, η Intel, η T-Mobile και η Sprint Nextel, ενώνουν τις δυνάμεις τους με την Google για τη δημιουργία μιας πλατφόρμας ανοιχτού κώδικα που θα έχει ενσωματωμένο το λογισμικό Linux και θα εκπροσωπείται από μια νέα συστάδα εταιριών που θα καλείται Open Handset Alliance.

Στο χορό δεν άργησαν να μπουν και άλλες εταιρείες, όπως η HTC, η Motorola και η LG, ανακοινώνοντας την πρόθεσή τους να δώσουν προς πώληση στην αγορά smartphones με λειτουργικό σύστημα Android σε διάφορα σχήματα και μεγέθη, με τα οποία θα μπορεί να έχει ο χρήστης να ενσωματώνει στο κινητό του πλήθος εφαρμογών.

Το Android είναι ένα λειτουργικό σύστημα που ενσωματώνεται σε συσκευές κινητής τηλεφωνίας, τα οποία διαθέτουν οθόνη αφής, τρέχουν τον πυρήνα (kernel) του λειτουργικού Linux και ακόμη, επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με τη χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας τη συσκευή μέσω βιβλιοθηκών λογισμικού της Google.

Συσκευές με Android υπάρχουν πλέον πάρα πολλές, η καθεμία με διαφορετικά χαρακτηριστικά και από διάφορες κατασκευάστριες εταιρίες: η LG, Samsung, HTC, Sony Ericsson, Motorola, είναι μερικές από τις εταιρίες που χρησιμοποιούν το λειτουργικό Android για τα smartphones τους.

Το πολύ θετικό με τις συσκευές Android είναι ότι είναι αφενός multimedia (σας δίνουν τη δυνατότητα να αναπαράγετε πολλαπλά μέσα) και multitasking (δίνουν τη δυνατότητα εκτέλεσης πολλών εφαρμογών ταυτόχρονα, π.χ. ακούτε τραγούδια ενώ σερφάρετε στο ίντερνετ και ταυτόχρονα απαντάτε σε ένα SMS χωρίς να κλείσετε καμία εφαρμογή ή να χάσετε τη σελίδα που επισκεφτήκατε).

Το web browsing στο Android είναι ταχύτατο, υποστηρίζεται από flash και υπάρχουν πολλοί browsers για να καλύψουν και τους πλέον απαιτητικούς.

Ανεξάρτητα από το κόστος, όλες οι συσκευές Android διαθέτουν GPS και Wi-fi, δικαιώνοντας έτσι το βασικό λόγο δημιουργίας του εν λόγω λειτουργικού συστήματος που δεν είναι άλλος παρά η ανεμπόδιστη και εύκολη πρόσβαση στο διαδίκτυο, σε συνδυασμό με ένα πλήθος εφαρμογών (apps), όπως χάρτες, αναζήτηση, chat και e-mail, που πραγματικά επιτρέπουν στο χρήστη να μένει διαρκώς δικτυωμένος και ενημερωμένος.

Βασικό χαρακτηριστικό του Android, επίσης, είναι η πληθώρα εφαρμογών που διατηρούν τη συνεχή σύνδεση με Facebook, MySpace, Twitter και δεκάδες άλλες υπηρεσίες social networking. Ακόμη, το Android σας δίνει τη δυνατότητα να προσθέσετε widgets, δηλαδή εικονίδια για την ταχύτερη πρόσβαση στα προγράμματα, τα οποία τοποθετούνται στη home screen του κινητού (launcher).

Επιπλέον η notification bar είναι εξαιρετικά χρήσιμη, καθώς με ένα απλό drag βλέπετε όλες τις ειδοποιήσεις για τη συσκευή σας, αλλά και τα προγράμματα (applications) που έχετε εγκαταστήσει.

Όσον αφορά το hardware, οι διπύρηνοι επεξεργαστές και οι διακεκριμένες GPU είναι πλέον γεγονός, ενώ αναμένουμε και επεξεργαστές τεσσάρων πυρήνων, καθώς αυτό έχει ήδη ανακοινωθεί από την Nvidia με τον επεξεργαστή Kal – El ο οποίος μάλιστα θα περιέχει και έναν πέμπτο –stealth- πυρήνα.

2.3.1.3 Android Market

Το Android Market, είναι ένα online κατάστημα της Google, προσφέρει σε κάθε χρήστη εφαρμογές για το κινητό τους που είναι συμβατό με το λειτουργικό της Google. Το συντριπτικό ποσοστό των εφαρμογών είναι δωρεάν ενώ πλέον οι πληρωμένες εφαρμογές είναι διαθέσιμες και στο ελληνικό κοινό.

2.3.1.4 Εκδόσεις Android

Το Android που κυκλοφορεί διάφορες εκδόσεις με ονομασίες που σου ανοίγουν την όρεξη για… νέα χαρακτηριστικά, όπως τα παλαιότερα CupCake (1.5), Donut (1.6), Éclair (2.0, 2.1), GingerBread (2.3) αλλά και FroYo (2.2) Honeycomb (3.0) που υλοποιείται σε ταμπλέτες ενώ υπάρχουν πλέον και οι εκδόσεις Honeycomb (3.1) και Honeycomb (3.2). Δεν πρέπει να ξεχάσουμε να αναφέρουμε και την επερχόμενη έκδοση Ice Cream Sandwich η οποία θα είναι διαθέσιμη από τον Οκτώβρη.

Από την «παρθενική» έκδοση Android 1.0, η οποία κυκλοφόρησε το Σεπτέμβριο του 2008, μέχρι την αμέσως επόμενη, 1.1 που παρουσιάστηκε το Φεβρουάριο του 2009,

χρειάστηκε ένας χρόνος για να γίνει η έκρηξη των καινοτόμων εκδόσεων και των σημαντικών αλλαγών που επέφεραν για τον χρήστη.

Το πρώτο smartphone που «έτρεξε» Android είναι το T-Mobile G1 κατασκευασμένο από την HTC με οθόνη αφής TFT-LCD 3,2", full qwerty πληκτρολόγιο, πρόσβαση σε Gmail, YouTube, Google maps, Google talk, Google calendar, κάμερα 3,2MP με αυτόματη εστίαση και κάρτα μνήμης micro SD.

Μερικές από τις χαρακτηριστικές αλλαγές των εκδόσεων που ακολούθησαν και οι εκπρόσωποί τους:

2.3.1.5 *Η έκδοση 1.5 (CupCake) σας δίνει τη δυνατότητα για ανέβασμα βίντεο στο YouTube και εικόνων στο Picasa απευθείας από την κινητή σας συσκευή, για αυτόματη σύνδεση ακουστικών headset σε συγκεκριμένη απόσταση, ενώ περιλαμβάνει εικονικό πληκτρολόγιο με πρόβλεψη λέξεων και νέα widgets για την αρχική οθόνη. Επίσης, το CupCake (1.5) «τρέχει» το HTC Hero με οθόνη αφής TFT-LCD, 3,2", ανάλυσης 320x480p (HVGA). Διαθέτει κάμερα 5MP με αυτόματη εστίαση και κάρτα μνήμης micro SD.*

2.3.1.6 *Με το Donut (1.6) δίνεται έμφαση στη φωνητική αναζήτηση αλλά και στην αναζήτηση από την αρχική οθόνη με bookmarks, ιστορικό, επαφές κ.ά., ενώ υποστηρίζονται και οθόνες αναλύσεων WVGA. Αυτό το έχει το LG GT 540, το οποίο επιπλέον διαθέτει qwerty πληκτρολόγιο για γρήγορη αποστολή SMS, προσφέρει ευκολία στην εύρεση των επαφών, ενώ διαθέτει 3G και Wi-Fi για να είστε δια-δικτυωμένοι όπου και να βρίσκεστε. Επίσης, το Sony Ericsson X10 με επεξεργαστή Snapdragon 1GHz της Qualcomm και οθόνη αφής 4", 854x480pixels. Στην πίσω όψη βρίσκεται η κάμερα 8,1MP και το LED Flash κάτω ακριβώς από τον φακό της.*

2.3.1.7 *Οι επόμενες επίσης γλυκές εκδόσεις, είναι οι Éclair (2.0, 2.1) που προχωρούν ακόμη πιο μπροστά, διαθέτοντας νέο browser UI, υποστήριξη HTML5, Google Maps 3.1.2, ενσωματωμένη υποστήριξη για flash στην κάμερα, ψηφιακό ζουμ, δυνατότητα αντίληψης multi-touch, βελτιωμένο εικονικό πληκτρολόγιο, bluetooth 2.1 και live wallpapers. Το Motorola Droid- είναι ένας ακόμη εκπρόσωπος της έκδοσης 2.0 (Éclair), με επεξεργαστή Arm Cortex A8 550MHz. Ο σχεδιασμός του είναι slide με πλήρες qwerty πληκτρολόγιο και οθόνη αφής 3,7", ανάλυσης 480×854p. Tosamsung galaxy S, με οθόνη αφής 4.0 Super AMOLED «τρέχει» Android 2.1 (Éclair). Με επεξεργαστή 1GHz, εσωτερική μνήμη 8GB, Wi-Fi και bluetooth 3.0, δίνει την ταχύτητα, το χώρο και τις απαιτούμενες δυνατότητες σύνδεσης. Επίσης, το HTC Wildfire με οθόνη αφής 3,2 QVGA, το οποίο υποστηρίζει Adobe flash για άνετη πλοήγηση στο ίντερνετ και online βίντεο και παιχνίδια. Διαθέτει κάμερα στα 5MP με αυτόματη εστίαση και LED φλας.*

2.3.1.8 *Η έκδοση 2.2 (FroYo-Frozen Yogurt) αναβάθμισε αισθητά την ταχύτητα του OS, αλλά και τη γενικότερη απόδοση, διαθέτει Market με δυνατότητα αυτόματων updates, επιλογή εγκατάστασης εφαρμογών στην κάρτα μνήμης, υποστήριξη Adobe flash 10.1 ενώ ενσωματώνει τον Chrome V8 JavaScript στα browsers applications. Στη FroYo βρίσκουμε και τη δυνατότητα χρήσης της συσκευής για διαμοιρασμό ίντερνετ μέσω Wi-Fi σε άλλες συσκευές (tethering). Το LG Optimus 2x διαθέτει οθόνη αφής 4,*

λειτουργικό Android Froyo και διπύρηνο επεξεργαστή Nvidia Tegra 2. Με ισχυρή κάμερα 8MP, δυνατότητα εγγραφής και αναπαραγωγής βίντεο full HD και συνδεσιμότητα HDMI και DLNA. Το HTC Desire Z με Android Froyo και οθόνη αφής 3,7, ανάλυσης WVGA (480x800p), συρόμενο Qwerty πληκτρολόγιο για γρήγορα e-mail, 5MP κάμερα και δέκτη GPS. Χάρη στον ενσωματωμένο Adobe Flash Player, προβάλλει κάθε ιστοσελίδα.

2.3.1.9 Η GingerBread (2.3), που κυκλοφόρησε το Δεκέμβριο του 2010, υποστηρίζει πλέον πολύ μεγάλα μεγέθη οθονών και αναλύσεων, διαθέτει προεγκατεστημένη υποστήριξη για τηλεφωνικές κλήσεις μέσω ίντερνετ (VoIP), επανασχεδιασμένο multi-touch πληκτρολόγιο, λειτουργίες copy-paste σε όλο το λειτουργικό, download manager για κατέβασμα μεγάλων αρχείων, καθώς και προεγκατεστημένη υποστήριξη για πολλαπλές κάμερες. Το Sony Xperia Play, συσκευή προσανατολισμένη στο gaming, «τρέχει» GingerBread (2.3) με επεξεργαστή Scorpion ARM7 και ταχύτητα στα 1GHz. Η οθόνη του είναι 4 ιντσών με ανάλυση 480x854p και η εσωτερική του μνήμη 400MB. Ακόμη τα HTC Cha Cha και HTC Salsa με GingerBread (2.3) ενσωματώνουν ένα εξειδικευμένο Facebook πλήκτρο, για πρόσβαση με ένα άγγιγμα, στην υπηρεσία του Facebook μέσα από την εμπειρία HTC Sense.

2.3.1.10 Και φτάνουμε στην πλέον πρόσφατη έκδοση, τη Honeycomb (3.0), μόνο για tablets. Διαθέτει ανανεωμένο γραφικό περιβάλλον με αρκετά 3D στοιχεία, καλύτερο web browser με tabs και ανανεωμένο εικονικό πληκτρολόγιο, βελτιστοποιημένη εφαρμογή για ανάγνωση Google e-books, ειδική έκδοση του Gmail για tablets, δυνατότητα βιντεοκλήσεων μέσω εφαρμογής Google Talk και ανανεωμένη έκδοση Google Maps.

Η έκδοση Honeycomb (3.1) προσέθεσε την επιλογή να μεταφέρεται περιεχόμενο απευθείας από συσκευές USB, ενώ τέλος η έκδοση 3.2 προσέθεσε διάφορες δυνατότητες και ευκολίες για χρήστες και developers όπως τη μεταφορά αρχείων από κάρτες SD και δυνατότητα Zoom to Fill. Αυτή τη στιγμή στην αγορά υπάρχουν αρκετές ταμπλέτες που τρέχουν Android Honeycomb όπως το Motorola Xoom, το Samsung Galaxy Tab 10.1, το επερχόμενο tablet της Sony, Sony Tablet S, το Asus Eee Pad Trasformer και το Toshiba AT200 το οποίο μάλιστα είναι εξοπλισμένο με την τελευταία έκδοση του Android Honeycomb 3.2.

2.4 Σχεσιακές βάσεις δεδομένων

2.4.1 Εισαγωγή στις Βάσεις Δεδομένων

Η αλματώδης ανάπτυξη της επιστήμης της πληροφορικής και των επικοινωνιών τα τελευταία χρόνια έχει καταστήσει την πληροφορία ως ένα από τα βασικότερα και πολυτιμότερα αγαθά. Είναι κοινός τόπος σήμερα η εκτίμηση ότι το αγαθό της πληροφορίας είναι επιθυμητό απ' όλους τους εργαζόμενους αλλά και τους

εκπαιδευόμενους, ώστε να είναι πιο αποδοτικοί, ανταγωνιστικοί αλλά και παραγωγικοί στην εργασία τους.

Τα συστήματα βάσεων δεδομένων τα χρησιμοποιούμε για να μπορούμε να αποθηκεύσουμε, να επεξεργαστούμε αλλά και να εκμεταλλευτούμε αποδοτικά αυτόν τον τεράστιο όγκο των πληροφοριών που αυξάνονται με αλματώδεις ρυθμούς καθημερινά.

2.4.1.1 Τα Δεδομένα και οι Πληροφορίες

Με τον όρο *πληροφορία* αναφερόμαστε συνήθως σε ειδήσεις, γεγονότα και έννοιες που αποκτάμε από την καθημερινή μας επικοινωνία και τα θεωρούμε ως αποκτηθείσα γνώση, ενώ τα δεδομένα μπορούν να είναι μη κατάλληλα επεξεργασμένα και μη ταξινομημένα σύνολα πληροφοριών. Ένας αυστηρός ορισμός για το τι είναι δεδομένα και τι είναι πληροφορία, σύμφωνα με την επιτροπή ANSI των ΗΠΑ, είναι ο εξής :

- *Δεδομένα (data)* είναι μια παράσταση, όπως γράμματα, αριθμοί, σύμβολα κ.ά. στα οποία μπορούμε να δώσουμε κάποια σημασία (έννοια).
- *Πληροφορία (information)* είναι η σημασία που δίνουμε σ' ένα σύνολο από δεδομένα, τα οποία μπορούμε να επεξεργαστούμε βάσει προκαθορισμένων κανόνων και να βγάλουμε έτσι κάποια χρήσιμα συμπεράσματα. Με τις πληροφορίες περιορίζεται η αβεβαιότητα που έχουμε για διάφορα πράγματα και βοηθιόμαστε έτσι στο να λάβουμε σωστές αποφάσεις.

Τα δεδομένα μπορούν να θεωρηθούν ως τρόποι αναπαράστασης εννοιών και γεγονότων που μπορούν να υποστούν διαχείριση και επεξεργασία. Η συλλογή και αποθήκευση ενός τεράστιου όγκου δεδομένων όπως απαιτούν οι κοινωνικές συνθήκες σήμερα, δεν λύνει τελείως το πρόβλημα της σωστής οργάνωσης και ταξινόμησης των δεδομένων. Τα δεδομένα θα πρέπει να οργανωθούν με τέτοιο τρόπο έτσι ώστε να μπορούμε να τα εντοπίζουμε και να τα αξιοποιούμε εύκολα και γρήγορα και τη στιγμή που τα χρειαζόμαστε.

Ένα κλασικό παράδειγμα μη σωστής οργάνωσης δεδομένων θα ήταν για παράδειγμα ο τηλεφωνικός κατάλογος της πόλης της Θεσσαλονίκης, όπου οι συνδρομητές δεν θα ήταν καταχωρημένοι αλφαβητικά σύμφωνα με το επώνυμο και το όνομά τους, αλλά εντελώς τυχαία. Ένας τέτοιος τηλεφωνικός κατάλογος θα περιείχε μια τεράστια ποσότητα δεδομένων αλλά θα ήταν ουσιαστικά άχρηστος.

Ένα άλλο παράδειγμα μη σωστής οργάνωσης δεδομένων θα ήταν μια πολύ μεγάλη βιβλιοθήκη με χιλιάδες τόμους βιβλίων και χωρίς να διαθέτει κάποιο υποτυπώδες σύστημα οργάνωσης και ταξινόμησης των βιβλίων. Ούτε οι υπάλληλοι της βιβλιοθήκης θα μπορούσαν να κάνουν τη δουλειά τους αλλά ούτε και οι επισκέπτες θα μπορούσαν να αξιοποιήσουν την πληθώρα των πληροφοριών που περιέχονται στα βιβλία. Εκτός λοιπόν από τη μόνιμη αποθήκευση των δεδομένων, χρειαζόμαστε και κάποιους τρόπους ευέλικτης και αποδοτικής οργάνωσής τους.

Χαρακτηριστικά παραδείγματα δεδομένων που απαιτούν σωστή και αποδοτική οργάνωση είναι τα εξής :

- Τα στοιχεία υπαλλήλων, πελατών, προμηθευτών και παραγγελιών μιας εμπορικής επιχείρησης.
- Τα στοιχεία υλικών μιας αποθήκης.

- Τα στοιχεία ταινιών, πελατών και δανεισμών μιας βιντεολέσχης.
- Τα στοιχεία υπαλλήλων, γιατρών, ασθενών αλλά και υλικών ενός νοσοκομείου.
- Τα στοιχεία βιβλίων, χρηστών (δανειστών) και δανεισμών μιας βιβλιοθήκης.

2.4.1.2 Η Οργάνωση Αρχείων

Ο πιο γνωστός τρόπος οργάνωσης δεδομένων με τη χρήση ηλεκτρονικών υπολογιστών είναι σε αρχεία εγγραφών. Για να κατανοήσουμε καλύτερα ορισμένες έννοιες, θα εξετάσουμε την περίπτωση ενός αρχείου πελατών και παραγγελιών μιας εμπορικής επιχείρησης. Για να οργανώσουμε σωστά το αρχείο μας, θα πρέπει να δημιουργήσουμε καρτέλες για τους πελάτες, αλλά και για τις παραγγελίες τους αργότερα, που θα πρέπει να περιέχουν τα εξής στοιχεία ανά πελάτη :

- Κωδικός
- Επώνυμο
- Όνομα
- Διεύθυνση
- ΤΚ
- Πόλη
- Τηλέφωνο
- ΑΦΜ
- ΔΟΥ

Η αντιστοίχηση του παλιού τρόπου οργάνωσης με τις καρτέλες σε σχέση με τον σύγχρονο ηλεκτρονικό τρόπο οργάνωσης, έχει ως εξής :

- Συρτάρι – Αρχείο Δεδομένων
- Καρτέλα πελάτη – Εγγραφή του αρχείου δεδομένων
- Στοιχείο της καρτέλας – Πεδίο της εγγραφής

Ένα *αρχείο (file)* θα μπορούμε να το χαρακτηρίσουμε σαν ένα σύνολο που αποτελείται από οργανωμένα ομοειδή στοιχεία. Τα στοιχεία ενός αρχείου μπορούμε να τα οργανώσουμε σε λογικές ενότητες και το σύνολο των στοιχείων που περιέχει μια λογική ενότητα καλείται *εγγραφή (record)*. Το κάθε στοιχείο της εγγραφής καλείται *πεδίο (field)*. Το πεδίο αποτελεί και τη μικρότερη δυνατή υποδιαίρεση των στοιχείων ενός αρχείου. Ένα πεδίο χαρακτηρίζεται από τον μέγιστο αριθμό των χαρακτήρων (bytes) που απαιτούνται για την καταχώρησή του στη μνήμη του υπολογιστή και που αποκαλείται *μήκος του πεδίου (field length)*.

Σε μια οργάνωση αρχείου όπως είναι οι πελάτες μιας εμπορικής επιχείρησης που είδαμε νωρίτερα, τα αντίστοιχα πεδία όλων των εγγραφών καταλαμβάνουν τον ίδιο αριθμό σε bytes που είναι αυτός που έχουμε ορίσει κατά τη δημιουργία του αρχείου. Για παράδειγμα, αν ορίσαμε ότι το πεδίο Επώνυμο θα έχει μήκος 15 χαρακτήρες, τότε το πεδίο της εγγραφής του πελάτη με επώνυμο Παπαδόπουλος, αλλά και το πεδίο της εγγραφής του πελάτη με επώνυμο Βες θα καταλαμβάνουν από 15 bytes στη μνήμη του υπολογιστή, ενώ αν ένας πελάτης ονομάζεται Παπαχριστοδουλόπουλος, τότε θα γίνει αποκοπή του επωνύμου του και θα καταχωρηθούν στη μνήμη του υπολογιστή μόνο τα 15 πρώτα γράμματα, δηλ. τα Παπαχριστοδουλό.

Ένα πεδίο χαρακτηρίζεται ακόμη και από το είδος των δεδομένων που μπορεί να περιέχει, όπως :

- *Αλφαριθμητικό (alphanumeric)*, μπορεί να περιέχει γράμματα, ψηφία ή και ειδικούς χαρακτήρες.
- *Αριθμητικό (numeric)*, μπορεί να περιέχει μόνο αριθμούς.
- *Αλφαβητικό (alphabetic)*, μπορεί να περιέχει μόνο γράμματα (αλφαβητικούς χαρακτήρες).
- *Ημερομηνίας (date)*, μπορεί να περιέχει μόνο ημερομηνίες.
- *Δυαδικό (binary)*, μπορεί να περιέχει ειδικού τύπου δεδομένα, όπως εικόνες, ήχους κ.ά.
- *Λογικό (logical)*, μπορεί να περιέχει μόνο μία από δύο τιμές, οι οποίες αντιστοιχούν σε δύο διακριτές καταστάσεις και μπορούν να χαρακτηρισθούν σαν 0 και 1 ή σαν αληθές (true) και ψευδές (false).
- *Σημειώσεων (memo)*, μπορεί να περιέχει κείμενο με μεταβλητό μήκος, το οποίο μπορεί να είναι και αρκετά μεγάλο και συνήθως αποθηκεύεται σαν ξεχωριστό αρχείο από το κύριο αρχείο.

Όσον αφορά τις εγγραφές, χρήσιμοι ορισμοί είναι οι εξής :

- *Μήκος εγγραφής (record length)* καλείται το άθροισμα που προκύπτει από τα μήκη των πεδίων που την αποτελούν.
- *Δομή εγγραφής (record layout)* ή *γραμμογράφηση* καλείται ο τρόπος με τον οποίο οργανώνουμε τα πεδία μιας εγγραφής.
- *Διάβασμα (read)* από αρχείο σημαίνει τη μεταφορά των δεδομένων του αρχείου, που γίνεται συνήθως ανά μία εγγραφή, από το μέσο αποθήκευσης (σκληρό δίσκο ή δισκέτα) στην κεντρική μνήμη του υπολογιστή για επεξεργασία.
- *Γράψιμο (write)* σε αρχείο σημαίνει μεταφορά των δεδομένων του αρχείου, που γίνεται συνήθως ανά μία εγγραφή, από την κεντρική μνήμη του υπολογιστή στο μέσο αποθήκευσης (σκληρό δίσκο ή δισκέτα).

2.4.1.3 Προβλήματα της Οργάνωσης Αρχείων

Στα αρχικά στάδια της οργάνωσης αρχείων, ήταν πολύ συνηθισμένη πρακτική η δημιουργία ξεχωριστών εφαρμογών (προγραμμάτων) και ξεχωριστών αρχείων, όπως για παράδειγμα η δημιουργία ενός αρχείου πελατών και ενός άλλου ανεξάρτητου αρχείου για τις παραγγελίες των πελατών. Τα προβλήματα που προέκυψαν από την πρακτική αυτή είναι τα εξής :

- *Πλεονασμός των δεδομένων (data redundancy)*. Υπάρχει η περίπτωση να έχουμε επανάληψη των ίδιων δεδομένων σε αρχεία διαφορετικών εφαρμογών. Για παράδειγμα, αν έχουμε ένα αρχείο πελατών και ένα αρχείο παραγγελιών αυτών των πελατών, είναι σχεδόν σίγουρο ότι θα υπάρχουν κάποια στοιχεία των πελατών που θα υπάρχουν και στα δύο αρχεία.
- *Ασυνέπεια των δεδομένων (data inconsistency)*. Αυτό μπορεί να συμβεί όταν υπάρχουν τα ίδια στοιχεία των πελατών (πλεονασμός) και στο αρχείο πελατών και στο αρχείο παραγγελιών και χρειασθεί να γίνει κάποια αλλαγή στη διεύθυνση

ή στα τηλέφωνα κάποιου πελάτη, οπότε είναι πολύ πιθανό να γίνει η διόρθωση μόνο στο ένα αρχείο και όχι και στο άλλο.

- *Αδυναμία μερισμού δεδομένων (data sharing)*. Μερισμός δεδομένων σημαίνει δυνατότητα για κοινή χρήση των στοιχείων κάποιων αρχείων. Για παράδειγμα, ο μερισμός δεδομένων θα ήταν χρήσιμος αν με την παραγγελία ενός πελάτη μπορούμε να έχουμε πρόσβαση την ίδια στιγμή στο αρχείο πελατών για να δούμε το υπόλοιπο του πελάτη και μετά στο αρχείο της αποθήκης για να δούμε αν είναι διαθέσιμα τα προϊόντα που παρήγγειλε ο συγκεκριμένος πελάτης. Η αδυναμία μερισμού δεδομένων δημιουργεί καθυστέρηση στη λήψη αποφάσεων και στην εξυπηρέτηση των χρηστών.
- *Αδυναμία προτυποποίησης*. Έχει να κάνει με την ανομοιομορφία και με την διαφορετική αναπαράσταση και οργάνωση των δεδομένων στα αρχεία των εφαρμογών. Η αδυναμία αυτή δημιουργεί προβλήματα προσαρμογής των χρηστών καθώς και προβλήματα στην ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων.

2.4.1.4 Οι Βάσεις Δεδομένων και τα ΣΔΒΔ (DBMS)

Για να δοθεί μια λύση σ' όλα τα παραπάνω προβλήματα, και με βάση το γεγονός ότι η χρήση των ηλεκτρονικών υπολογιστών και συνεπώς η ηλεκτρονική καταχώρηση και επεξεργασία δεδομένων αυξήθηκε κατακόρυφα ήδη από τη δεκαετία του '70 στις μεγάλες επιχειρήσεις και άρα είχαμε πάρα πολλές εφαρμογές να επεξεργάζονται δεδομένα σε πάρα πολλά αρχεία ταυτόχρονα, προτάθηκε η συνένωση όλων των αρχείων μιας εφαρμογής. Εκτός, όμως, από τη συνένωση των αρχείων, απαιτείτο και μια σωστή οργάνωσή τους. Δημιουργήθηκαν έτσι οι Τράπεζες Πληροφοριών ή Βάσεις Δεδομένων (Data Bases).

Μια *Βάση Δεδομένων (ΒΔ)* είναι ένα σύνολο αρχείων με υψηλό βαθμό οργάνωσης τα οποία είναι συνδεδεμένα μεταξύ τους με λογικές σχέσεις, έτσι ώστε να μπορούν να χρησιμοποιούνται από πολλές εφαρμογές και από πολλούς χρήστες ταυτόχρονα. Υπάρχει ένα ειδικό λογισμικό το οποίο μεσολαβεί ανάμεσα στις αρχεία δεδομένων και τις εφαρμογές που χρησιμοποιούν οι χρήστες και αποκαλείται *Σύστημα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ)* ή *DBMS (Data Base Management System)*. Το ΣΔΒΔ είναι στην ουσία ένα σύνολο από προγράμματα και υπορουτίνες που έχουν να κάνουν με τον χειρισμό της βάσης δεδομένων, όσον αφορά τη δημιουργία, τροποποίηση, διαγραφή στοιχείων, με ελέγχους ασφαλείας κ.ά.

Οι χρήστες των εφαρμογών αντλούν τα στοιχεία που τους ενδιαφέρουν από τη βάση δεδομένων χωρίς να είναι σε θέση να γνωρίζουν με ποιο τρόπο είναι οργανωμένα τα δεδομένα σ' αυτήν. Το ΣΔΒΔ παίζει τον ρόλο του μεσάζοντα ανάμεσα στον χρήστη και τη βάση δεδομένων και μόνο μέσω του ΣΔΒΔ μπορεί ο χρήστης να αντλήσει πληροφορίες από τη βάση δεδομένων. Ένα ΣΔΒΔ μπορεί να είναι εγκατεστημένο σ' έναν μόνο υπολογιστή ή και σ' ένα δίκτυο υπολογιστών και μπορεί να χρησιμοποιείται από έναν χρήστη ή και από πολλούς χρήστες.

Ένα *Σύστημα Βάσης Δεδομένων (ΣΒΔ)* ή *DBS (Data Base System)* αποτελείται από το υλικό, το λογισμικό, τη βάση δεδομένων και τους χρήστες. Είναι δηλαδή ένα σύστημα με το οποίο μπορούμε να αποθηκεύσουμε και να αξιοποιήσουμε δεδομένα με τη βοήθεια ηλεκτρονικού υπολογιστή. Αναλυτικά :

- Το *υλικό (hardware)* αποτελείται όπως είναι γνωστό από τους ηλεκτρονικούς υπολογιστές, τα περιφερειακά, τους σκληρούς δίσκους, τις μαγνητικές ταινίες κ.ά., όπου είναι αποθηκευμένα τα αρχεία της βάσης δεδομένων αλλά και τα προγράμματα που χρησιμοποιούνται για την επεξεργασία τους.
- Το *λογισμικό (software)* είναι τα προγράμματα που χρησιμοποιούνται για την επεξεργασία των δεδομένων (στοιχείων) της βάσης δεδομένων.
- Η *βάση δεδομένων (data base)* αποτελείται από το σύνολο των αρχείων όπου είναι αποθηκευμένα τα δεδομένα του συστήματος. Τα στοιχεία αυτά μπορεί να βρίσκονται αποθηκευμένα σ' έναν φυσικό υπολογιστή αλλά και σε περισσότερους. Όμως, στον χρήστη δίνεται η εντύπωση ότι βρίσκονται συγκεντρωμένα στον ίδιο υπολογιστή. Τα δεδομένα των αρχείων αυτών είναι *ενοποιημένα (data integration)*, δηλ. δεν υπάρχει πλεονασμός (άσκοπη επανάληψη) δεδομένων και *μερισμένα (data sharing)*, δηλ. υπάρχει δυνατότητα ταυτόχρονης προσπέλασης των δεδομένων από πολλούς χρήστες. Ο κάθε χρήστης έχει διαφορετικά δικαιώματα και βλέπει διαφορετικό κομμάτι της βάσης δεδομένων, ανάλογα με τον σκοπό για τον οποίο συνδέεται.
- Οι *χρήστες (users)* μιας βάσης δεδομένων χωρίζονται στις εξής κατηγορίες :
    - *Τελικοί χρήστες (end users)*. Χρησιμοποιούν κάποια εφαρμογή για να παίρνουν στοιχεία από μια βάση δεδομένων, έχουν τις λιγότερες δυνατότητες επέμβασης στα στοιχεία της βάσης δεδομένων, χρησιμοποιούν ειδικούς κωδικούς πρόσβασης και το σύστημα τούς επιτρέπει ανάλογα πρόσβαση σε συγκεκριμένο κομμάτι της βάσης δεδομένων.
    - *Προγραμματιστές εφαρμογών (application programmers)*. Αναπτύσσουν τις εφαρμογές του ΣΒΔ σε κάποια από τις γνωστές γλώσσες προγραμματισμού.
    - *Διαχειριστής δεδομένων (data administrator – DA)*. Έχει τη διοικητική αρμοδιότητα και ευθύνη για την οργάνωση της βάσης δεδομένων και την απόδοση δικαιωμάτων πρόσβασης στους χρήστες.
    - *Διαχειριστής βάσης δεδομένων (database administrator – DBA)*. Λαμβάνει οδηγίες από τον διαχειριστή δεδομένων και είναι αυτός που διαθέτει τις τεχνικές γνώσεις και αρμοδιότητες για τη σωστή και αποδοτική λειτουργία του ΣΔΒΔ.

2.4.1.5 Η Αρχιτεκτονική των ΣΔΒΔ

Όπως είδαμε νωρίτερα, ένα ΣΔΒΔ (Σύστημα Διαχείρισης Βάσης Δεδομένων) έχει σαν αποστολή τη διαχείριση των δεδομένων των αρχείων της βάσης, δηλ. την προσθήκη, διαγραφή, τροποποίηση εγγραφών, την αναζήτηση μέσα στις εγγραφές κ.ά.). Το ΣΔΒΔ δέχεται αιτήσεις από τους χρήστες των εφαρμογών και επικοινωνεί με τα αρχεία της βάσης δεδομένων για να τις διεκπεραιώσει.

Αυτή η κοινή διεπαφή (interface) των εφαρμογών με τα αρχεία αποκαλείται *λογική διεπαφή*. Οι εφαρμογές που δημιουργούμε δεν απασχολούνται με τον τρόπο που είναι αποθηκευμένα τα δεδομένα, πόσο χώρο καταλαμβάνουν κοκ και αυτή η ιδιότητα είναι γνωστή ως *ανεξαρτησία δεδομένων*.

Αυτό σημαίνει πρακτικά ότι οποιαδήποτε αλλαγή στον τρόπο οργάνωσης των αρχείων της βάσης δεδομένων δεν θα συνεπάγεται και αλλαγή στις εφαρμογές· ένα

πρόβλημα που ταλαιπωρούσε πολύ τους προγραμματιστές παλαιοτέρων εποχών. Ακόμη, η προσθήκη, η κατάργηση ή και η τροποποίηση κάποιων εφαρμογών δεν θα έχει καμία επίπτωση στον τρόπο οργάνωσης των αρχείων της βάσης δεδομένων. Στα ΣΔΒΔ έχει επικρατήσει η λεγόμενη αρχιτεκτονική των τριών επιπέδων (βαθμίδων), όπου τα τρία επίπεδα είναι τα εξής :

- *Εσωτερικό επίπεδο (internal level)*, έχει να κάνει με την αποθήκευση των αρχείων στον σκληρό δίσκο, δηλ. την πραγματική ή φυσική κατάστασή τους.
- *Εξωτερικό επίπεδο (external level)*, έχει να κάνει με τους χρήστες είτε αυτοί είναι απλοί χειριστές, είτε προγραμματιστές ή και οι διαχειριστές της βάσης δεδομένων.
- *Εννοιολογικό επίπεδο (conceptual level)*, είναι ένα ενδιάμεσο επίπεδο που διασυνδέει τα δύο άλλα επίπεδα και έχει να κάνει με τη λογική σχεδίαση των αρχείων της βάσης δεδομένων.

2.4.1.6 Οι Οντότητες (Entities)

Με τον όρο *οντότητα (entity)* εννοούμε ένα αντικείμενο, ένα πρόσωπο, μια κατάσταση και γενικά οτιδήποτε μπορεί να προσδιορισθεί σαν ανεξάρτητη ύπαρξη (αυτόνομη μονάδα του φυσικού κόσμου). Για παράδειγμα, σε μια βάση δεδομένων μιας εμπορικής εταιρείας, οντότητες μπορεί να είναι οι εργαζόμενοι, οι πελάτες, οι προμηθευτές, οι παραγγελίες, τα είδη της αποθήκης (προϊόντα) κ.ά.

Το *Μοντέλο Οντοτήτων Συσχετίσεων (Entity Relationship Model, ER Model)* είναι μια διαγραμματική αναπαράσταση της δομής μιας βάσης δεδομένων και χρησιμοποιείται κατά τη φάση του λογικού σχεδιασμού της βάσης. Δηλαδή, δεν ασχολείται με τον τρόπο που αποθηκεύονται τα δεδομένα της βάσης, αλλά με την ταυτοποίηση των δεδομένων και με τον τρόπο με τον οποίο αυτά συσχετίζονται μεταξύ τους.

Θα δούμε ένα παράδειγμα μιας εταιρείας, η οποία περιέχει δεδομένα που αφορούν τους υπαλλήλους της (employees), τα τμήματά της (departments) και τα έργα (projects) που έχουν αναλάβει αυτά τα τμήματα. Ένα τμήμα της εταιρείας μπορεί να εποπτεύει ένα ή περισσότερα έργα (projects) και ένας υπάλληλος ανήκει σ' ένα μόνο τμήμα της εταιρείας αλλά μπορεί να απασχολείται ταυτόχρονα σε πολλά έργα, τα οποία δεν είναι υποχρεωτικό να παρακολουθούνται από το ίδιο τμήμα.

2.4.1.7 Οι Ιδιότητες (Attributes)

Με τον όρο *ιδιότητα* ή *χαρακτηριστικό* ή και *πεδίο (attribute)* μιας οντότητας, αναφερόμαστε σ' ένα από τα συστατικά της στοιχεία που την περιγράφουν και την κάνουν να ξεχωρίζει από τα άλλα στοιχεία της ίδιας οντότητας. Για παράδειγμα, η οντότητα ΠΕΛΑΤΗΣ μπορεί να έχει ως ιδιότητες (χαρακτηριστικά) τον κωδικό, το επώνυμο, το όνομα, τη διεύθυνση, το τηλέφωνο, το ΑΦΜ κ.ά., με τη βοήθεια των οποίων μπορούμε να ξεχωρίσουμε τους πελάτες μεταξύ τους.

Επίσης, η οντότητα ΠΑΡΑΓΓΕΛΙΑ μπορεί να έχει ως ιδιότητες (χαρακτηριστικά) τον κωδικό, τον αριθμό παραστατικού, την ημερομηνία, τον κωδικό πελάτη, το προϊόν κ.ά., με τη βοήθεια των οποίων μπορούμε να ξεχωρίσουμε τις παραγγελίες μεταξύ τους

Στο παράδειγμα της εταιρείας, μπορούμε να ορίσουμε έναν τύπο οντότητας για τους υπαλλήλους της εταιρείας (EMPLOYEE), έναν τύπο οντότητας για τα τμήματα που έχει η εταιρεία (DEPARTMENT) και έναν τύπο οντότητας για τα έργα που έχει αναλάβει η εταιρεία (PROJECT). Καθένας από τους παραπάνω τύπους οντοτήτων περιγράφεται από ένα όνομα και από το σύνολο των πεδίων που περιέχει. Οι πληροφορίες αυτές αποτελούν το *σχήμα (schema)* της οντότητας.

2.4.1.8 Τα Στιγμιότυπα (Snapshots)

Το κάθε διαφορετικό (αυτόνομο) στοιχείο μιας οντότητας αποκαλείται *στιγμιότυπο (snapshot)* ή και *εμφάνιση* της οντότητας. Για παράδειγμα, στην οντότητα ΠΕΛΑΤΗΣ, άλλο στιγμιότυπο είναι ο πελάτης με επώνυμο Παπαδόπουλος και άλλο στιγμιότυπο είναι ο πελάτης με επώνυμο Σουμπάσης.

2.4.1.9 Το Πρωτεύον Κλειδί (Primary Key)

Πρωτεύον κλειδί ή *πεδίο κλειδί (primary key)* μιας οντότητας καλείται εκείνη η ιδιότητα (ή ο συνδυασμός ιδιοτήτων) που έχει μοναδική τιμή για όλα τα στιγμιότυπα (εμφανίσεις) της οντότητας. Για παράδειγμα, στην οντότητα ΠΕΛΑΤΗΣ πρωτεύον κλειδί είναι ο κωδικός πελάτη, στην οντότητα ΠΑΡΑΓΓΕΛΙΑ πρωτεύον κλειδί μπορεί να είναι ο κωδικός παραγγελίας ή ο αριθμός παραστατικού κοκ.

Υπάρχουν περιπτώσεις όπου το πεδίο κλειδί ενός τύπου οντότητας μπορεί να μην είναι απλό αλλά σύνθετο, να αποτελείται δηλαδή από πολλά απλά πεδία και τότε η συνθήκη της μοναδικότητας για την τιμή του κλειδιού δεν εφαρμόζεται σε κάθε πεδίο του σύνθετου κλειδιού αλλά στο σύνολο του συνδυασμού αυτών των πεδίων.

2.4.1.10 Οι Συσχετίσεις (Relationships)

Με τον όρο *συσχέτιση (relationship)* αναφερόμαστε στον τρόπο σύνδεσης (επικοινωνίας) δύο ξεχωριστών οντοτήτων, ώστε να μπορούμε να αντλούμε στοιχεία (πληροφορίες) από τον συνδυασμό τους.

Για παράδειγμα, η οντότητα ΓΙΑΤΡΟΣ συσχετίζεται με την οντότητα ΑΣΘΕΝΗΣ αλλά και με την οντότητα ΚΛΙΝΙΚΗ στη βάση δεδομένων ενός νοσοκομείου. Μπορούμε να δεχθούμε ότι ένας γιατρός παρακολουθεί (συσχετίζεται με) πολλούς ασθενείς, αλλά ένας ασθενής παρακολουθείται από (συσχετίζεται με) έναν μόνο γιατρό και επίσης ένας γιατρός συσχετίζεται με (ανήκει σε) μία μόνο κλινική, αλλά μια κλινική συσχετίζεται με (απασχολεί) πολλούς γιατρούς.

Στο παράδειγμα της εταιρείας, η οντότητα EMPLOYEE συσχετίζεται με την οντότητα DEPARTMENT και η οντότητα DEPARTMENT συσχετίζεται με την οντότητα PROJECTS. Ένας υπάλληλος ανήκει σ' ένα μόνο τμήμα και ένα τμήμα μπορεί να έχει πολλούς υπαλλήλους. Επίσης, ένα τμήμα εποπτεύει πολλά έργα αλλά ένα έργο εποπτεύεται από ένα μόνο τμήμα.

2.4.1.11 Το Ιεραρχικό Μοντέλο Βάσεων Δεδομένων

Υπάρχουν τρία βασικά μοντέλα που έχουν επικρατήσει στις βάσεις δεδομένων, το ιεραρχικό, το δικτυωτό και το σχεσιακό, και τα οποία αναπτύχθηκαν με βάση αντίστοιχες δομές. Το ιεραρχικό μοντέλο (hierarchical) έχει μια ιεραρχική δομή που θυμίζει δένδρο. Οι οντότητες μοιάζουν με απολήξεις από κλαδιά δένδρων και τοποθετούνται σε επίπεδα ιεραρχίας. Τα κλαδιά παριστάνουν τις συσχετίσεις ανάμεσα στις οντότητες.

Από μια οντότητα που βρίσκεται σ' ένα ανώτερο επίπεδο εκκινούν πολλά κλαδιά, καθένα από τα οποία καταλήγει σε μια οντότητα που βρίσκεται σ' ένα χαμηλότερο επίπεδο. Αλλά, σε κάθε οντότητα που βρίσκεται σ' ένα χαμηλότερο επίπεδο αντιστοιχεί μία και μόνο μία οντότητα που βρίσκεται σ' ένα ανώτερο επίπεδο. Το μοντέλο αυτό ήταν το πρώτο που εμφανίσθηκε αλλά σήμερα θεωρείται δύσχρηστο και ξεπερασμένο.

2.4.1.12 Το Δικτυωτό Μοντέλο Βάσεων Δεδομένων

Και στο δικτυωτό (network) μοντέλο, τα στοιχεία τοποθετούνται σ' ένα επίπεδο ιεραρχίας, αλλά κάθε στοιχείο μπορεί να συσχετισθεί με πολλά στοιχεία είτε σ' ένα κατώτερο ή σ' ένα ανώτερο επίπεδο.

2.4.1.13 Το Σχεσιακό Μοντέλο Βάσεων Δεδομένων

Το σχεσιακό (relational) μοντέλο έχει επικρατήσει σήμερα στην αναπαράσταση των δεδομένων καθώς διαθέτει σημαντικά πλεονεκτήματα ως προς τα άλλα δύο και οι βάσεις δεδομένων που σχεδιάζονται σύμφωνα μ' αυτό αποκαλούνται σχεσιακές (relational databases). Με τις σχεσιακές βάσεις δεδομένων διαθέτουμε έναν σαφή, απλό και εύκολα κατανοητό τρόπο για να μπορέσουμε να αναπαραστήσουμε και να διαχειριστούμε τα δεδομένα μας. Υστερούν μόνο σε ταχύτητα υπολογισμών και σε χώρο αποθήκευσης, αλλά μόνο όταν έχουμε να κάνουμε με πολύ μεγάλες βάσεις δεδομένων.

Στο μοντέλο αυτό οι βάσεις δεδομένων περιγράφονται με αυστηρές μαθηματικές έννοιες και ο χρήστης βλέπει τις οντότητες και τις συσχετίσεις με τη μορφή πινάκων (tables) και σχέσεων (relations) αντίστοιχα.

Ένας *πίνακας (table)* αποτελείται από γραμμές (rows) και στήλες (columns), όπου τοποθετούμε τα στοιχεία σε οριζόντια και κάθετη μορφή. Η κάθε στήλη του πίνακα χαρακτηρίζει κάποια ιδιότητα της οντότητας και αποκαλείται *χαρακτηριστικό (attribute)* ή *πεδίο (field)*, ενώ η κάθε γραμμή του πίνακα περιέχει όλες τις πληροφορίες (στήλες) που αφορούν ένα στοιχείο της οντότητας και αποκαλείται *πλειάδα (tuple)* ή *εγγραφή (record)*.

Κάθε πεδίο του πίνακα μπορεί να πάρει ορισμένες μόνο τιμές, οι οποίες μπορεί να καθορίζονται από τον τύπο δεδομένων της ιδιότητας, όπως ονόματα ή αριθμοί για παράδειγμα, ή και από αυτό που εκφράζει, όπως το ότι δεν μπορούμε να έχουμε αρνητικό βάρος ή αρνητικό ΑΦΜ, για παράδειγμα. Το σύνολο των αποδεκτών τιμών μιας οντότητας αποκαλείται *πεδίο ορισμού (domain)*.

Για να μπορέσουμε να κατανοήσουμε τις σχεσιακές βάσεις δεδομένων, ένα πολύ χαρακτηριστικό παράδειγμα αποτελεί ένας πίνακας πελατών και ένας πίνακας παραγγελιών μιας εμπορικής εταιρείας.

Τα πεδία που μπορούμε να ορίσουμε στους πίνακες αυτούς είναι τα εξής :

ΠΙΝΑΚΑΣ (ΟΝΤΟΤΗΤΑ) ΠΕΛΑΤΕΣ

(ΚωδικόςΠελάτη, Επώνυμο, Όνομα, Διεύθυνση, ΤΚ, Πόλη, ΑΦΜ, Υπόλοιπο)

ΠΙΝΑΚΑΣ (ΟΝΤΟΤΗΤΑ) ΠΑΡΑΓΓΕΛΙΕΣ

(ΚωδικόςΠελάτη, ΚωδικόςΠαραγγελίας, Ημερομηνία, Είδος, Ποσότητα, ΤιμήΜονάδας)

Βλέπουμε ότι οι δύο πίνακες έχουν ένα κοινό πεδίο (στήλη), τον ΚωδικόςΠελάτη, και αυτό είναι απαραίτητο στις σχεσιακές βάσεις δεδομένων για να μπορέσουμε να κάνουμε τη δουλειά μας και να συνδυάσουμε πληροφορίες και από τους δύο πίνακες.

Όπως είναι εύκολα κατανοητό, η βασικότερη εργασία που έχουμε να κάνουμε κατά τον σχεδιασμό μιας σχεσιακής βάσης δεδομένων είναι να ορίσουμε τους πίνακες που θα χρησιμοποιήσουμε καθώς και τα πεδία που θα περιέχει ο καθένας απ' αυτούς. Η διαδικασία αυτή αποκαλείται κατασκευή του *σχήματος (schema)* μιας βάσης δεδομένων.

Οι κανόνες που πρέπει να ακολουθούμε πιστά κατά τον σχεδιασμό μιας σχεσιακής βάσης δεδομένων είναι οι εξής :

- Η κάθε οντότητα πρέπει να παριστάνεται ως ένας ξεχωριστός πίνακας.
- Η κάθε στήλη του πίνακα αντιστοιχεί σε μια ιδιότητα της οντότητας.
- Η κάθε γραμμή του πίνακα αντιστοιχεί σε μια εμφάνιση της οντότητας.
- Η κάθε γραμμή πρέπει να είναι μοναδική, δηλ. αποκλείεται να υπάρχουν δύο ή και περισσότερες γραμμές που να περιέχουν τα ίδια ακριβώς στοιχεία.
- Η σειρά εμφάνισης των γραμμών δεν έχει καμία σημασία.
- Η κάθε στήλη έχει μια δική της μοναδική ονομασία.
- Οι τιμές που ανήκουν στην ίδια στήλη πρέπει να είναι του ιδίου τύπου, δηλ. ή όλες αριθμοί ή όλες αλφαριθμητικές κοκ.
- Η στήλη που αποτελεί το πρωτεύον κλειδί (primary key) μιας οντότητας, δεν πρέπει να είναι ποτέ κενή (null).
- Αποκλείεται να υπάρχουν δύο ή και περισσότερες γραμμές που να περιέχουν την ίδια τιμή στο πρωτεύον κλειδί.
- Το πρωτεύον κλειδί μιας οντότητας αποκαλείται ξένο κλειδί (foreign key) σε μια άλλη οντότητα, με την οποία υπάρχει συσχετισμός.
- Μπορεί να υπάρχουν πολλές γραμμές που να έχουν την ίδια τιμή στο ξένο κλειδί.

2.4.1.14 Τα Σχεσιακά ΣΔΒΔ (RDBMS)

Τα Σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΣΔΒΔ) ή RBMS (Relational DataBase Management Systems) αναπτύχθηκαν με βάση το σχεσιακό μοντέλο και έχουν επικρατήσει πλήρως στον χώρο. Κατά τον σχεδιασμό και τη

δημιουργία μιας σχεσιακής βάσης δεδομένων, οι πίνακες αποτελούν το μοναδικό δομικό και απαραίτητο στοιχείο για μπορέσουν να αναπαρασταθούν οι πληροφορίες που περιέχονται στη βάση δεδομένων.

Για να μπορέσουμε να προσθέσουμε, διαγράψουμε ή τροποποιήσουμε τα στοιχεία που περιέχονται σε μια βάση δεδομένων, χρησιμοποιούμε ειδικές γλώσσες προγραμματισμού που αποκαλούνται *γλώσσες ερωταπαντήσεων (query languages)*. Η γλώσσα που αποτελεί σήμερα ένα διεθνές πρότυπο για την επικοινωνία των χρηστών με τα Σχεσιακά ΣΔΒΔ είναι η *SQL (Structured Query Language)* ή *Δομημένη Γλώσσα Ερωτημάτων*. Μπορεί να λειτουργήσει αυτόνομα αλλά και σε συνεργασία μ' άλλες γλώσσες προγραμματισμού.

Μια άλλη, φιλική προς τον χρήστη γλώσσα προγραμματισμού για να μπορούμε να υποβάλουμε ερωτήματα σε σχεσιακές βάσεις δεδομένων και να λαμβάνουμε απαντήσεις είναι η *QBE (Query By Example)*, η οποία χρησιμοποιεί φόρμες για τη γραφική απεικόνιση των ερωτημάτων μας.

Σήμερα, υπάρχουν εξελιγμένα εργαλεία διαχείρισης σε γραφικό και φιλικό προς τον χρήστη περιβάλλον για να κάνουμε τα εξής :

- Δημιουργία πινάκων
- Δημιουργία φορμών
- Δημιουργία ερωτημάτων
- Δημιουργία εκθέσεων (αναφορών)

Τα Σχεσιακά ΣΔΒΔ τα διακρίνουμε στα *μεγάλα*, τα οποία αφορούν κυρίως μεγάλους οργανισμούς και επιχειρήσεις, έχουν τεράστιο όγκο δεδομένων και πολλούς χρήστες ταυτόχρονα, και τέτοια συστήματα είναι τα Oracle, Ingres, Informix, SQL Server κ.ά. και τα *μικρά*, τα οποία αφορούν κυρίως απλούς χρήστες, όπως είναι η Microsoft Access, η Paradox, η FoxPro κ.ά.

2.4.1.15 Το Μοντέλο Οντοτήτων–Συσχετίσεων

Το μοντέλο που έχει επικρατήσει σήμερα για να παραστήσει τις έννοιες ή τη δομή μιας βάσης δεδομένων είναι το *Μοντέλο Οντοτήτων–Συσχετίσεων (ΟΣ)*. Οι βασικές (θεμελιώδεις) έννοιες του μοντέλου αυτού είναι οι εξής :

- Οντότητες
- Ιδιότητες ή Χαρακτηριστικά
- Συσχετίσεις

Για να αναπαραστήσουμε ένα Μοντέλο Οντοτήτων – Συσχετίσεων χρησιμοποιούμε ειδικά διαγράμματα, όπου τα ορθογώνια συμβολίζουν τις οντότητες, οι ρόμβοι τις συσχετίσεις και οι ελλείψεις τις ιδιότητες. Με ευθείες γραμμές συνδέουμε τις οντότητες που συσχετίζονται με κάποιο τρόπο μεταξύ τους. Όλα τα παραπάνω αποτελούν τη λογική δομή μιας βάσης δεδομένων, μια εργασία που είναι απαραίτητο να γίνει πριν από την καταχώριση και την επεξεργασία των στοιχείων (πληροφοριών) της βάσης δεδομένων.

Το μοντέλο οντοτήτων–συσχετίσεων αποτελεί μια γενική περιγραφή των γενικών στοιχείων που απαρτίζουν μια βάση δεδομένων και απεικονίζει την αντίληψη που έχουμε για τα δεδομένα (εννοιολογικό), χωρίς να υπεισέρχεται σε λεπτομέρειες υλοποίησης.

2.4.1.16 Οι Οντότητες

Με τον όρο *οντότητα (entity)* αναφερόμαστε σε κάθε αντικείμενο, έννοια, πρόσωπο ή κατάσταση που έχει μια ανεξάρτητη ύπαρξη. Είναι κάτι που ξεχωρίζει και μπορούμε να συγκεντρώσουμε πληροφορίες (στοιχεία) γι' αυτό. Η οντότητα είναι αντίστοιχη με την έννοια της εγγραφής που συναντάμε στα αρχεία και στους πίνακες αλλά και με την έννοια του αντικειμένου στις σύγχρονες αντικειμενοστραφείς γλώσσες προγραμματισμού.

Παραδείγματα οντοτήτων είναι τα εξής : ΠΕΛΑΤΗΣ, ΠΑΡΑΓΓΕΛΙΑ, ΠΡΟΜΗΘΕΥΤΗΣ, ΑΠΟΘΗΚΗ, ΜΑΘΗΤΗΣ, ΚΑΘΗΓΗΤΗΣ, ΑΘΛΗΤΗΣ, ΑΓΩΝΙΣΜΑ, ΧΩΡΑ, ΠΟΛΕΙΣ κ.ά.

Μια βάση δεδομένων μπορεί να περιέχει πολλές διαφορετικές οντότητες, οι οποίες απεικονίζονται με ορθογώνια παραλληλόγραμμα και συσχετίζονται μεταξύ τους ανά δύο.

2.4.1.17 Οι Ιδιότητες (Χαρακτηριστικά) των Οντοτήτων

Με τον όρο *ιδιότητες (properties)* ή *χαρακτηριστικά (attributes)* αναφερόμαστε στα συστατικά (δομικά) στοιχεία που προσδιορίζουν (αποτελούν) μια οντότητα. Η ιδιότητα είναι αντίστοιχη με την έννοια του πεδίου που συναντάμε στα αρχεία και στους πίνακες αλλά και με την έννοια της μεταβλητής στις γλώσσες προγραμματισμού.

Για παράδειγμα, η οντότητα ΓΙΑΤΡΟΣ μπορεί να αποτελείται από τις ιδιότητες (χαρακτηριστικά) ΑριθμόςΜητρώου, Επώνυμο, Όνομα, Πατρώνυμο, Ειδικότητα, Βαθμός, ΈτοςΓέννησης, Διεύθυνση, ΑΦΜ, Τηλέφωνο, Κινητό κ.ά., ενώ η οντότητα ΑΘΛΗΤΗΣ μπορεί να αποτελείται από τις ιδιότητες (χαρακτηριστικά) ΚωδικόςΑθλητή, Επώνυμο, Όνομα, Πατρώνυμο, Αγώνισμα, Επίδοση, Σύλλογος, ΈτοςΓέννησης, Διεύθυνση, Τηλέφωνο, Κινητό κ.ά.

Απ' όλες τις ιδιότητες μιας οντότητας, υπάρχει μία μόνο ιδιότητα, και σπανιότερα ένας συνδυασμός δύο ή και περισσοτέρων ιδιοτήτων, η τιμή της οποίας είναι μοναδική και προσδιορίζει την κάθε εμφάνιση (στιγμιότυπο) της οντότητας και αποκαλείται *πρωτεύον κλειδί (primary key)*. Το πρωτεύον κλειδί εμφανίζεται στα διαγράμματα με υπογράμμιση ή με έντονη γραφή ή έχει ως πρόθεμα τον χαρακτήρα #.

Στο διάγραμμα οντοτήτων–συσχετίσεων οι ιδιότητες απεικονίζονται με σχήματα ελλειπτικής μορφής, τα οποία ενώνονται με ευθείες γραμμές με την οντότητα στην οποία ανήκουν.

2.4.1.18 Τα Κλειδιά

Όπως είδαμε και νωρίτερα, με τον όρο *κλειδί (key)* ή πιο σωστά *πρωτεύον κλειδί (primary key)* αναφερόμαστε σε μια ιδιότητα (πεδίο), ή σπανιότερα σ' ένα σύνολο ιδιοτήτων (πεδίων), η τιμή της οποίας είναι μοναδική σ' ολόκληρη την οντότητα (πίνακας). Στην πράξη, το πρωτεύον κλειδί έχει διαφορετική τιμή για κάθε εμφάνιση της

οντότητας ή για κάθε γραμμή (εγγραφή) του πίνακα και ποτέ δεν μπορεί να έχει μηδενική (κενή) τιμή (null). Προσοχή, άλλο πράγμα είναι ο αριθμός 0 και άλλο πράγμα είναι η κενή τιμή (null), δηλ. η μη ύπαρξη τιμής.

Ο συνδυασμός δύο ή και περισσοτέρων ιδιοτήτων (πεδίων) για τη δημιουργία ενός πρωτεύοντος κλειδιού αποκαλείται *σύνθετο κλειδί*. Ένα παράδειγμα σύνθετου κλειδιού θα μπορούσε να είναι ο συνδυασμός των ιδιοτήτων Επώνυμο, Όνομα και Πατρώνυμο, εφόσον φυσικά είμαστε απολύτως βέβαιοι ότι δεν υπάρχουν δύο ή και περισσότερα άτομα με κοινές τιμές στις παραπάνω ιδιότητες.

Ξένο κλειδί αποκαλείται μια ιδιότητα (πεδίο) που είναι πρωτεύον κλειδί σε μια οντότητα (πίνακας) αλλά που υπάρχει και σε μια άλλη οντότητα (πίνακας) σαν απλή ιδιότητα. Τα ξένα κλειδιά είναι απαραίτητα για να μπορέσουμε να κάνουμε τις συσχετίσεις (συνδέσεις, επικοινωνίες) ανάμεσα στις οντότητες (πίνακες).

Για παράδειγμα, στην οντότητα ΣΥΛΛΟΓΟΣ, το πεδίο ΚωδικόςΣυλλόγου είναι πρωτεύον κλειδί, ενώ στην οντότητα ΑΘΛΗΤΗΣ, το πεδίο ΚωδικόςΣυλλόγου είναι ξένο κλειδί και είναι απαραίτητο για να μπορέσουμε να υλοποιήσουμε τη συσχέτιση ΑΝΗΚΕΙ, δηλ. να αντλήσουμε την πληροφορία ποιοι αθλητές ανήκουν σε ποιους συλλόγους. Προφανώς, στην οντότητα ΣΥΛΛΟΓΟΣ, το πεδίο Κωδικός Συλλόγου θα έχει μοναδικές τιμές, ενώ στην οντότητα ΑΘΛΗΤΗΣ, το πεδίο Κωδικός Συλλόγου θα έχει επαναλαμβανόμενες τιμές και αυτό γιατί πολλοί αθλητές θα ανήκουν στον ίδιο σύλλογο, αλλά ένας αθλητής ανήκει υποχρεωτικά σ' έναν και μόνο έναν σύλλογο.

Σ' ένα άλλο παράδειγμα, στην οντότητα ΓΙΑΤΡΟΣ, το πεδίο ΚωδικόςΓιατρού είναι πρωτεύον κλειδί, ενώ στην οντότητα ΑΣΘΕΝΗΣ, το πεδίο ΚωδικόςΓιατρού είναι ξένο κλειδί και είναι απαραίτητο για να μπορέσουμε να υλοποιήσουμε τη συσχέτιση ΠΑΡΑΚΟΛΟΥΘΕΙΤΑΙ, δηλ. να αντλήσουμε την πληροφορία ποιοι ασθενείς παρακολουθούνται από ποιους γιατρούς. Προφανώς, στην οντότητα ΓΙΑΤΡΟΣ, το πεδίο ΚωδικόςΓιατρού θα έχει μοναδικές τιμές, ενώ στην οντότητα ΑΣΘΕΝΗΣ, το πεδίο ΚωδικόςΓιατρού θα έχει επαναλαμβανόμενες τιμές και αυτό γιατί πολλοί ασθενείς θα παρακολουθούνται από τον ίδιο γιατρό, αλλά ένας ασθενής παρακολουθείται μόνο από έναν γιατρό.

Αυτό αποτελεί βέβαια μια παραδοχή που κάνουμε για να μπορέσουμε να υλοποιήσουμε μια συσχέτιση σαν την παραπάνω σε μια βάση δεδομένων ενός Νοσοκομείου, αλλά μπορεί να θεωρήσει κάποιος ότι ένας ασθενής μπορεί να παρακολουθείται από πολλούς γιατρούς ταυτόχρονα, οπότε θα πρέπει να μεταβάλλουμε και τον τρόπο συσχέτισης των παραπάνω οντοτήτων.

2.4.1.19 Οι Συσχετίσεις Μεταξύ Οντοτήτων

Ο σωστός σχεδιασμός και προσδιορισμός των οντοτήτων και των ιδιοτήτων τους αποτελούν τα θεμελιώδη βήματα για τη σωστή σχεδίαση και υλοποίηση μιας βάσης δεδομένων. Μια συσχέτιση συνδέει δύο ή και περισσότερες οντότητες μεταξύ τους και παριστάνεται στο διάγραμμα οντοτήτων–συσχετίσεων μ' έναν ρόμβο.

Οι συσχετίσεις είναι απαραίτητες για να μπορέσουμε να αντλήσουμε πληροφορίες που αφορούν δύο ή και περισσότερες οντότητες, όπως για παράδειγμα ποιοι πελάτες έκαναν παραγγελίες κάποια συγκεκριμένη χρονική περίοδο (συσχέτιση ΠΑΡΑΓΓΕΛΝΕΙ) ή ποιοι αθλητές ανήκουν σε ποιους συλλόγους (συσχέτιση ΑΝΗΚΕΙ) ή

ποιοι αθλητές έλαβαν μέρος σε αγωνίσματα μια συγκεκριμένη χρονιά (συσχέτιση ΣΥΜΜΕΤΕΧΕΙ) κοκ.

Όταν οι οντότητες που συμμετέχουν σε μια συσχέτιση είναι δύο, η συσχέτιση αποκαλείται *διμελής* ή *δυαδική*. Ο βαθμός μιας συσχέτισης είναι ίσος με το πλήθος των οντοτήτων που συμμετέχουν σ' αυτήν. Μια συσχέτιση μπορεί και η ίδια να έχει ιδιότητες που να περιγράφουν ορισμένα χαρακτηριστικά της, όπως για παράδειγμα η συσχέτιση ΠΑΡΑΓΓΕΛΙΑ ανάμεσα στις οντότητες ΠΕΛΑΤΗΣ και ΠΡΟΪΟΝ μπορεί να περιέχει τις ιδιότητες (πεδία) ΚωδικόςΠελάτη, ΚωδικόςΠροϊόντος, ΚωδικόςΠαραγγελίας, ΗμερομηνίαΠαραγγελίας, Ποσότητα κ.ά.

Στην περίπτωση αυτή το σωστό είναι να δημιουργήσουμε μια ακόμα οντότητα, την οντότητα ΠΑΡΑΓΓΕΛΙΑ, η οποία και θα περιέχει όλες τις παραπάνω ιδιότητες, και να μετονομάσουμε την προηγούμενη συσχέτιση από ΠΑΡΑΓΓΕΛΙΑ σε ΣΥΝΑΛΛΑΓΗ, που δεν θα περιέχει τώρα ιδιότητες. Έτσι, η παραπάνω συσχέτιση θα μετατραπεί από διμελή σε τριμελή.

Όταν σχεδιάζουμε μια βάση δεδομένων, θα πρέπει να εκχωρούμε ιδιότητες μόνο στις οντότητες και να έχουμε τις συσχετίσεις απλά και μόνο για να κατανοούμε τις λογικές συνδέσεις ανάμεσα στις οντότητες.

2.4.1.20 Οι Διμελείς Συσχετίσεις

Οι διμελείς συσχετίσεις μεταξύ οντοτήτων είναι αυτές που θα μας απασχολήσουν ιδιαίτερα και υπάρχουν τρία βασικά είδη συνδέσεων σ' αυτές, τα εξής :

- *Ένα-προς-ένα (1:1)*, όπου μια εμφάνιση της μιας οντότητας συνδέεται με μία και μόνο μία εμφάνιση της άλλης οντότητας. Για παράδειγμα, η οντότητα ΣΥΛΛΟΓΟΣ έχει έναν μόνο προπονητή, ενώ η οντότητα ΠΡΟΠΟΝΗΤΗΣ συνδέεται μ' έναν και μόνο έναν σύλλογο. Σ' ένα άλλο παράδειγμα, η οντότητα ΝΟΜΟΣ έχει μία μόνο πόλη σαν πρωτεύουσα, ενώ η οντότητα ΠΡΩΤΕΥΟΥΣΑ αντιστοιχεί σ' έναν και μόνο έναν νομό. Στην περίπτωση των διμελών συσχετίσεων του τύπου ένα-προς-ένα, μπορούμε να ενώσουμε τα στοιχεία και των δύο ιδιοτήτων και να δημιουργήσουμε μια μοναδική οντότητα (πίνακα).
- *Ένα-προς-πολλά (1:M)*, όπου μια εμφάνιση της μιας οντότητας συνδέεται με πολλές εμφανίσεις της άλλης οντότητας αλλά κάθε εμφάνιση της δεύτερης οντότητας συνδέεται με μία και μόνο μία εμφάνιση της πρώτης οντότητας. Για παράδειγμα, ένας ΠΕΛΑΤΗΣ κάνει πολλές παραγγελίες, αλλά μια ΠΑΡΑΓΓΕΛΙΑ αντιστοιχεί σ' έναν και μόνο έναν πελάτη. Σ' ένα άλλο παράδειγμα, ένας ΣΥΛΛΟΓΟΣ έχει πολλούς αθλητές, αλλά ένας ΑΘΛΗΤΗΣ ανήκει σ' έναν και μόνο έναν σύλλογο. Οι διμελείς συσχετίσεις του τύπου ένα-προς-ένα είναι οι πιο συχνά συναντώμενες και οι πιο βολικές στη διαχείριση.
- *Πολλά-προς-πολλά (M:N)*, όπου σε μια εμφάνιση της μιας οντότητας αντιστοιχούν πολλές εμφανίσεις της άλλης οντότητας και σε κάθε εμφάνιση της δεύτερης οντότητας αντιστοιχούν πολλές εμφανίσεις της πρώτης οντότητας. Για παράδειγμα, ένας ΑΘΛΗΤΗΣ συμμετέχει σε πολλούς αγώνες αλλά και σ' έναν ΑΓΩΝΑ λαμβάνουν μέρος πολλοί αθλητές. Σ' ένα άλλο παράδειγμα, ένας ΚΑΘΗΓΗΤΗΣ διδάσκει σε πολλούς μαθητές αλλά και ένας ΜΑΘΗΤΗΣ διδάσκεται από πολλούς καθηγητές. Για να μπορέσουμε να διαχειριστούμε μια διμελή σχέση του τύπου πολλά-προς-πολλά, θα πρέπει να δημιουργήσουμε έναν τρίτο πίνακα που θα περιέχει δύο μόνο ιδιότητες (πεδία), δηλ. τα πεδία κλειδιά

των δύο οντοτήτων, οπότε ο συνδυασμός τους θα είναι και το πεδίο κλειδί (σύνθετο κλειδί) του νέου πίνακα.

2.4.1.21 Το Διάγραμμα Οντοτήτων Συσχετίσεων

Για να μπορέσουμε να διαμορφώσουμε το διάγραμμα οντοτήτων συσχετίσεων, θα πρέπει να ακολουθήσουμε τα εξής βήματα :

- Να ορίσουμε τις οντότητες (πίνακες) που θα ανήκουν στη βάση δεδομένων που θέλουμε να δημιουργήσουμε.
- Να ορίσουμε τις ιδιότητες (πεδία) και τα πρωτεύοντα κλειδιά της κάθε οντότητας (πίνακα).
- Να ορίσουμε τις συσχετίσεις ανάμεσα στις οντότητες.
- Δημιουργούμε το διάγραμμα οντοτήτων συσχετίσεων, όπου θα απεικονίσουμε τις οντότητες, τις ιδιότητές τους και τις συσχετίσεις τους.

Θα δούμε το διάγραμμα οντοτήτων συσχετίσεων για μια βάση δεδομένων με ομάδες (συλλόγους) ποδοσφαίρου, όπου θα έχουμε τις οντότητες ΑΘΛΗΤΗΣ, ΣΥΛΛΟΓΟΣ, ΠΡΟΠΟΝΗΤΗΣ και ΑΓΩΝΑΣ. Οι συσχετίσεις ανάμεσα στις οντότητες αυτές θα είναι οι εξής :

- ΑΘΛΗΤΗΣ – ΣΥΛΛΟΓΟΣ : ένα-προς-πολλά (1:Μ)
- ΣΥΛΛΟΓΟΣ – ΠΡΟΠΟΝΗΤΗΣ : ένα-προς-ένα (1:1)
- ΑΘΛΗΤΗΣ – ΑΓΩΝΑΣ : πολλά-προς-πολλά (Μ:Ν)

Λογικός Σχεδιασμός μιας Βάσης Δεδομένων

Αφού έχουμε δημιουργήσει το διάγραμμα οντοτήτων συσχετίσεων και έχουμε επιλέξει το σχεσιακό μοντέλο δεδομένων για την υλοποίηση της βάσης δεδομένων, ακολουθούμε τη διαδικασία της κανονικοποίησης και είμαστε έτοιμοι για την καταχώριση των στοιχείων της βάσης δεδομένων. Ανάλογα τώρα με το είδος της διμελούς συσχέτισης, διακρίνουμε τις εξής περιπτώσεις ως προς τον λογικό σχεδιασμό που θα πρέπει να ακολουθήσουμε :

Αν η συσχέτιση των δύο πινάκων είναι ένα-προς-ένα, τότε μπορούμε είτε να συνενώσουμε τους δύο πίνακες, με τις αντίστοιχες εγγραφές φυσικά, ή να προσθέσουμε το ένα από τα δύο πεδία κλειδιά σαν ξένο κλειδί στον άλλον πίνακα ή τέλος να δημιουργήσουμε έναν καινούργιο πίνακα με μόνα πεδία τα πεδία κλειδιά των δύο πινάκων (σύνθετο κλειδί). Η προτιμότερη διαδικασία είναι η πρώτη, δηλ. η συνένωση των δύο πινάκων σ’ έναν ενιαίο πίνακα.

Αν η συσχέτιση των δύο πινάκων είναι ένα-προς-πολλά, τότε μπορούμε είτε να προσθέσουμε το ένα από τα δύο πεδία κλειδιά σαν ξένο κλειδί στον άλλον πίνακα ή να δημιουργήσουμε έναν καινούργιο πίνακα με μόνα πεδία τα πεδία κλειδιά των δύο πινάκων (σύνθετο κλειδί). Η προτιμότερη διαδικασία είναι η πρώτη, δηλ. η προσθήκη του ξένου κλειδιού στην πλευρά ‘πολλά’ της σχέσης.

Αν η συσχέτιση των δύο πινάκων είναι πολλά-προς-πολλά, τότε το μόνο που μπορούμε και πρέπει να κάνουμε είναι να δημιουργήσουμε έναν καινούργιο πίνακα με μόνα πεδία τα πεδία κλειδιά των δύο πινάκων (σύνθετο κλειδί), όπου το κάθε πεδίο κλειδί από μόνο του γίνεται ξένο κλειδί. Οι δύο αρχικοί πίνακες δεν μεταβάλλονται.

2.4.1.22 Η Κανονικοποίηση

Τα προβλήματα που είναι πιθανό να παρουσιασθούν κατά τη διαδικασία της υλοποίησης του σχεδιασμού μιας βάσης δεδομένων είναι η περιττή (άσκοπη) επανάληψη πληροφοριών, που είναι γνωστή με τον όρο redundancy, καθώς και δυσκολίες στην ενημέρωση της βάσης δεδομένων. Τα παραπάνω προβλήματα είναι γνωστά ως *πλεονασμοί δεδομένων* και *ανωμαλίες ενημέρωσης* και για να αντιμετωπιστούν με επιτυχία, θα πρέπει να διασπάσουμε τις μεγάλες σχέσεις σε μικρότερες. Αυτό γίνεται με τη διαδικασία της κανονικοποίησης, έτσι ώστε η βάση δεδομένων να είναι έτοιμη για καταχώριση στοιχείων.

Η *κανονικοποίηση (normalization)* είναι μια τεχνική που ασχολείται με την ανάλυση των σχέσεων (συσχετίσεων) σε μια βάση δεδομένων, όπου κάνουμε μετατροπή των αρχικών μεγάλων σχέσεων σε μικρότερες.

2.4.1.23 Πλεονασμός Δεδομένων και Ανωμαλίες Ενημέρωσης

Με τον όρο πλεονασμός δεδομένων (data redundancy) εννοούμε την άσκοπη επανάληψη στοιχείων (πληροφοριών). Τα προβλήματα που προκύπτουν από τον πλεονασμό δεδομένων είναι γνωστά με τον όρο ανωμαλίες ενημέρωσης (update anomalies). Για να μπορέσουμε να κατανοήσουμε τις παγίδες του πλεονασμού δεδομένων, θα δούμε ένα παράδειγμα με τους πίνακες ΠΕΛΑΤΗΣ και ΠΑΡΑΓΓΕΛΙΑ. Η σχέση μεταξύ τους είναι ένα-προς-πολλά, δηλ. ένας πελάτης μπορεί να κάνει πολλές παραγγελίες, αλλά μια παραγγελία γίνεται μόνο από έναν πελάτη.

Αν αποφασίσουμε να δημιουργήσουμε έναν μόνον πίνακα, όπου σε κάθε γραμμή (εγγραφή) του θα υπάρχουν όλα τα στοιχεία της παραγγελίας και δίπλα όλα τα στοιχεία του πελάτη που έχει κάνει την παραγγελία, τότε θα έχουμε πέσει στην παγίδα του πλεονασμού δεδομένων και αυτό γιατί τα στοιχεία του κάθε πελάτη θα επαναλαμβάνονται για κάθε παραγγελία που έχει κάνει.

Ανωμαλία εισαγωγής έχουμε στην περίπτωση που θελήσουμε να καταχωρήσουμε μια καινούργια παραγγελία, οπότε θα πρέπει να καταχωρήσουμε εκ νέου κι όλα τα στοιχεία του πελάτη που έκανε τη συγκεκριμένη παραγγελία, κάτι που είναι κουραστικό, χρονοβόρο και περιέχει τον κίνδυνο λαθών.

Ένα άλλο πρόβλημα με ανωμαλία εισαγωγής έχουμε στην περίπτωση που θελήσουμε να καταχωρήσουμε ένα καινούργιο πελάτη ο οποίος δεν έχει κάνει ακόμα καμία παραγγελία, οπότε η βάση δεδομένων δεν θα μας το επιτρέψει και αυτό γιατί δεν δέχεται κενή τιμή (null) για το πεδίο κλειδί Κωδικός Παραγγελίας.

Ανωμαλία διαγραφής έχουμε στην περίπτωση που θελήσουμε να διαγράψουμε μια παραγγελία που είναι η μοναδική ενός πελάτη, οπότε θα χάσουμε και όλες τις πληροφορίες του συγκεκριμένου πελάτη.

Ανωμαλία τροποποίησης έχουμε στην περίπτωση που θελήσουμε να αλλάξουμε κάποιο στοιχείο ενός πελάτη, όπως τη διεύθυνση ή το τηλέφωνό του, οπότε θα πρέπει να τροποποιήσουμε όλες τις εγγραφές του πίνακα όπου εμφανίζεται ο συγκεκριμένος πελάτης. Αν δεν κάνουμε την αλλαγή σ' όλες τις εγγραφές, τότε ο πελάτης θα εμφανίζεται να έχει δύο διευθύνσεις ή δύο τηλέφωνα κοκ. Μιλάμε τότε για *μη συνεπή (inconsistent)* βάση δεδομένων.

Ως γνωστόν, η λύση στο παραπάνω πρόβλημα είναι να δημιουργήσουμε έναν πίνακα με τα στοιχεία των πελατών και έναν ξεχωριστό πίνακα με τα στοιχεία των παραγγελιών, όπου θα υπάρχει και σαν πεδίο (ξένο κλειδί) ο ΚωδικόςΠελάτη.

2.4.1.24 Η Διαδικασία της Κανονικοποίησης

Η μέθοδος της κανονικοποίησης βοηθάει στον λογικό σχεδιασμό μιας βάσης δεδομένων και είναι συμπληρωματική του μοντέλου οντοτήτων συσχετίσεων. Το κέρδος για μας είναι ότι δεν υπάρχουν προβλήματα συνέπειας, πλεονασμού και εγκυρότητας των πληροφοριών της βάσης δεδομένων. Ακολουθώντας τη διαδικασία της κανονικοποίησης κάνουμε συνεχείς διασπάσεις των πινάκων σε πιο απλές και συμπαγείς μορφές, με στόχο πάντα να αποφύγουμε τον πλεονασμό (επανάληψη) των δεδομένων. Αφαιρούμε πεδία από τις αρχικές μεγάλες σχέσεις και τα τοποθετούμε σε νέες σχέσεις έτσι ώστε να μπορούμε να έχουμε τις ίδιες πληροφορίες και με τις νέες σχέσεις.

Μπορούμε να πούμε ότι *κανονικοποίηση (normalization)* είναι η διαδικασία μετατροπής των δεδομένων κάποιων σχέσεων (πινάκων) σε πιο απλές και πιο σαφείς σχέσεις, χωρίς πλεονασμούς (επαναλήψεις) των δεδομένων. Οι βασικές μορφές της κανονικοποίησης είναι τρεις, η πρώτη ($1^{η}$ NF), η δεύτερη ($2^{η}$ NF) και η τρίτη ($3^{η}$ NF).

MySQL

MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα (relational database management system – RDBMS), η οποία χρησιμοποιεί την Structured Query Language (SQL), την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία βάση δεδομένων.

Λόγω του ότι είναι ανοιχτού κώδικα, οποιοσδήποτε μπορεί να την χρησιμοποιήσει δωρεάν και να την διαμορφώσει σύμφωνα με τις ανάγκες του και σύμφωνα με την γενική άδεια που υπάρχει. Η MySQL είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία κα την ευελιξία που παρέχει. Μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix και Windows.

2.5 Εφαρμογές ανοικτού κώδικα

2.5.1.1 Λογισμικό Ανοιχτού Κώδικα

Για την καλύτερη κατανόηση όλων των θεμάτων της εργασίας, πρέπει να κατανοηθεί στο μεγαλύτερο δυνατό βαθμό, η έννοια του λογισμικού ανοιχτού κώδικα (ΛΑΚ) (Open Source Software). Είναι επιτακτική, δηλαδή, η ανάγκη να δοθεί ο ορισμός του συγκεκριμένου λογισμικού. Το λογισμικό ανοιχτού κώδικα είναι ένα μοντέλο διανομής λογισμικού, όπου η διανομή του λογισμικού γίνεται μαζί με τον πηγαίο κώδικα

(source code), σε αντίθεση με τα εμπορικά πακέτα λογισμικού1. Αυτή η φιλοσοφία, η οποία διαπνέει το εν λόγω λογισμικό, δίνει τη δυνατότητα επέμβασης στον κώδικα με στόχο την προσαρμογή του λογισμικού στις ανάγκες και τις απαιτήσεις του κάθε χρήστη2. Επιπρόσθετα, δίνεται η δυνατότητα βελτίωσης και διόρθωσης λαθών του λογισμικού, μέσω επεμβάσεων στον κώδικα, από κάθε ενδιαφερόμενο. Η διάθεση του λογισμικού ανοιχτού κώδικα, μπορεί να ξεκινήσει από οποιονδήποτε (οργανισμό ή ιδιώτη), ο οποίος θα «γράψει» την αρχική έκδοση και στη συνέχεια θα το διαθέσει στο ευρύτερο σύνολο χρηστών και όχι μόνο.

Το περιβάλλον του διαδικτύου έχει προσφερθεί και προσφέρεται ακόμα και σήμερα για τη διάδοση (και όχι μόνο) του λογισμικού ανοιχτού κώδικα. Έχουν δημιουργηθεί κοινότητες χρηστών και προγραμματιστών, οι οποίες συμβάλλουν στη διαρκή βελτίωση του λογισμικού και γίνεται μια εξαιρετικής σημασίας προσπάθεια για τη διάδοση του συγκεκριμένου λογισμικού. Δε θα έπρεπε σε καμία περίπτωση να παραλειφθεί το γεγονός ότι η διαδικασία ανάπτυξης του διαδικτύου βασίζεται εν πολλοίς στο λογισμικό ανοιχτού κώδικα.

Στο σημείο αυτό κρίνεται απαραίτητο να δοθεί μια διευκρίνιση. Έχει υπάρξει έντονος προβληματισμός και εκτενείς συζητήσεις έχουν γίνει σχετικά με το κόστος του λογισμικού ανοιχτού κώδικα. Αυτό συμβαίνει γιατί πολλές φορές το λογισμικού ανοιχτού κώδικα, αναφέρεται και ως ελεύθερο λογισμικό. Αυτό προήλθε από την αγγλική βιβλιογραφία, η οποία αναφέρει ως «free software» το λογισμικό αυτό. Ωστόσο, η αγγλική λέξη «free» δεν υπονοεί τη δωρεάν διάθεση του λογισμικού, αλλά την ελεύθερη διάθεσή του, ήτοι της ελεύθερης πρόσβασης στον πηγαίο κώδικα. Το πιο αντιπροσωπευτικό παράδειγμα της συγκεκριμένης κατάστασης είναι το λειτουργικό σύστημα Linux, το οποίο ανήκει στην κατηγορία του ΛΑΚ. Μολονότι, οι περισσότερες διανομές του Linux διανέμονται δωρεάν, διάφορες εταιρίες, οι οποίες δημιουργούν διανομές βάσει του Linux, χρεώνουν τις διανομές αυτές (π.χ. RedHat κ.λπ)3.

Το θέμα των πνευματικών δικαιωμάτων δεν απασχολεί ιδιαίτερα, όταν πρόκειται για ένα λογισμικό που διατίθεται κάτω από μία άδεια λογισμικού ανοικτού κώδικα. Αυτό που κυρίως ενδιαφέρει είναι να κατασκευαστεί το λογισμικό. Πάντως, μαζί με τον πηγαίο κώδικα διατίθενται και εμπορικά λογισμικά, τα δικαιώματα των οποίων παραμένουν στον κατασκευαστή τους.

- Το Open Source Initiative (OSI)4, δίνει κάποιες κατευθυντήριες γραμμές και καθορίζει το λογισμικό ανοιχτού κώδικα, βάσει των ακόλουθων δικαιωμάτων και υποχρεώσεων:
    - Δεν επιβάλλεται στην αναδιανομή του ανοιχτού κώδικα καμιά αμοιβή δικαιωμάτων εκμετάλλευσης ή άλλων
    - Ο πηγαίος κώδικας επιβάλλεται να διατίθεται
    - Υπάρχει αναφαίρετο δικαίωμα για τροποποιήσεις και παράγωγες εργασίες
    - Όλα τα δικαιώματα που χορηγούνται πρέπει να διατηρούνται στις αναδιανεμημένες εκδόσεις
    - Η άδεια ισχύει συνολικά για το πρόγραμμα, αλλά και για κάθε ένα από τα συστατικά του

2.5.1.2 Βιβλιοθήκες και χρήση λογισμικού ανοιχτού κώδικα

Στη σημερινή εποχή, όπου το ΛΑΚ έχει εξελιχθεί ικανοποιητικά και αποτελεί σε πολλές περιπτώσεις δυνατό ανταγωνιστή των εμπορικών λογισμικών, πολλοί τομείς και κλάδοι πολλών επιστημών στρέφονται στο ΛΑΚ, ως λύση για τις ανάγκες τους. Σε αυτό

το κλίμα δεν έμεινε ανεπηρέαστος και ο χώρος των βιβλιοθηκών. Δεν είναι λίγα τα προγράμματα ανοιχτού κώδικα, τα οποία κυκλοφορούν και είναι εξειδικευμένα για το χώρο των βιβλιοθηκών. Άμεσο επακόλουθο είναι η στροφή πολλών βιβλιοθηκών σε αυτού του είδος το λογισμικό. Όπως είναι φυσικό, το μεγαλύτερο ενδιαφέρον στο ΛΑΚ, δίνεται από τις ακαδημαϊκές βιβλιοθήκες.

Το λογισμικό ανοιχτού κώδικα, το οποίο προορίζεται για τις βιβλιοθήκες διακρίνεται σε δύο κατηγορίες:

Ολοκληρωμένα προγράμματα αυτοματοποίησης βιβλιοθηκών: Πρόκειται για λογισμικό ανοιχτού κώδικα, το οποίο προορίζεται να διεκπεραιώνει τις κυριότερες λειτουργίες της βιβλιοθήκης (π.χ. Koha).

Πρόγραμμα για την κάλυψη συγκεκριμένων αναγκών: Τα προγράμματα αυτά δημιουργήθηκαν για την κάλυψη μιας ή μερικών λειτουργιών της βιβλιοθήκης (π.χ. Fedora)

Σε οποιαδήποτε από τις δύο περιπτώσεις, η βιβλιοθήκη χρειάζεται να είναι ιδιαίτερα προσεχτική στην επιλογή του προγράμματος. Όπως ερευνάται σε επόμενο κεφάλαιο (3. Συμπεράσματα), υπάρχουν εξίσου σημαντικά πλεονεκτήματα και μειονεκτήματα στο ΛΑΚ, που αφορά στις βιβλιοθήκες. Συν τοις άλλοις, οι ανάγκες των βιβλιοθηκών για λογισμικό διαφέρουν βάσει πολλών παραγόντων (π.χ. είδος βιβλιοθήκης).

2.6 Eclipse

G-Eclipse

2.6.1.1 Το g-Eclipse είναι ένα ελεύθερο λογισμικό προοριζόμενο για προγραμματισμό και διαχείριση υπολογιστικών πλεγμάτων και στηριζόμενο στην πλατφόρμα ανάπτυξης εφαρμογών Eclipse, αποτελώντας κατ' ουσίαν ένα σύνολο επεκτάσεων αυτής.

Το g-Eclipse είναι βασισμένο στην εγγενή αρχιτεκτονική επεκτάσεων του Eclipse, επαυξάνοντας τη λειτουργικότητα του τελευταίου και παρέχοντας πρόσβαση σε υφιστάμενες υποδομές πλέγματος. Το g-Eclipse παρέχει λειτουργίες που διευκολύνουν τον χρήστη στη διαχείριση ενός πλέγματος και των επιμέρους εργασιών λογισμικού που εκτελούνται σε αυτό, αλλά και στην ανάπτυξη εφαρμογών για υπολογιστικά πλέγματα.

Το g-eclipse παρέχει σημεία επέκτασης τα οποία διευκολύνουν την επέκτασή του. Αναπτύσσεται από μία κοινοπραξία που αποτελείται από το Forschungszentrum Karlsruhe, το Poznań Supercomputing and Networking Center, το Πανεπιστήμιο Γιοχάνες Κέπλερ του Λιντς, το Πανεπιστήμιο Κύπρου, την εταιρεία Innoopract Gmbh, το Πανεπιστήμιο του Ρέντινγκ, το IT Innovation Centre, το Πανεπιστήμιο του Σαουθάμπτον και το εταιρικό 6ο Πρόγραμμα Πλαίσιο για εργαστήριο NEC Laboratories Europe. Η κοινοπραξία του g-Eclipse χρηματοδοτείται από το την Έρευνα.

2.6.1.2 *Χαρακτηριστικά*

Το g-Eclipse παρέχει τρείς διαφορετικές «προοπτικές» (τρόπους λειτουργίας):

- Προοπτική Χρήστη: Οι χρήστες του πλέγματος έχουν περιορισμένη τεχνογνωσία σε σχέση με τα υπολογιστικά πλέγματα. Διαμέσου της *Προοπτικής Χρήστη* παρέχεται η δυνατότητα στον χρήστη να εκτελέσει εφαρμογές στο πλέγμα. Παράλληλα δίνεται η δυνατότητα παρακολούθησης της προόδου εκτέλεσης της εφαρμογής καθώς και διαχείρισης αρχείων δεδομένων.
- Προοπτική Χειριστή: Οι χειριστές υποδομών υπολογιστικού πλέγματος κατέχουν την τεχνογνωσία διαχείρισης τέτοιων υποδομών. Η *Προοπτική Χειριστή* πρέπει να παρέχει την ευχέρεια στους χειριστές τέτοιων υποδομών να διαχειρίζονται τοπικούς πόρους καθώς και πόρους που ανήκουν στον εικονικό οργανισμό όπου ανήκουν.
- Προοπτική Προγραμματιστή: Οι προγραμματιστές εφαρμογών για το υπολογιστικό πλέγμα κατέχουν την απαραίτητη τεχνογνωσία για ανάπτυξη εφαρμογών σε τέτοιο περιβάλλον αλλά δεν αναμένεται από αυτούς να γνωρίζουν εις βάθος την τεχνολογία πίσω από το υπολογιστικό πλέγμα ή τις υποδομές που το στηρίζουν. Η *Προοπτική Προγραμματιστή* επιδιώκει να προσφέρει στους προγραμματιστές τα κατάλληλα εργαλεία για την ανάπτυξη εφαρμογών (μέσω JDT/CDT) και εξουδετέρωση τυχόν σφαλμάτων στο υπολογιστικό πλέγμα.

Υποστηριζόμενα ενδιάμεσα λογισμικά

Το g-Eclipse έχει σχεδιαστεί έχοντας ως κεντρικό άξονα ένα μοντέλο ανεξάρτητο από οποιαδήποτε υλοποίηση ενδιάμεσου λογισμικού. Η υποστήριξη κάποιου συγκεκριμένου ενδιάμεσου λογισμικού υλοποιείται με την επέκταση του ανεξάρτητου μοντέλου. Στο παρόν στάδιο το g-Eclipse υποστηρίζει το gLite Middleware, διαμέσου της υλοποίησης επεκτάσεων υπεύθυνων για τη διαχείριση εικονικών οργανισμών, τη διαχείριση εργασιών και αρχείων, την παρακολούθηση υποδομής και εφαρμογών καθώς και για τον σχεδιασμό workflows. Η αρχική υποστήριξη ενός δευτέρου ενδιάμεσου λογισμικού με την ονομασία GRIA, έχει ξεκινήσει. Παράλληλα μία νέα επέκταση υλοποιείται για πρόσβαση στην υπηρεσία Amazon Simple Storage.

2.7 Netbeans

2.7.1.1 Το εργαλείο NetBeans παρέχει μια πλατφόρμα για την ανάπτυξη διάφορων εφαρμογών που είτε θα χρησιμοποιηθούν στα πλαίσια κάποιου δικτύου είτε θα χρησιμοποιηθούν ως ανεξάρτητες (Stand-alone) εφαρμογές. Το εργαλείο NetBeans υποστηρίζει διάφορες γλώσσες

προγραμματισμού. (Java, C, C++, PHP, Python,JavaScript).Παρέχει ένα πλήρες περιβάλλον ανάπτυξης εφαρμογών(IDE: Integrated Development Enviroment).

Το περιβάλλον ανάπτυξης γενικά παρέχει βοηθήματα στον προγραμματιστή τα οποία κάνουν ευκολότερη και αποδοτικότερη την ανάπτυξη των εφαρμογών.Τα βασικότερα από τα βοηθήματα αυτά αφορούν κυρίως τον Source Code Editor (Επισήμανση συντακτικού στον κώδικα, live επισήμανση λαθών και live parsing, pop up παράθυρα βοήθειας).

Παρέχονται βοηθήματα για κατασκευή GUI, αλλά και βοηθήματα διαχείρισης βάσεων δεδομένων.

3 Εργαλεία που χρησιμοποιήθηκαν

3.1 WampServer

3.1.1 Εγκατάσταση προγραμμάτων

Για την εγκατάσταση των τεχνολογιών θα χρησιμοποιήσουμε το πρόγραμμα WAMPServer το οποίο εγκαθιστά τον Apache web server, τον μεταγλωττιστή PHP και την βάση δεδομένων MySQL. Η εγκατάσταση αυτών των προγραμμάτων μπορεί να γίνει και ανεξάρτητα, όμως οι απαραίτητες ρυθμίσεις ώστε να συνεργάζονται είναι αρκετά πολύπλοκες και κάτι τέτοιο δεν προσφέρει κάποιο ιδιαίτερο πλεονέκτημα.

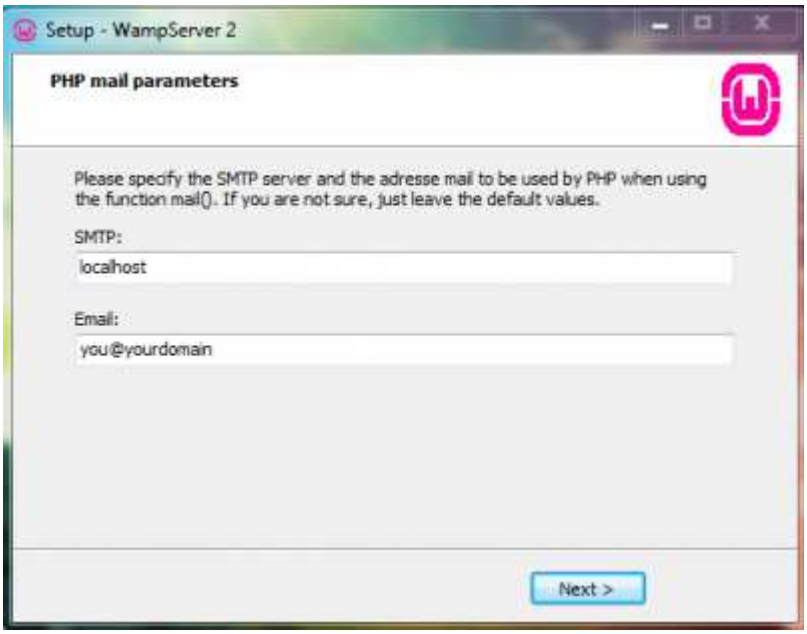
Ξεκινώντας χρειάζεται το αρχείο εγκατάστασης του WAMPServer από την διεύθυνση http://www.wampserver.com/en/. Εκτελώντας το αρχείο αυτό εμφανίζονται οι εκδόσεις των επιμέρους προγραμμάτων που θα εγκατασταθούν.



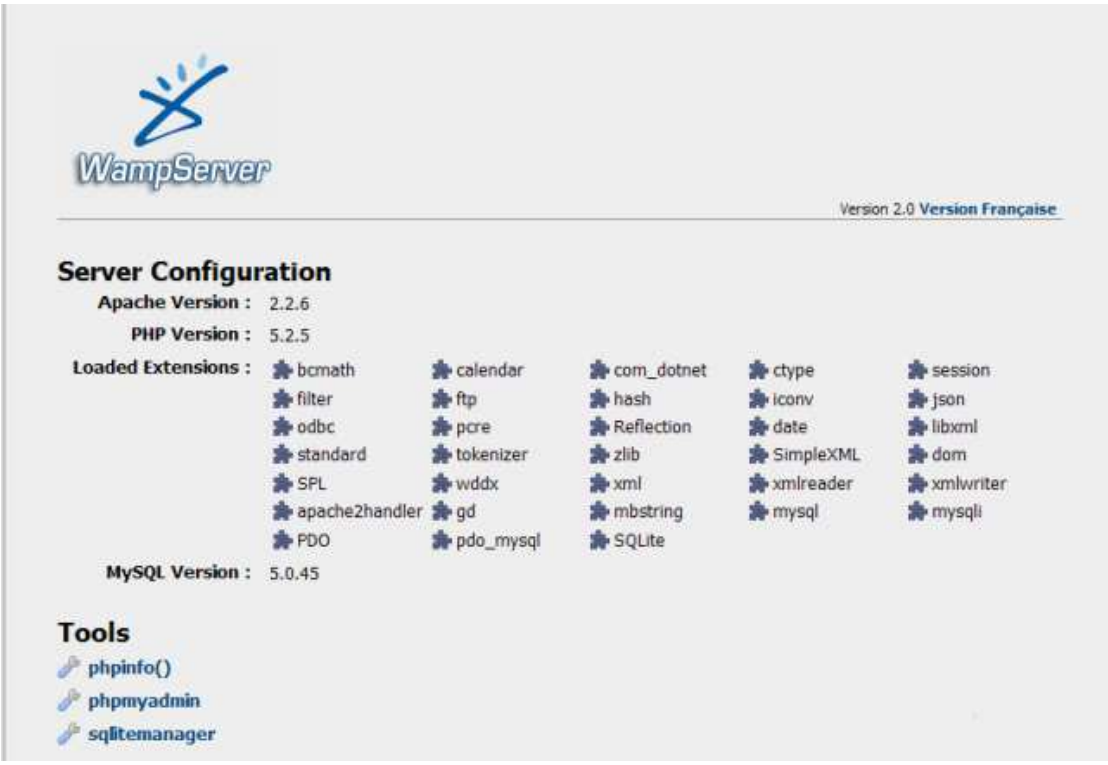
Προχωρώντας μας ζητείται ο φάκελος στον οποίο θα εγκατασταθεί. Καλό είναι να μην βρίσκονται μέσα στον φάκελο Program Files καθώς μπορεί να δημιουργηθούν προβλήματα με τα δικαιώματα χρηστών των Windows. Ο προτεινόμενος φάκελος C:/wamp δεν παρουσιάζει προβλήματα



Μετά την εγκατάσταση ζητούνται η διεύθυνση ενός διακομιστή email και η διεύθυνση του διαχειριστή του συστήματος. Αν χρησιμοποιούσαμε έναν κανονικό server που είχε την δυνατότητα αποστολής email θα δίναμε τις κατάλληλες ρυθμίσεις, όμως καθώς ο τοπικός υπολογιστής δεν έχει τέτοια δυνατότητα αφήνουμε τις ρυθμίσεις ως έχουν.



Με το τέλος της εγκατάστασης ελέγχουμε αν όλα λειτουργούν σωστά τρέχοντας το Wamp ώστε να ξεκινήσουν οι servers και μετά ανοίγουμε τον browser και πηγαίνουμε στην διεύθυνση localhost.

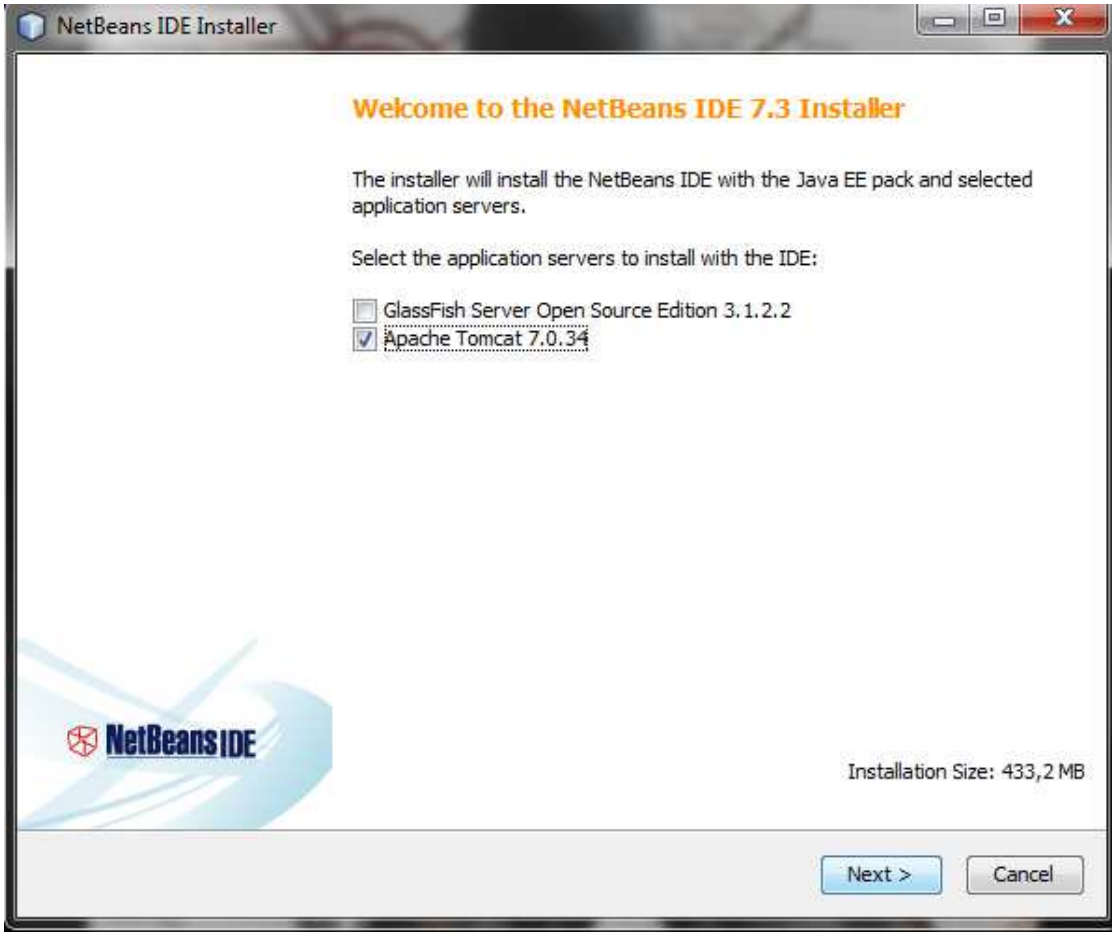


Εξ' ορισμού ο Apache λειτουργεί στην θύρα 80, που είναι η προεπιλεγμένη θύρα για τις σελίδες HTML, οπότε δεν χρειάζεται να προσθέσουμε την θύρα στη διεύθυνση.

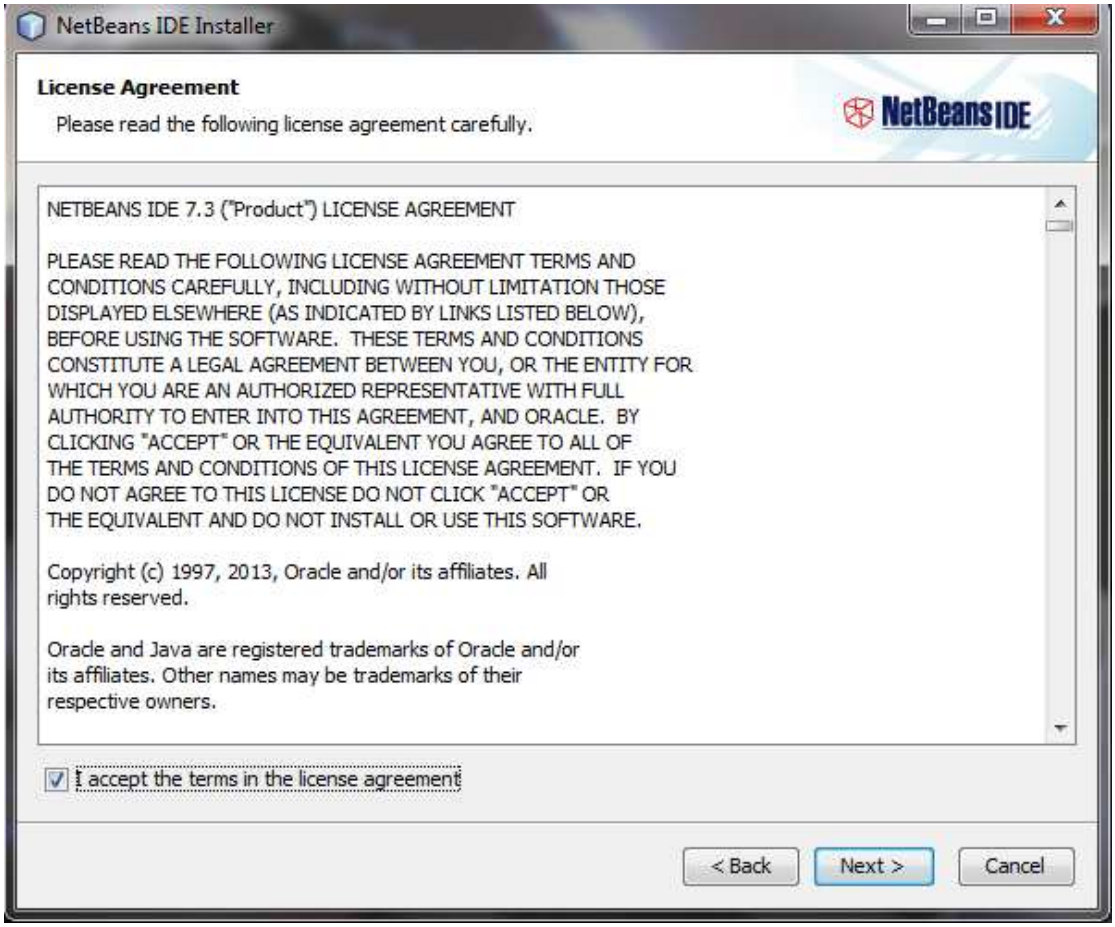
3.2 NetBeans

3.2.1 Εγκατάσταση προγραμμάτων.

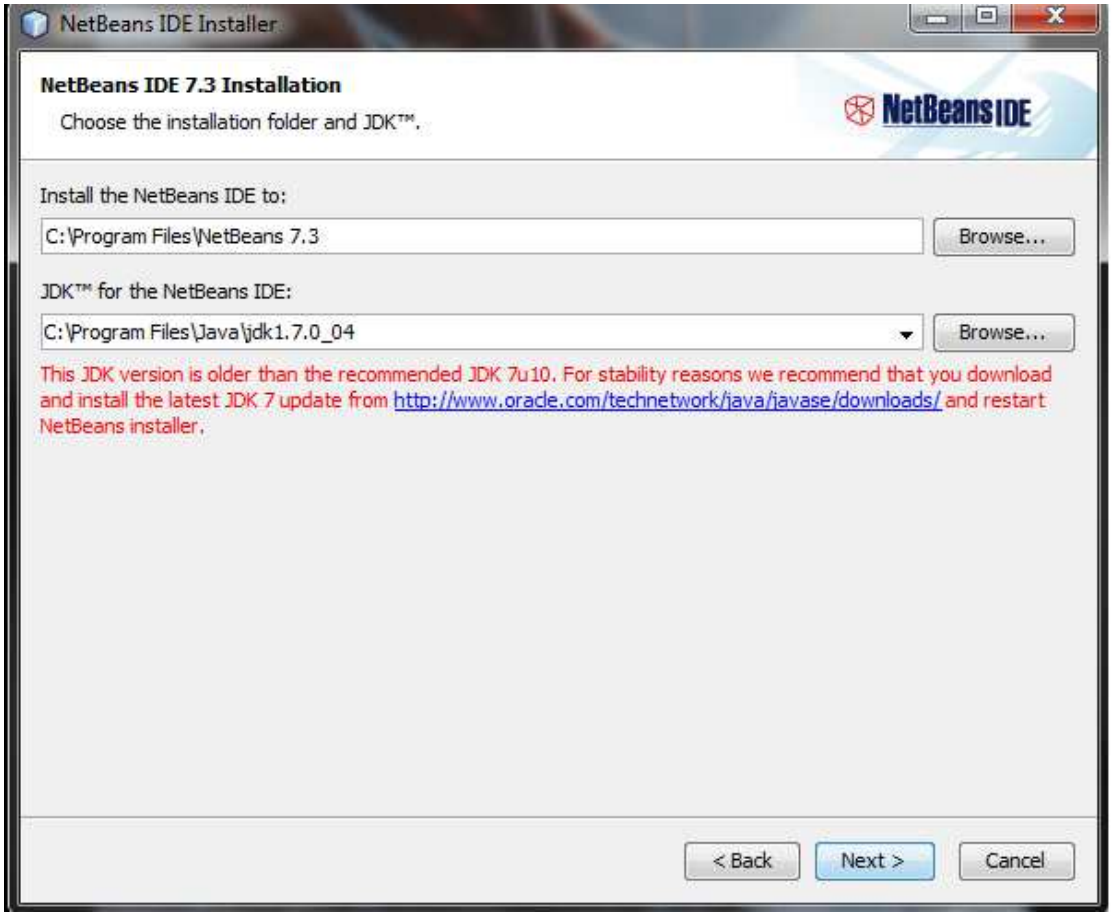
Για την εγκατάσταση των τεχνολογιών θα χρησιμοποιήσουμε το πρόγραμμα NetBeans.Ξεκινώντας χρειαζόμαστε το αρχείο netbeans-7.3-javaee-windows.exe για να αρχίσει η εγκατάσταση στη φάση αυτή επιλεγούμε τον Apache Tomcat και επιλεγούμε τον Glash Fish, μιας και είναι μια τεχνολογία που δεν μας χρησιμεύει στην παρούσα φάση, όπως φαίνεται παρακάτω.



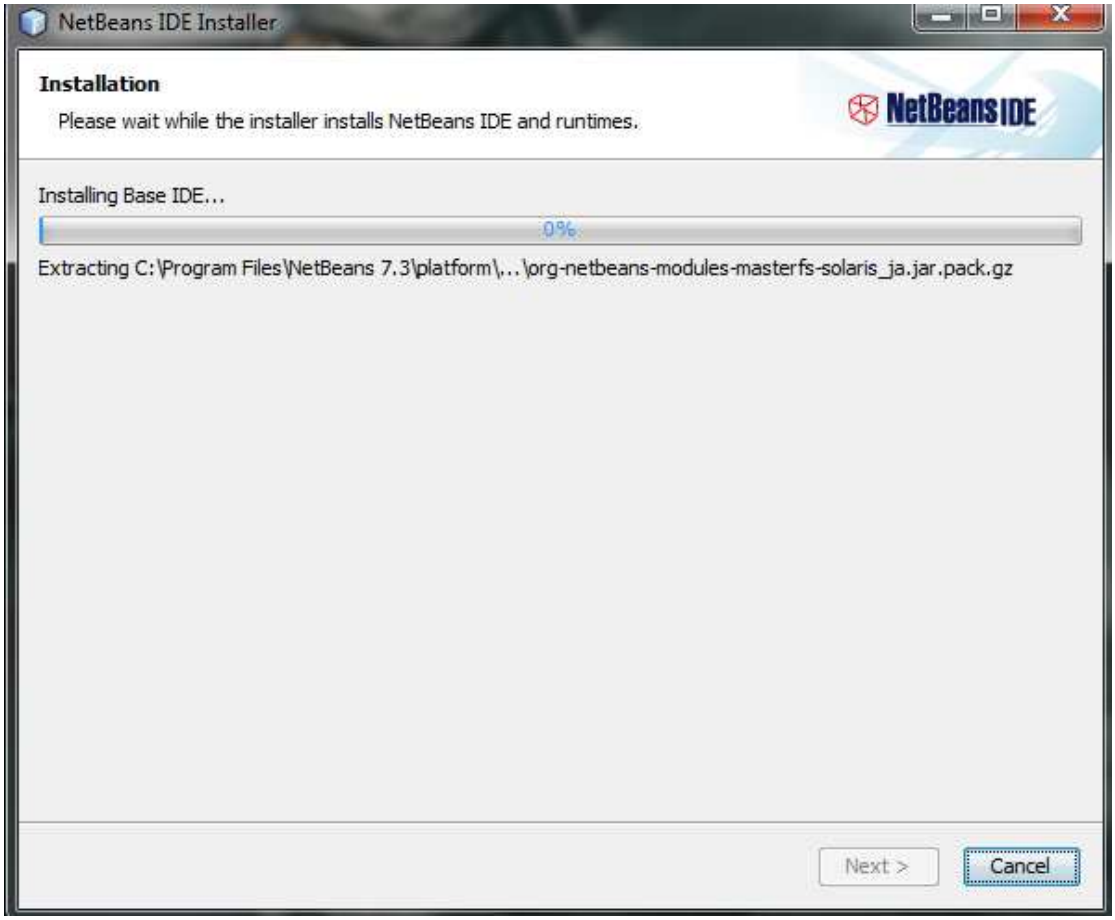
Στη συνεχεια πρεπει να διαβασουμε προσεκτικα και να αποδεκτουμε τους ορους της εγκαταστασης για να προσχορησει η διαδικασια .Παταμε NEXT για να συνεχισουμε.



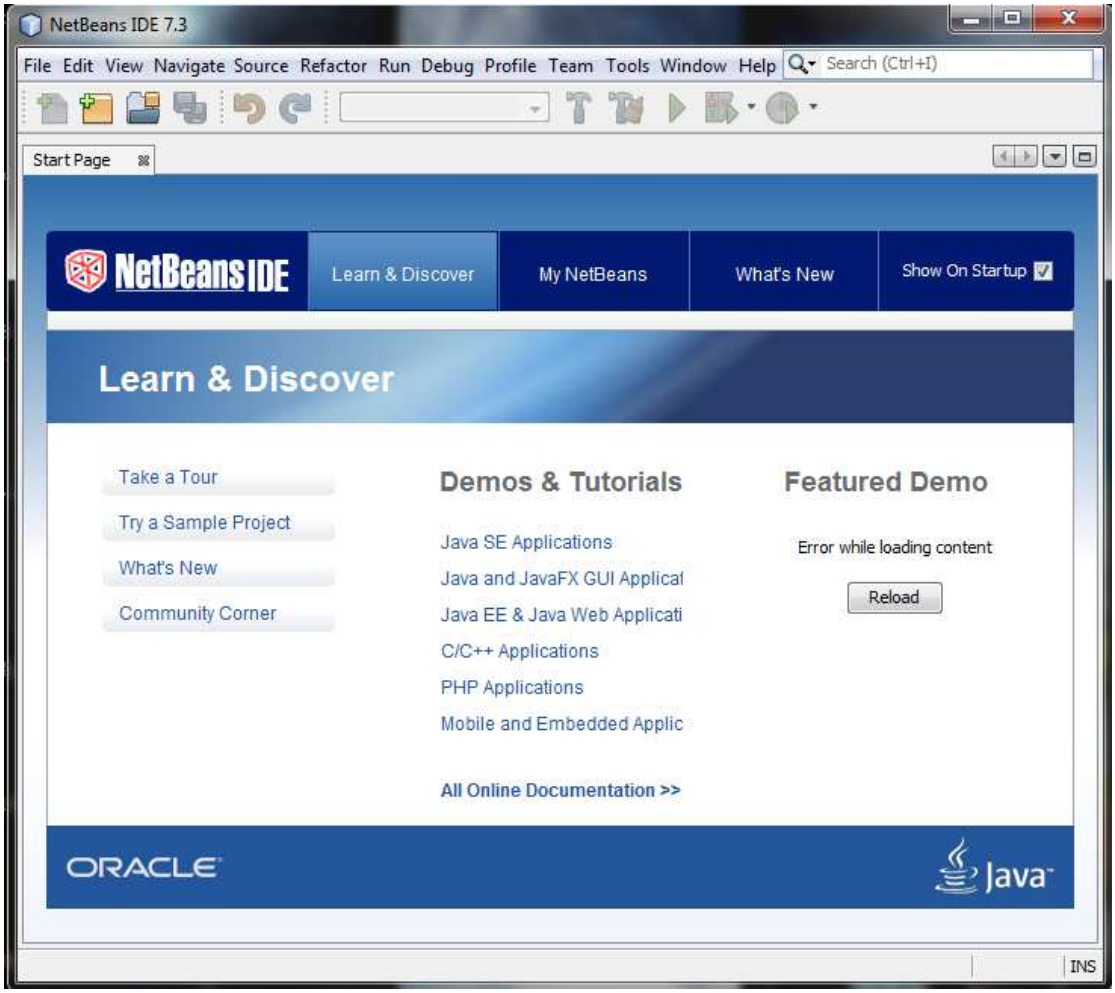
Προχωρώντας μας ζητείται ο φάκελος στον οποίο θα εγκατασταθεί το NetBeans καθώς και το JDK.Η προεπιλογή μας καλύπτει και δεν δημιουργεί προβλήματα τόσο στην εγκατάσταση όσο και στην λειτουργικότητα του προγράμματος. Παρατηρούμε ένα μήνυμα που μας ενημερώνει ότι το JDK που θα εγκατασταθεί είναι παλιότερο από το προαπαιτούμενο , το οποίο δεν αποτελεί πρόβλημα αφού εκτελώντας την λειτουργία " NetBeans update" θα ενημερωθεί αυτόματα. Παταμε NEXT για να συνεχισουμε.



Στη συνεχεία ξεκάνει η εγκατάσταση η όποια ολοκληρώνεται μόλις η διαδικασία φτάσει στο 100% όπως φαίνεται παρακάτω. Μπορούμε να ακυρώσουμε την εγκατάσταση πατώντας το κουμπί " cancel" όπως φαίνεται παρακάτω.



Μόλις ολοκληρωθεί η εγκατάσταση ανοίγει το κεντρικό παράθυρο του NetBeans όπως βλέπουμε παρακάτω.

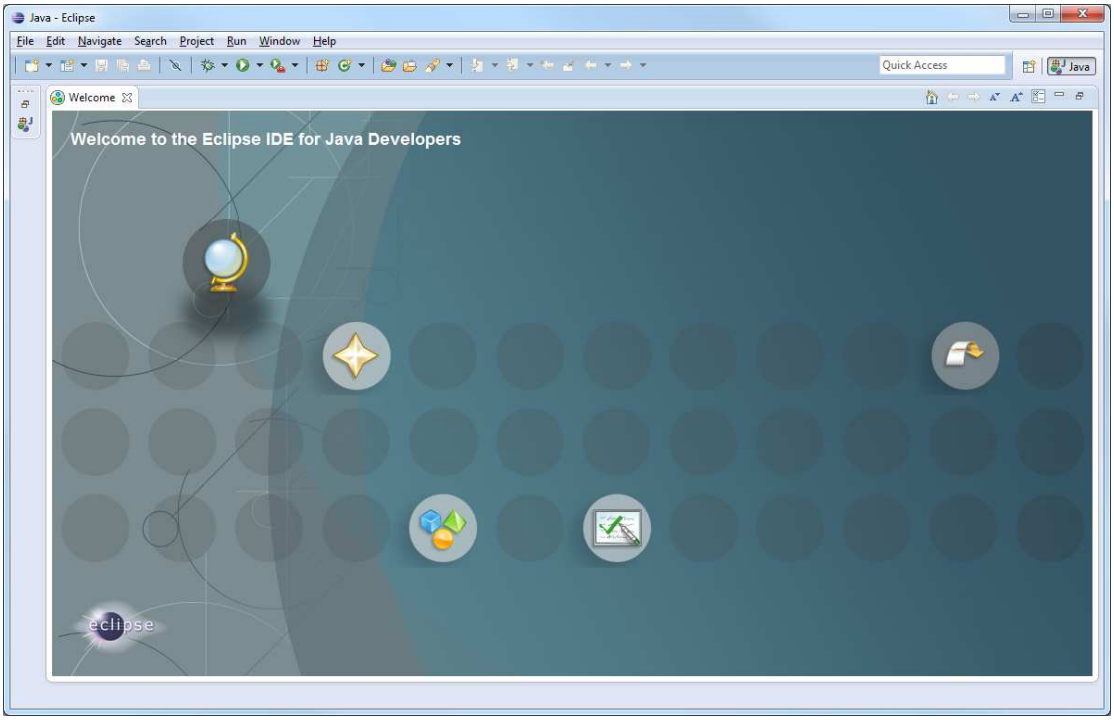


3.3 Android SDK

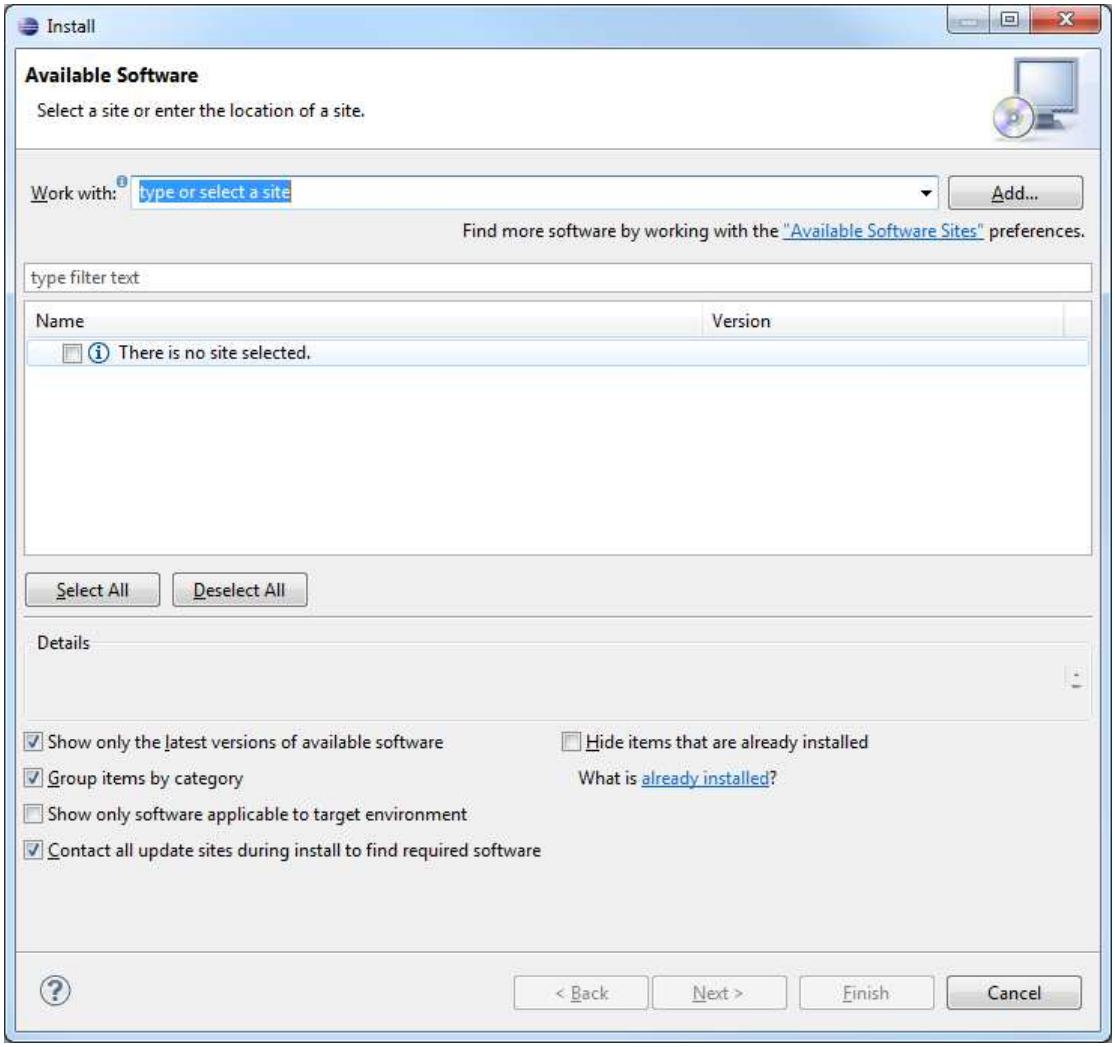
3.3.1 Εγκατάσταση προγραμμάτων.

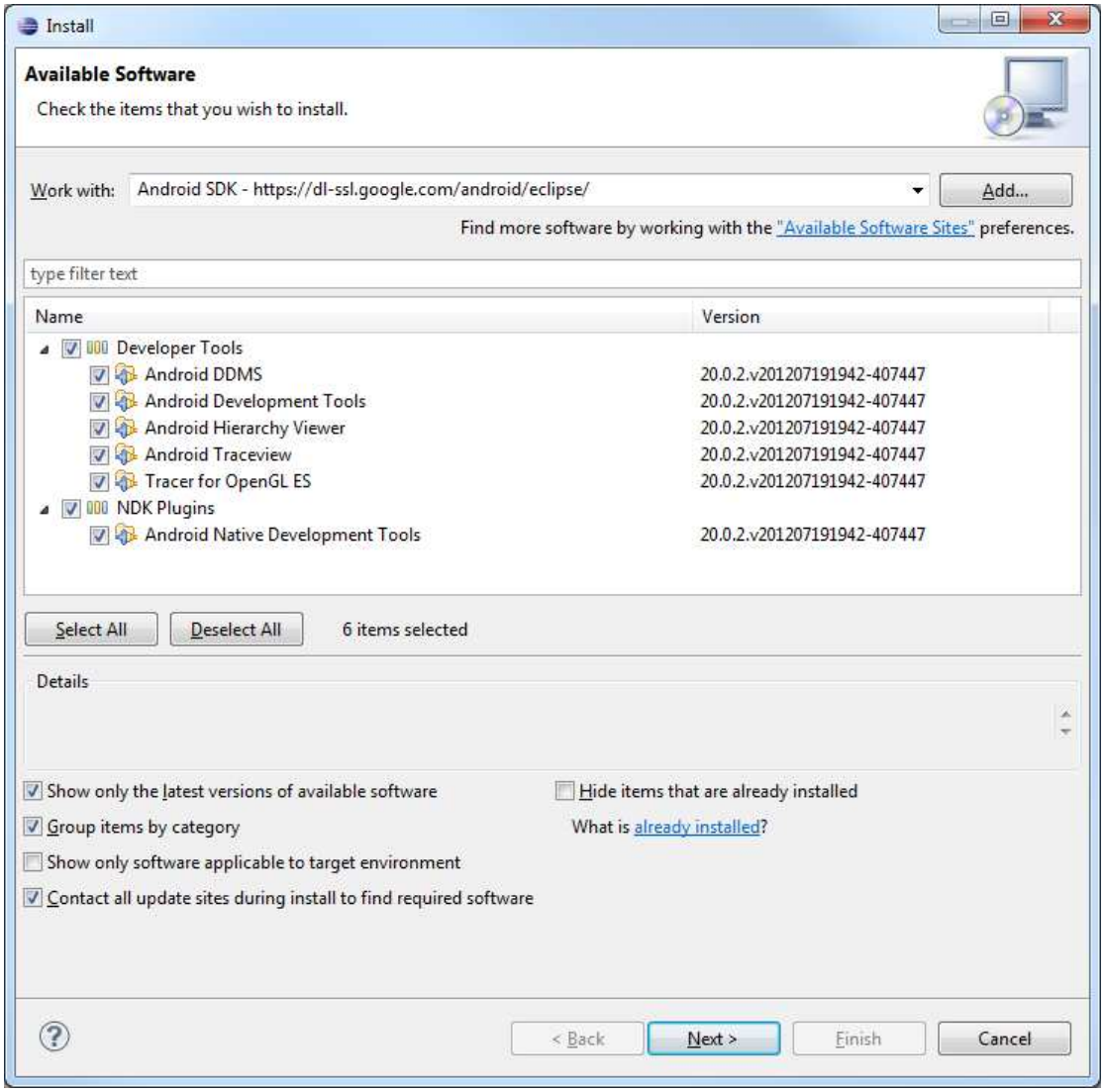
Αυτό το σεμινάριο είναι μέρος των Mobiletuts and session "Μάθετε Android SDK Development από το μηδέν!" Ο κύριος στόχος για αυτό το σεμινάριο είναι ο Developer του project Java να ξεκινήσει την εκμάθηση Android που έχει ήδη εγκαταστήσει το JDK και έχει ήδη εξοικειωθεί λίγο με το Eclipse. Παρακάτω βλέπουμε την αρχική οθόνη εγκατάστασης.

B

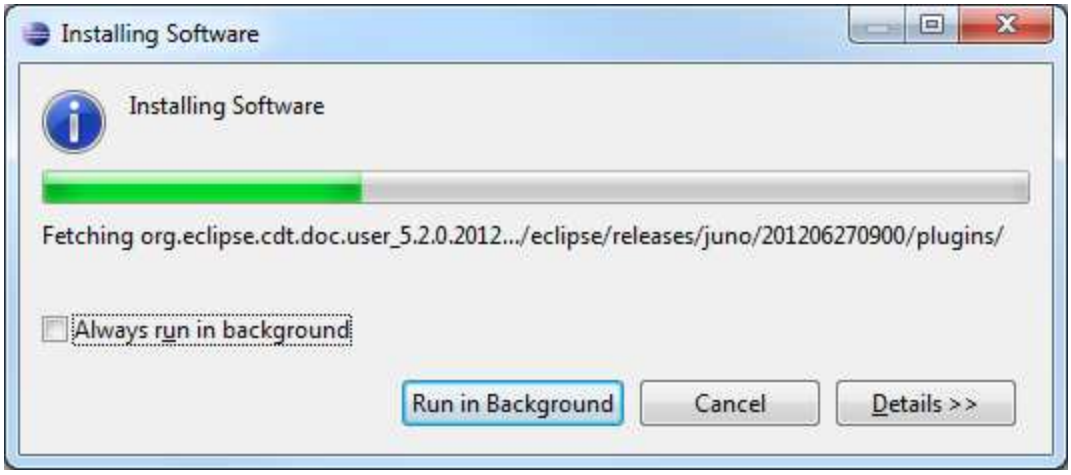


Πατάμε Help και επίτα , install new software.Έτσι εμφανίζεται το παρακάτω παράθυρο.


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
There is no site selected.
Select All
Deselect All
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

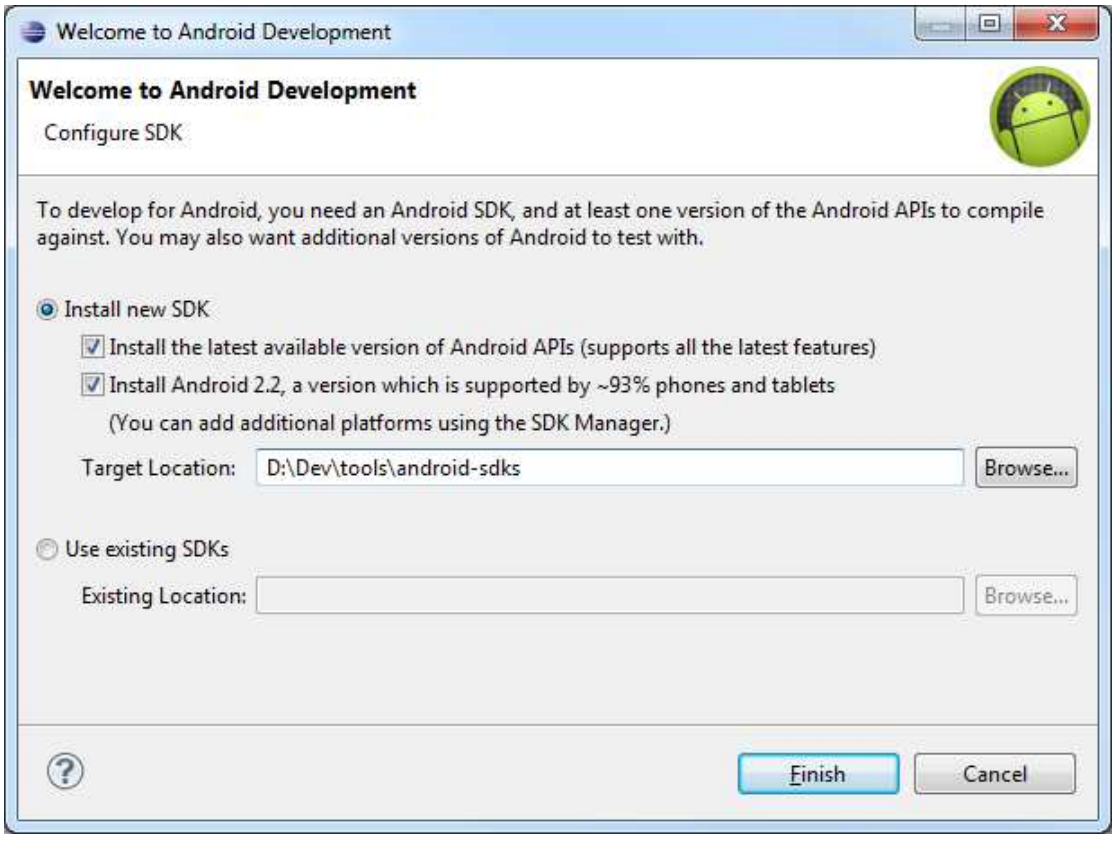


Πατάμε το κουμπί Next όπου εμφανίζεται μια οθόνη που επιβεβαιώνει τα εργαλεία που θέλουμε να εγκαταστήσουμε.



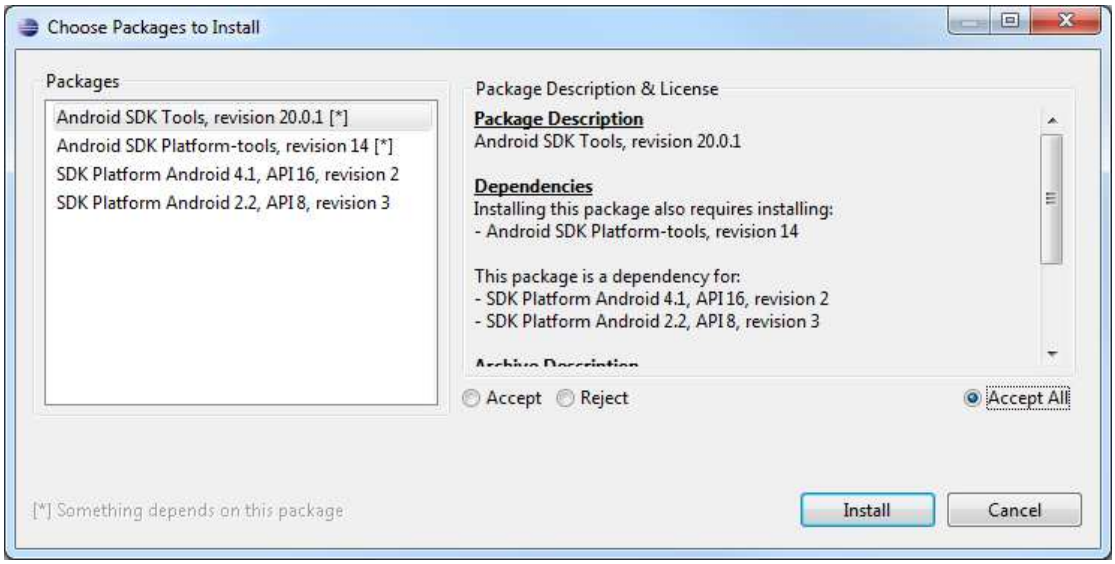
Σε αυτό το σημείο θα αρχίσει η λήψη του αρχείου Android Tools.Η ταχύτητα της λήψης εξαρτάται από την ταχύτητα της σύνδεσης στο internet.

Μόλις το Eclipse ξανανοίξει εμφανίζεται το παρακάτω παράθυρο με ένα μήνυμα καλωσορίσματος και εγκατάστασης του Android SDK.



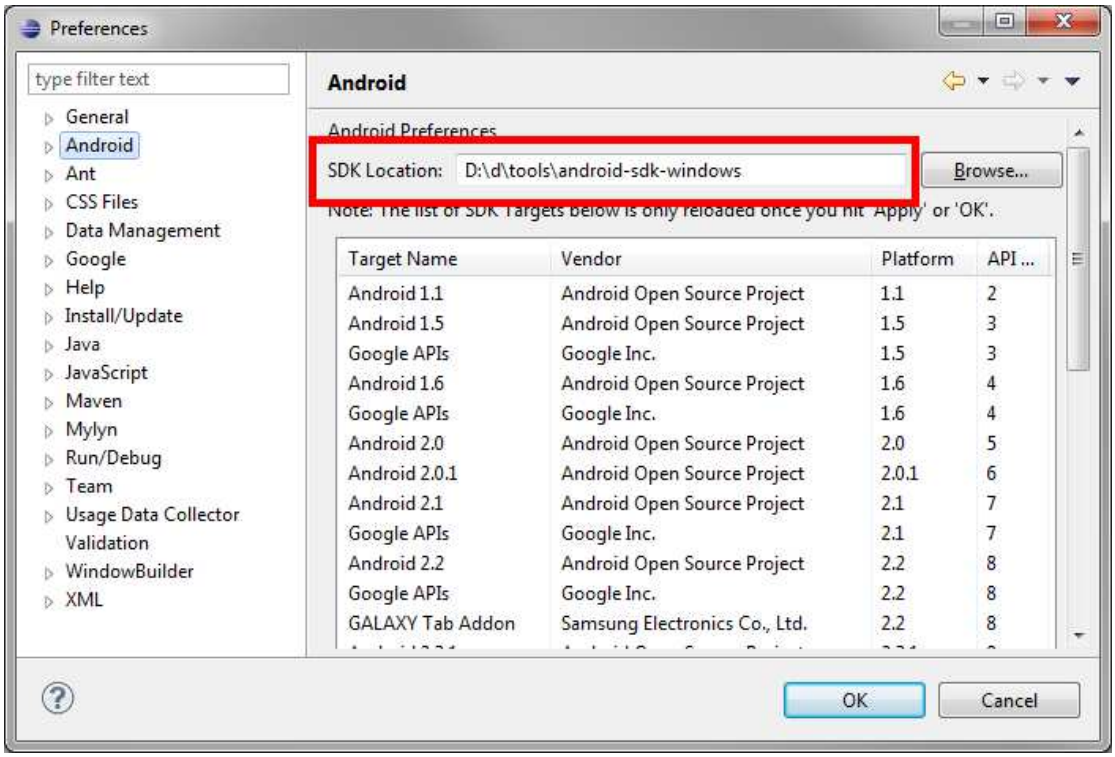
Εάν έχετε εγκαταστήσει το Android SDK στο παρελθόν, αυτό το βήμα μπορεί να είναι μια έκπληξη. Ωστόσο, η ομάδα του Android έχει απλοποιήσει σε μεγάλο βαθμό την εγκατάσταση του SDK μέσω του ADT εργαλείου στη Eclipse. Συνίσταται να επιλέξουμε να εγκαταστήσουμε το τελευταίο SDK και το 2.2 SDK. Μπορούμε να επιλέξουμε πού θέλουμε να εγκαταστήσουμε στα SDK, αν και η προεπιλεγμένη θέση συνήθως λειτουργεί μια χαρά.

Στο παρακάτω βήμα επιλεγούμε Accept all και πατάμε OK.



Ενημερώσεις.

Το Android SDK περιλαμβάνει πολλά εργαλεία της γραμμής εντολών. Είναι πιο εύκολο να τρέξει, αν έχουμε τους καταλόγους με αυτά τα εργαλεία σε αυτά μέσα τα Path . Στα Linux και Mac, αυτό συνήθως περιλαμβάνει την επεξεργασία. Προφίλ η στο αρχείο Bashrc. Στα Windows, μπορούμε να επεξεργαστούμε τη μεταβλητή περιβάλλοντος PATH. Θα χρειαστεί να προσθέσουμε τις διαδρομές τόσο για τα εργαλεία όσο και για την πλατφόρμα εργαλείων καταλόγων, όπου και αν βρίσκονται στο σύστημά μας. Και οι δύο είναι στο ανώτερο επίπεδο του καταλόγου SDK.



Η μηχανή ανάπτυξης είναι τώρα έτοιμη να ξεκινήσει.

3.3.2 Εγκατάσταση plug-in

http://mobile.tutsplus.com/tutorials/android/android-sdk-installation/

3.3.3 Δημιουργία project (χωρίς το command line)

http://developer.android.com/training/basics/firstapp/creating-project.html

Επιλεγούμε το New στην γραμμή εργαλείων.

Στο παράθυρο που εμφανίζεται ανοίγουμε τον φάκελο Android επιλεγούμε Android Application Project και πατάμε Next.

Πρέπει να συμπληρώσουμε την φόρμα που εμφανίζεται.

Το Application name είναι το όνομα της εφαρμογής που θα δημιουργήσουμε.

Το Project name είναι το όνομα του project directory μας που είναι ορατό στον Eclipse.

Το Package Name είναι το όνομα του πακέτου που θα χρησιμοποιήσουμε. Το όνομα του πακέτου μας πρέπει να είναι μοναδικό σε όλα τα πακέτα που είναι εγκατεστημένα στο σύστημα Android. Για το λόγο αυτό, είναι γενικά καλύτερα αν χρησιμοποιούμε ένα όνομα που ξεκινά με το αντίστροφη όνομα τομέα του οργανισμού μας ή της οντότητας του εκδότη . Για το έργο αυτό, μπορούμε να χρησιμοποιήσουμε κάτι σαν "com.example.myfirstapp." Ωστόσο, δεν μπορούμε να δημοσιεύσουμε app σας στο Google Play χρησιμοποιώντας το "com.example" namespace.

Το Minimum Required SDK είναι η χαμηλότερη έκδοση του Android που υποστηρίζει η εφαρμογή μας, δηλώνεται με το επίπεδο API. Για να υποστηρίξει όσες συσκευές όσο το δυνατόν, θα πρέπει να ρυθμίσουμε αυτή με τη χαμηλότερη διαθέσιμη έκδοση που επιτρέπει το app μας για την παροχή του βασικού συνόλου των χαρακτηριστικών. Αν κάποιο χαρακτηριστικό του app μας είναι δυνατό μόνο σε νεότερες εκδόσεις του Android και δεν είναι κρίσιμο, μπορούμε να ενεργοποιήσουμε τη λειτουργία μόνο όταν τρέχουμε τις εκδόσεις που το υποστηρίζουν .

Το Target SDK δηλώνει η υψηλότερη έκδοση του Android (επίσης με το επίπεδο API), με την οποία έχει δοκιμαστεί η εφαρμογή μας.

Compile With είναι η έκδοση της πλατφόρμας κατά την οποία θα γράψουμε το application μας. Από προεπιλογή, αυτό ορίζεται στην τελευταία έκδοση του Android SDK που είναι διαθέσιμο σε μας. Μπορούμε να χτίσουμε ακόμα το application μας για να υποστηρίξει παλαιότερες εκδόσεις, αλλά και τον καθορισμό του στόχου κατασκευής με την τελευταία έκδοση που μας επιτρέπει να ενεργοποιήσουμε νέα χαρακτηριστικά και τη βελτιστοποίηση application μας .

Το Theme καθορίζει το Android UI στυλ που θα εφαρμόσουμε για την εφαρμογή μας. Μπορούμε να αφήσουμε αυτό από μόνο του. Στη φάση αυτή πατάμε επόμενο.

Στην επόμενη οθόνη για να ρυθμίσουμε τις παραμέτρους του έργου, αφήνουμε τις προεπιλεγμένες ρυθμίσεις και κάνουμε κλικ στο Next.

Η επόμενη οθόνη μπορεί να μας βοηθήσει να δημιουργήσουμε ένα εικονίδιο εκκίνησης για την εφαρμογή μας.

Μπορούμε να τροποποιήσουμε ένα εικονίδιο με διάφορους τρόπους και το εργαλείο παράγει μια εικόνα για όλες τις πυκνότητες οθόνης. Πριν να δημοσιεύσουμε την εφαρμογή μας, θα πρέπει να βεβαιωθούμε ότι η εικόνα μας πληροί τις προδιαγραφές που ορίζονται στον οδηγό σχεδιασμού εικονογραφία.

Κάνουμε κλικ στο Επόμενο.

Τώρα μπορούμε να επιλέξετε ένα πρότυπο δραστηριότητας από την οποία να αρχίσει η κατασκευή app μας.

Για το έργο αυτό, επιλέγουμε BlankActivity και κάνουμε κλικ στο Next.

Αφήνουμε όλες τις λεπτομέρειες για τη δραστηριότητα στην προεπιλεγμένη κατάστασή τους και κάνουμε κλικ στο Finish.

Το Project μας έχει συσταθεί με κάποια προεπιλεγμένα αρχεία και είμαστε έτοιμοι να αρχίσει η κατασκευή του app μας.

3.3.4 Εκτέλεση project (Run on a Real Device και Run on the emulator)

http://developer.android.com/training/basics/firstapp/running-app.html

Τρέχοντας σε μια πραγματική συσκευή.

Εφόσον έχουμε μια πραγματική Android-powered συσκευή, θα δούμε παρακάτω με ποιον τρόπο μπορούμε να τρέξουμε την εφαρμογή μας:

Συνδέουμε τη συσκευή μας στον υπολογιστή με ένα καλώδιο USB. Στο λειτουργικό Windows, ίσως χρειαστεί να εγκαταστήσουμε το κατάλληλο πρόγραμμα οδήγησης USB για τη συσκευή μας. Για τους drivers βοήθεια για την εγκατάσταση, ανατρέχουμε στο OEM USB έγγραφο Drivers.

Ενεργοποίηση εντοπισμός σφαλμάτων USB στη συσκευή μας.

Στις περισσότερες συσκευές που τρέχουν Android 3.2 ή μεγαλύτερο, μπορούμε να βρούμε την επιλογή στις Ρυθμίσεις> Εφαρμογές> Ανάπτυξη.

Με το Android 4.0 και νεότερες, είναι στις Ρυθμίσεις> Επιλογές για προγραμματιστές.

Σημείωση: Με το Android 4.2 και νεότερες, Developer επιλογές είναι κρυφή από προεπιλογή. Για να είναι διαθέσιμη, μεταβαίνουμε στις Ρυθμίσεις> Σχετικά με το τηλέφωνο και πατάμε Build αριθμό επτά φορές. Επιστρέφουμε στην προηγούμενη οθόνη για να βρίσκουμε τις επιλογές Developer.

3.3.4.1 Για να εκτελέσουμε το app από το Eclipse:

Ανοίγουμε ένα από τα αρχεία του project μας και κάνουμε κλικ στο κουμπί Εκτέλεση από τη γραμμή εργαλείων.

Στο Εκτέλεση ως παράθυρο που εμφανίζεται, επιλέγουμε το Android Εφαρμογών και κάνουμε κλικ στο OK.

Το Eclipse εγκαθιστά την εφαρμογή στη συνδεδεμένη συσκευή και ξεκινάει .

3.3.4.2 Για να τρέξει το application μας από μια γραμμή εντολών:

Αλλάζουμε directories στο root του Android και εκτελούμε:

```
ant debug
```

Αφού σιγουρευτούμε ότι το Android SDK platform-tools ανήκει στο path μας εκτελούμε :

```
adb install bin/MyFirstApp-debug.apk
```

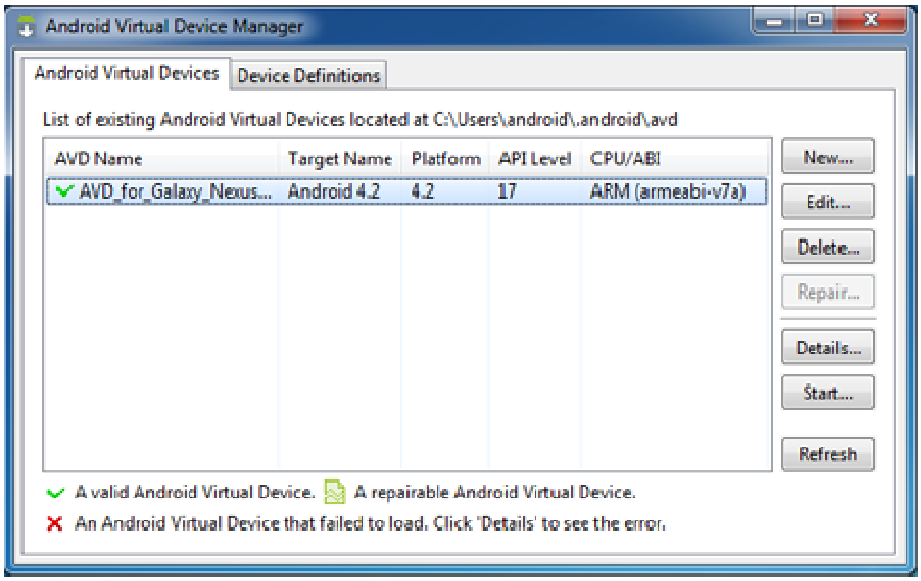
Στη συσκευή μας βρίσκουμε το MyFirstActivity και το ανοίγουμε.

Έτσι κατασκευάζουμε και τρέχουμε το Android Application μας σε μια πραγματική συσκευή!

3.3.5 Τρέχοντας στον Emulator.

Είτε χρησιμοποιούμε Eclipse ή τη γραμμή εντολών, για να τρέξει η εφαρμογή μας για τον εξομοιωτή θα πρέπει να δημιουργήσουμε πρώτα ένα Android Virtual Device (AVD). Ένα AVD είναι μια διαμόρφωση της συσκευής για τον εξομοιωτή Android που μας επιτρέπει να τρέξουμε σε διαφορετικές συσκευές.

Σχήμα 1. Το AVD Διευθυντής δείχνει μερικές εικονικές συσκευές.



3.3.5.1 Για να δημιουργήσουμε ένα AVD:

Εκκινούμε το Android Virtual Διαχείριση συσκευών:

Στο Eclipse, κάνουμε κλικ στο Android Virtual Device Manager από την γραμμή εργαλείων.

Από τη γραμμή εντολών, αλλάζουμε καταλόγους < sdk/tools/ και εκτελούμε :

```
android avd
```

στο Android Virtual Device Manager Panel κάνουμε κλικ στο New.

Γεμίζουμε τη φόρμα για το AVD και έπειτα πατάμε κλικ στο Create AVD.

Επιλέγουμε το νέο AVD από το Android Virtual Διαχείριση συσκευών και κάνουμε κλικ στο Έναρξη.

Μετά την εκκίνηση του εξομοιωτή , ξεκλειδώνουμε την οθόνη εξομοιωτή.

3.3.5.2 Για να εκτελέσουμε το app από το Eclipse

Ανοίγουμε ένα από τα αρχεία του project μας και κάνουμε κλικ στο κουμπί Εκτέλεση από τη γραμμή εργαλείων.

Στο Εκτέλεση ως παράθυρο που εμφανίζεται, επιλέγουμε το Android για Εφαρμογές και κάνουμε κλικ στο OK.

Το Eclipse εγκαθιστά το app για AVD και ξεκινάει.

3.3.5.3 Για να τρέξει το application μας από τη γραμμή εντολών

Αλλάζουμε τον κατάλογο στη ρίζα του Android έργου μας και εκτελούμε:

```
ant debug
```

```
1. adb install bin/MyFirstApp-debug.apk
```

Από τον εξομοιωτή, εντοπίζουμε το MyFirstActivity και το ανοίγουμε.

3.4 JSON

Ένα JSONObject είναι μια συλλογή ζευγών ονομάτων και τιμών χωρίς σειρά. Η εξωτερική μορφή της είναι μια σειρά από άγκιστρα με άνω και κάτω τελεία μεταξύ των ονομάτων και των τιμών, και κόμματα μεταξύ των τιμών και των ονομάτων. Η εσωτερική μορφή είναι ένα αντικείμενο που έχει τις μεθόδους για την πρόσβαση των τιμών με βάση το όνομα, και θέτει τις μεθόδους για την προσθήκη ή αντικατάσταση των τιμών με βάση το όνομα. Οι τιμές μπορούν να είναι οποιοσδήποτε από αυτούς τους τύπους: Boolean, JSONArray, JSONObject, Αριθμός, String, ή την JSONObject.NULL αντικείμενο. Ένας JSONObject constructor μπορεί να χρησιμοποιηθεί για να μετατρέψει μια εξωτερική μορφή κειμένου JSON σε μια εσωτερική μορφή των οποίων οι τιμές μπορούν να ανακτηθούν με την get και την opt μέθοδο, ή να μετατρέψει τις τιμές σε ένα κείμενο JSON χρησιμοποιώντας την put και την toString μέθοδο. Μια get μέθοδος επιστρέφει μια τιμή, αν μπορεί να βρεθεί μια τέτοια, και ρίχνει μια εξαίρεση, αν δεν μπορεί να βρεθεί. Μια μέθοδος opt επιστρέφει μια προκαθορισμένη τιμή(default value), αντί να κάνει throw μια εξαίρεση, και έτσι είναι χρήσιμη για την απόκτηση προαιρετικών τιμών.

4 Ανάλυση εφαρμογής

4.1 Γενικά

Σε αυτή την ενότητα θα αναλύσουμε τα κομμάτια τα εφαρμογής

4.2 Βάση δεδομένων

Η βάση δεδομένων χρησιμοποιεί δύο πίνακες, τους user και game.

Ο Πίνακας user περιέχει τις τιμές που πρέπει να καταχωρηθούν ώστε να υπάρχει η οντότητα user υπάρχουν διάφορα πεδία όπως το πεδίο id, που είναι το id του χρηστή αναπαρίσταται ως ακέραιος . Το πεδίο name που είναι το όνομα του χρηστή και το όποιο αναπαριστάται ως χαρακτήρας μεγάλου μήκους, το πεδίο password είναι ο κωδικός του χρηστή όπως τον ορίζει ο ίδιος αναπαρίσταται και αυτό ως χαρακτήρας μεγάλου μήκους. Έπειτα τα πεδία wins και losses τα όποια είναι ακέραιοι, αντιπροσωπεύουν τις νίκες και τις ήττες του κάθε παίχτη αντίστοιχα .Τέλος το πεδίο image είναι τύπου συμβολοσειρά με αυτό το πεδίο απεικονίζουμε την εικόνα με την όποια απεικονίζεται-αναπαριστάται ο κάθε χρήστης.

#	Name	Type
1	id	int(11)
2	name	varchar(20)
3	password	varchar(20)
4	wins	int(11)
5	losses	int(11)
6	image	varchar(20)

Ο Πίνακας game περιέχει τις τιμές που πρέπει να καταχωρηθούν ώστε να υπάρχει η οντότητα game. . Βλέπουμε και εδώ το πεδίο id το όποιο είναι τύπου ακέραιος μας δίνει το id του εκάστου παιχνιδιού. Το πεδίο user1_id μας δίνει το id του πρώτου χρηστή και το user2_id μας δίνει το id του δευτέρου χρηστή είναι και τα δυο τύπου ακέραιος. Επίσης το πεδίο last_move στο όποιο καταχωρείται η τελευταία κίνηση που έγινε στο παιχνίδι. Παρατηρούμε στο σημείο αυτό ότι το πεδίο lastMove είναι τύπου Date άρα η καταχώρηση στην όποια προαναφερθήκαμε γίνεται σε μορφή ημερομηνίας και ώρας. Η

μεταβλητή board είναι τύπου αλφαριθμητικό και αναπαριστά με αυτόν τον τρόπο το παιχνίδι.

	#	Name	Type
☐	1	id	int(11)
☐	2	user1_id	int(11)
☐	3	user2_id	int(11)
☐	4	last_move	datetime
☐	5	board	varchar(20)
☐	6	turn	int(11)

4.3 Ανάλυση server

4.3.1.1 User

```
public class User {
    private int id;
    private String name;
    private String password;
    private int wins;
    private int losses;
    private String image;
```

Στην κλάση User υλοποιείται ο παίχτης user ο όποιος αντιπροσωπεύει το φυσικό πρόσωπο που χρησιμοποιεί την εφαρμογή. Παραπάνω βλέπουμε τα πεδία που υλοποιούνται στον user ,τα όποια διαμορφώνονται ως εξής. Το πεδίο id είναι το id του χρηστή το όποιο κάθε φορά είναι μοναδικό δεν μπορεί να υπάρξει δυο φορές αναπαρίσταται ως ακέραιος. Το πεδίο name που είναι το όνομα του χρηστή και το όποιο αναπαριστάται ως συμβολοσειρά ,το πεδίο password είναι ο κωδικός του χρηστή όπως τον ορίζει ο ίδιος αναπαρίσταται και αυτό ως συμβολοσειρά. Έπειτα τα πεδία wins και losses τα όποια είναι ακέραιοι, αντιπροσωπεύουν τις νίκες και τις ήττες του κάθε παίχτη αντίστοιχα .Τέλος το πεδίο image είναι τύπου συμβολοσειρά με αυτό το πεδίο απεικονίζουμε την εικόνα με την όποια απεικονίζεται-αναπαριστάται ο κάθε χρήστης.

4.3.1.2 Game

```
public class Game {
    private int id;
    private int user1Id;
    private int user2Id;
    private Date lastMove;
    private String board = "---------";
    private int turn = 1;
```

Στην κλάση Game υλοποιείται το παιχνίδι , το όποιο έχει κάποια δικά του χαρακτηριστικά. Βλέπουμε και εδώ το πεδίο id το όποιο είναι τύπου ακέραιος μας δίνει το id του εκάστου παιχνιδιού και είναι και αυτό μοναδικό για κάθε παιχνίδι . Το πεδίο user1Id μας δίνει το id του πρώτου χρηστή και το user2Id μας δίνει το id του δευτέρου χρηστή είναι και τα δυο τύπου ακέραιος. Επίσης το πεδίο lastMove στο όποιο καταχωρείται η τελευταία κίνηση που έγινε στο παιχνίδι. Παρατηρούμε στο σημείο αυτό ότι το πεδίο lastMove είναι τύπου Date άρα η καταχώρηση στην όποια προαναφερθήκαμε γίνεται σε μορφή ημερομηνίας και ώρας. Η μεταβλητή board είναι τύπου συμβολοσειρά και αναπαριστά με αυτόν τον τρόπο το παιχνίδι με ‘’9 παύλες’. Σε κάθε μια από αυτές αποθηκεύεται σε μορφή συμβολοσειράς η κίνηση του κάθε παίχτη.

```
- - -    x o x

- - -    o x x

- - -    x o o
```

Το όποιο τελικά θα αποθηκευτεί σε μορφή συμβολοσειράς στην μεταβλητή board ως εξής : xoxoxxxoo .Τέλος η μεταβλητή turn η όποια είναι τύπου int και με βάση αυτήν διαμορφώνεται η σειρά που παίζουν οι παίχτες.

4.3.1.3 DataConnection

```
public class DataConnection {

    public static final String DB_URL = "jdbc:mysql://127.0.0.1/";
    public static final String DB_NAME = "tictactoe";
    public static final String DB_USER = "root";
    public static final String DB_PASSWORD = "";

    private Connection con;

    public DataConnection() {
        try {
            Class.forName( "com.mysql.jdbc.Driver" );
            con = DriverManager.getConnection( DB_URL + DB_NAME, DB_USER, DB_PASSWORD );

        } catch ( SQLException ex ) {
            ex.printStackTrace();
        } catch ( ClassNotFoundException ex ) {
            ex.printStackTrace();
        }
    }

    public void close() {
        try {
            if ( con != null ) {
                con.close();
            }

        } catch ( SQLException ex ) {
            ex.printStackTrace();
        }
    }
```

Στην κλάση DataConnection γίνεται η σύνδεση με το server. Αρχικά όπως παρατηρούμε συνδεόμαστε με τη βάση δεδομένων , χρησιμοποιούμε μια μεταβλητή με τον παραπάνω κώδικα προσπαθούμε να κάνουμε την σύνδεση χρησιμοποιώντας κάποια exceptions , μόλις συνδεθούμε υπάρχει η συνάρτηση close που χρησιμοποιείται για να κλείσει η σύνεση. Αν υπάρχει σύνδεση τότε κλείνει.

Παρακάτω βλέπουμε τη συνάρτηση get User η όποια επιλεγεί από τη βάση δεδομένων μας από τον πινάκα user με το όνομα που θέλουμε και το password . Παρακάτω δημιουργούμε ένα νέο χρηστή με το id, το name, το Password, το Wins, το Losses και το Image με την εντολή get, που αντλεί τα δεδομένα από την κλάση User .

```
public User getUser( String name, String password ) {
    try {
        String q = "SELECT * FROM `user` WHERE `name` = '" + name + "' AND `password` = '" + password + "'";
        Statement stm = con.createStatement( ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE );

        ResultSet rs = stm.executeQuery( q );

        if ( rs.next() ) {
            User user = new User();

            user.setId( rs.getInt( "id" ) );
            user.setName( rs.getString( "name" ) );
            user.setPassword( rs.getString( "password" ) );
            user.setWins( rs.getInt( "wins" ) );
            user.setLosses( rs.getInt( "losses" ) );
            user.setImage( rs.getString( "image" ) );

            return user;
        }

    } catch ( SQLException ex ) {
        ex.printStackTrace();
    }
    return null;
}
```

Όπως βλέπουμε παρακάτω βρίσκουμε τον user με βάση το id πεδίο του.

```
public User getUserById( int id ) {
    try {
        String q = "SELECT * FROM `user` WHERE `id` = " + id;
        Statement stm = con.createStatement( ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE );

        ResultSet rs = stm.executeQuery( q );

        if ( rs.next() ) {
            User user = new User();

            user.setId( rs.getInt( "id" ) );
            user.setName( rs.getString( "name" ) );
            user.setPassword( rs.getString( "password" ) );
            user.setWins( rs.getInt( "wins" ) );
            user.setLosses( rs.getInt( "losses" ) );
            user.setImage( rs.getString( "image" ) );

            return user;
        }

    } catch ( SQLException ex ) {
        ex.printStackTrace();
    }
    return null;
}
```

Η διαδικασία είναι παρόμοιε με την αντίστοιχη του get User όπως την περιγράψαμε παραπάνω.

Στην παρακάτω εικόνα βλέπουμε την διαδικασία για το registration του user .Τα πεδία του user γίνονται update με τις εντολές όπως φαίνονται παρακάτω.

```
public void registerUser( User user ) {
    try {
        String q = "SELECT * FROM `user` WHERE 1 = 0";
        Statement stm = con.createStatement( ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE );

        ResultSet rs = stm.executeQuery( q );

        rs.moveToInsertRow();

        rs.updateString( "name", user.getName() );
        rs.updateString( "password", user.getPassword() );
        rs.updateString( "image", user.getImage() );
        rs.updateInt( "wins", user.getWins() );
        rs.updateInt( "losses", user.getLosses() );

        rs.insertRow();

    } catch ( SQLException ex ) {
        ex.printStackTrace();
    }
}
```

Στο σημείο αυτό διαγράφουμε τις έγγραφες οι οποίες είναι παλαιότερες των 5 λεπτών με βάση την ημερομηνία και την ώρα όπως φαίνεται παρακάτω .

Στη συνεχεία επιλεγούμε τα παιχνίδια που περιμένουν δεύτερο παιχτεί να συνδεθεί , αν ένα τέτοιο παιχνίδι υπάρχει τότε φορτώνουμε ένα τέτοιο και επιστρέφουμε. Αν όχι τότε δημιουργούμε ένα τέτοιο παιχνίδι και επιστρέφω περιμένοντας για σύνδεση από άλλο παίχτη. Τέλος ξαναδιαλεγω το παιχνίδι που έσωσα για να πάρω το id του και να συνδεθώ σε αυτό

Στο σημείο αυτό διαγράφουμε τις έγγραφες οι οποίες είναι παλιότερες από των 5 λεπτών .

```
// Διαγραφή εγγραφών παλιότερων από 5 λεπτά
String q = "DELETE FROM game WHERE DATEDIFF( minute, getdate(), last_move) < -5";
Statement stm = con.createStatement( ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE );
stm.executeUpdate( q );
```

Επιλέγουμε τα παιχνίδια που περιμένουν για τον δεύτερο παίχτη όπως φαίνεται παρακάτω.

```
// Επιλογή παιχνιδιών που περιμένουν 2ο παίκτη
q = "SELECT * FROM game WHERE user2Id = -1";
ResultSet rs = stm.executeQuery( q );
```

Εάν υπάρχει τέτοιο παιχνίδι δυο ατόμων το φορτώνω και το επιστρέφω.

```
if ( rs.next() ) { // Αν υπάρχει ένα τέτοιο παιχνίδι το φορτώνω και το επιστρέφω
    Game game = new Game();

    game.setId( rs.getInt( "id" ) );
    game.setUser1Id( rs.getInt( "user1_id" ) );
    game.setLastMove( rs.getDate( "last_move" ) );
```

Εάν δεν υπάρχει ένα τέτοιο παιχνίδι δημιουργώ ένα νέο και το επιστρέφω.

```
} else { // Αν δεν υπάρχει τέτοιο παιχνίδι δημιουργώ ένα νέο και το επιστρέφω
    Game game = new Game();

    game.setLastMove( new Date() );
    game.setUser1Id( userId );

    rs.moveToInsertRow();

    rs.updateInt( "user1_id", game.getUser1Id() );
    rs.updateInt( "user2_id", -1 );
    rs.updateString( "board", game.getBoard() );
    rs.updateDate( "last_move", new java.sql.Date( game.getLastMove().getTime() ) );

    rs.insertRow();
```

Ξαναδιαλεγω το παιχνίδι που έδωσα για να πάρω το id του .

```
        // ξαναδιαλέγω το παιχνίδι που έσωσα για να πάρω το id του
        q = "SELECT * FROM game WHERE user2Id = -1";
        rs = stm.executeQuery( q );

        rs.next();
        game.setId( rs.getInt( "id") );

        return game;
    }
```

Παίρνω το παιχνίδι που ζητήθηκε

```
blic Game getGameById( int gameId ) {
  try {

    // Παίρνω το παιχνίδι που ζητήθηκε
    String q = "SELECT * FROM game WHERE id = " + gameId;
    Statement stm = con.createStatement( ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE );
    ResultSet rs = stm.executeQuery( q );
```

Εάν υπάρχει ένα τέτοιο και εφόσον δεν έχουν περάσει 5 λεπτά το φορτώνω και το επιστέφω.

```
if ( rs.next() ) { // Αν υπάρχει (δεν έχουν περάσει 5 λεπτά) το φορτώνω και το επιστρέφω
    Game game = new Game();

    game.setId( rs.getInt( "id" ) );
    game.setUser1Id( rs.getInt( "user1_id" ) );
    game.setUser2Id( rs.getInt( "user2_id" ) );
    game.setLastMove( rs.getDate( "last_move" ) );
    game.setBoard( rs.getString( "board" ) );
    game.setTurn( rs.getInt( "turn" ) );

    return game;
}
```

4.3.1.4 Login (Servlet)

Παρακάτω βλέπουμε την κλάση Login.

Αρχικά παίρνουμε τα στοιχεία που έχει στείλει ο χρήστης, δηλαδή το όνομα και τον κωδικό.

```
// Παίρνω τα στοιχεία από τη φόρμα
String name = request.getParameter( "name" );
String password = request.getParameter( "password" );
```

Αρχικά κάνουμε μια νέα σύνδεση και με την μέθοδο get User παίρνουμε τον χρήστη που έχει τέτοια στοιχεία. Στη συνέχεια κλείνουμε την σύνδεση με τη βάση.

```
DataConnection connection = new DataConnection(); // σύνδεση στη βάση
User user = connection.getUser( name, password ); // παίρνω τον χρήστη

connection.close(); // αποσύνδεση από τη βάση
```

Στη συνέχεια δημιουργούμε την απάντηση. Υπάρχουν δύο περιπτώσεις, να είναι σωστά τα στοιχεία, οπότε η user περιέχει αντικείμενο ή να είναι null, αν είναι λάθος τα στοιχεία.

Αν ο χρήστης αυτός υπάρχει τότε δημιουργούμε ένα JSONObject .Αν τα στοιχειά είναι σωστά τότε εμφανίζεται μήνυμα “ok” και “Log in successful” και τυπώνονται τα τοιχία.

```
// Δημιουργώ την απάντηση
if ( user != null ) { // Αν τα στοιχεία είναι έγκ
    JSONObject resp = new JSONObject();
    resp.put( "status", "OK" );
    resp.put( "message", "Log in succesfull." );

    resp.put( "name", name );
    resp.put( "id", user.getId() );
    resp.put( "wins", user.getWins() );
    resp.put( "losses", user.getLosses() );
    resp.put( "image", user.getImage() );

    resp.writeJSONString( out );
```

Αν τα τοιχία που δόθηκαν είναι λάθος τότε εμφανίζεται μήνυμα λάθους " Wrong name or password."

```
} else { // Αν έδωσε λάθος στοιχεία (δεν υπάρχει τέτοι
    JSONObject resp = new JSONObject();
    resp.put( "status", "ERROR" );
    resp.put( "message", "Wrong name or password." );

    resp.writeJSONString( out );
}
```

4.3.1.5 Register (Servlet)

Στη συνεχεία βλέπουμε την κλάση Rgister.H κλάση αυτή καλείται κατά τη δημιουργία του χρηστή για να γίνει το registration του. Παίρνει τα στοιχειά και δημιουργεί το νέο χρηστή.

```
@WebServlet( name = "Register", urlPatterns = { "/Register" } )
public class Register extends HttpServlet {
```

Στο σημείο αυτό δίνεται η δυνατότητα στο νέο χρηστή να εισάγει όνομα κωδικό και να επιλέξει την εικόνα για τον χρηστή που θέλει να δημιουργήσει.

```
// Παίρνω τα στοιχεία
String name = request.getParameter( "name" );
String password = request.getParameter( "password" );
String image = request.getParameter( "image" );
```

Γίνεται σύνδεση στη βάση και δημιουργείται ο χρήστης όπως επέλεγε.

```
DataConnection connection = new DataConnection();  // σύνδεση στη βάση
User user = new User( 0, name, password, 0, 0, image); // Φτιάχνω ένα αντικείμενο User με τα στοιχεία πο
```

Γίνεται η αποθήκευση στη βάση και στη συνεχεία η αποσύνδεση από αυτή όπως φαίνεται παρακάτω.

```
            connection.registerUser( user ); // αποθηκεύω τον χρήστη στη βάση

            connection.close();  // αποσύνδεση από τη βάση
```

Τέλος χρειάζεται να δημιουργήσουμε μια απάντηση με όλα τα νέα στοιχειά. Αυτό γίνεται εφικτό με την χρήση ενεός JSONObject.

```
            // Δημιουργώ την απάντηση
            JSONObject resp = new JSONObject();
            resp.put( "status", "OK" );
            resp.put( "message", "Registretion succesfull." );

            resp.put( "name", name );
            resp.put( "id", user.getId() );
            resp.put( "wins", user.getWins() );
            resp.put( "losses", user.getLosses() );
            resp.put( "image", user.getImage() );

            resp.writeJSONString( out );

        } finally {
            out.close();
        }
    }
```

4.3.1.6 JoinGame (Servlet)

Τέλος βλέπουμε την κλάση JoinGame .Η κλάση αυτή Καλείται όταν ο παίκτης πατήσει play. Είτε δημιουργεί νέο παιχνίδι (user2Id = -1), οπότε ο παίκτης καλεί περιοδικά την CheckGameState μέχρι να συμπληρωθεί. Είτε βρίσκει ένα παιχνίδι που έχει user2Id = -1, οπότε βάζει σε αυτό το id του παίκτη.

```
@WebServlet( name = "JoinGame", urlPatterns = { "/JoinGame" } )
public class JoinGame extends HttpServlet {
```

Παρακάτω θα εξηγήσουμε βήμα προς βήμα πως παίρνουμε τα στοιχειά για να επιτευχτεί η διαδικασία που προαναφέραμε .

Αρχικά κάνουμε το string ακέραιο και επιτα γίνεται η σύνδεση στη βάση.

```
        int userId = Integer.parseInt( request.getParameter( "user_id" ) ); // κάνω το string ακέραιο

        DataConnection connection = new DataConnection();  // σύνδεση στη βάση
```

Παίρνουμε το ανοιχτό παιχνίδι και φορτώνουμε τον πρώτο παίκτη.

```
Game game = connection.getOpenGame( userId ); // Παίρνω το ανοικτό παιχνίδι

User user1 = connection.getUserById( game.getUser1Id() ); // Φορτώνω τον 1ο παίκτη
User user2 = null;
```

Αν υπάρχει δεύτερος παίκτης τον φορτώνουμε και αυτόν. Σε κάθε περίπτωση γίνεται αποσύνδεση από τη βάση.

```
if ( game.getUser2Id() != -1 ) { // Αν υπάρχει 2ος παίκτης τον φορτώνω
    user2 = connection.getUserById( game.getUser2Id() );
}

connection.close(); // αποσύνδεση από τη βάση
```

Στο σημείο αυτό χρειάζεται να δημιουργήσουμε μια απάντηση με όλα τα νέα στοιχειά. Αυτό γίνεται εφικτό με την χρήση ενεός JSONObject.

```
// Δημιουργώ την απάντηση
JSONObject resp = new JSONObject();
resp.put( "status", "OK" );

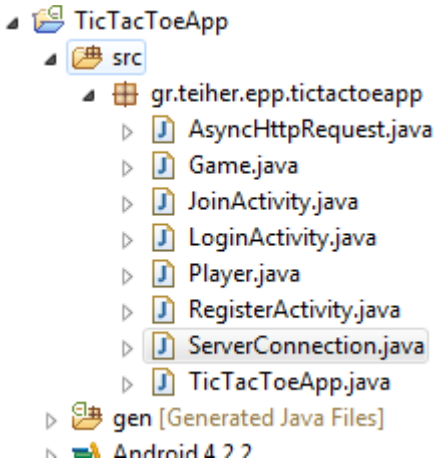
resp.put( "id", game.getId() );
resp.put( "user1_id", game.getUser1Id() );
resp.put( "user1_name", user1.getName() );
resp.put( "user1_image", user1.getImage() );
```

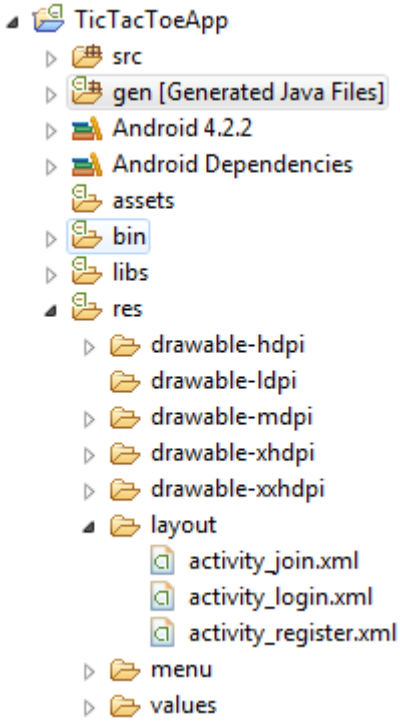
Βάζουμε τα στοιχειά του δευτέρου παίκτη αν υπάρχουν αλλιώς στο id του βάζουμε -1.

```
if ( user2 != null ) { // Βάζω τα στοιχεία του 2ου παίκτη αν υπάρχουν, αλλιώς βάζω το id -1
    resp.put( "user2_id", game.getUser2Id() );
    resp.put( "user2_name", user2.getName() );
    resp.put( "user2_image", user2.getImage() );
} else {
    resp.put( "user2_id", -1 );
```

4.4 Ανάλυση Εφαρμογής Android

Εδώ θα παρουσιάσουμε την εφαρμογή που κατασκευάσαμε για το κινητό. Θα αναλύσουμε τις κλάσεις μία προς μία.





4.4.1 Κλάση Game

Η κλάση Game είναι η κλίσει που κρατεί πληροφορίες για το παιχνίδι. Είναι το μοναδικό στιγμιότυπο της κλάσης (το μοναδικό αντικείμενο που δημιουργείται) και το instance το φτιάχνουμε μόνο μία φορά και μετά το χρησιμοποιούμε. Με αυτόν τον τρόπο δεν χρειάζεται να κάνουμε κάθε φορά new.Έτσι κρατάει τις τιμές που έχουμε δώσει σε μία οθόνη και στις επόμενες.

```
public class Game {
        // Μοναδικό στιγμιότυπο της κλάσης (το
        // Το instance το φτιάχνουμε μόνο μία φ
        // Με αυτόν τον τρόπο δεν χρειάζεται να
        // Έτσι κρατάει τις τιμές που έχουμε δώ
        private static Game instance = null;
```

Στη συνεχεία βλέπουμε τις μεταβλητές που περιέχει η κλάση όπως φαίνονται παρακάτω.

```
private int id;
private Player currentPlayer; // O
private Player player1, player2;
private String board;
private boolean turn;
```

Αρχικά βλέπουμε το πεδίο id που είναι το μοναδικό και χαρακτηριστικό πεδίο του παιχνιδιού. Είναι τύπου integer.Στη συνεχεία βλέπουμε τον currentPlayer ο όποιος είναι ο τρέχον παίκτης του παιχνιδιού και θα τον αναλύσουμε παρακάτω. Οι player1 και player2 είναι οι μεταβλητές που αντιπροσωπεύουν τους φυσικούς παίκτες. Τον τίτλο player1 η player2 τον παίρνουν ανάλογα με τη σειρά προτεραιότητας που θα εγγράφουν στο

παιχνίδι. Αν για παράδειγμα ο player1 δημιουργήσει ένα παιχνίδι τότε ο επόμενος που θα εγγράφει θα είναι ο player2. Αυτή η διαδικασία σχετίζεται άμεσα με τον currentPlayer που αναφέραμε παραπάνω. Ο currentPlayer είναι ο τρέχον παίκτης. Για παράδειγμα ο παίκτης που θα εγγράφει πρώτος είναι ο player1 άρα ο currentPlayer θα έχει τιμή ώση με player1, όποτε ο currentPlayer του player2 θα έχει τιμή player2. Η μεταβλητή board είναι τύπου συμβολοσειρά και αναπαριστά το παιχνίδι όπως οπτικοποιηται μπροστά στους χρήστες η μεταβλητή turn παίρνει τιμή ΄΄1΄΄ η ΄΄0΄΄ ανάλογα τον εκάστου παίκτη που παίζει κάθε φορά, δηλώνει τη σειρά των παικτών.

Παρακάτω βλέπουμε τους getters και setters της κλάσης Game.

```
public Game() {
}

public int getId() {
        return id;
}

public void setId(int id) {
        this.id = id;
}

public Player getPlayer1() {
        return player1;
}

public void setPlayer1(Player player1) {
        this.player1 = player1;
}

public Player getPlayer2() {
        return player2;
```

```
public Player getPlayer2() {
        return player2;
}

public void setPlayer2(Player player2) {
        this.player2 = player2;
}

public Player getCurrentPlayer() {
        return currentPlayer;
}

public void setCurrentPlayer(Player currentPlayer) {
        this.currentPlayer = currentPlayer;
}

public String getBoard() {
        return board;
}

  public void setBoard(String board) {
          this.board = board;
  }

  public boolean isTurn() {
          return turn;
  }

  public void setTurn(boolean turn) {
          this.turn = turn;
  }
```

Και τέλος βλέπουμε τη συνάρτηση getInstance.

```
public static Game getInstance() {
        if(instance == null) { // Αν είναι
                instance = new Game();
        }
        // Επιστρέφει το Game
        return instance;
}
```

Αν το στιγμιότυπο είναι null τότε αυτό σημαίνει ότι δεν υπάρχει ένα στιγμιότυπο του παιχνιδιού άρα ένα παιχνίδι. Όποτε καλείται για πρώτη φορά και έτσι δημιουργεί ένα νέο παιχνίδι και στο τέλος το επιστέφει.

4.4.2 Κλάση Player

Η κλάση Player αφορά τον φυσικό παίκτη και σε αυτήν δηλώνονται όλα τα στοιχειά του.

Αρχικά βλέπουμε το πεδίο id .

```
private int id;
```

Είναι τύπου ακέραιος, είναι μοναδικός αριθμός και χαρακτηρίζεται με αυτό το αναγνωριστικό του παίκτη.

Στην συνεχεία βλέπουμε το πεδίο name .

```
private String name;
```

Είναι τύπου συμβολοσειρά και αποθηκεύεται σε αυτό το όνομα του χρηστή.

Έπειτα βλέπουμε την μεταβλητή wins.

```
private int wins;
```

Είναι τύπου ακέραιος και αντιπροσωπεύει τις νίκες του παίκτη.

Παρόμοιε και η μεταβλητή losses.

```
private int losses;
```

Είναι τύπου ακέραιος και αντιπροσωπεύει τις ήττες του παίκτη.

Τέλος βλέπουμε το πεδίο image.

```
private String image;
```

Είναι τύπου συμβολοσειρά και αντιπροσωπεύει την εικόνα που έχει επιλέξει ο παίκτης στο παιχνίδι.

Στη συνεχεία βλέπουμε εικόνες από τους getters και setters της κλάσης Player.

```
        public Player() {
        }

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public int getWins() {
                return wins;
        }

        public void setWins(int wins) {
                this.wins = wins;
        }

        public int getLosses() {
                return losses;
        }

    public void setLosses(int losses) {
            this.losses = losses;
    }

    public String getImage() {
            return image;
    }

    public void setImage(String image) {
            this.image = image;
    }


}
```

4.4.3 Κλάση AsyncHttpRequest

Η κλάση AsyncHttpRequest χρησιμοποιείται για να ζητάμε δεδομένα από τον server σε ένα διαφορετικό thread, ώστε να μην παγώνει η εφαρμογή όσο κρατάει η διαδικασία.

Παίρνει για παραμέτρους ένα αντικείμενο τύπου BasicNameValuePair, δηλαδή μία ομάδα από ζευγάρια τιμών (πχ name, Aris) και το στέλνει στην κατάλληλη διεύθυνση.

Έπειτα όταν ο server απαντήσει δημιουργεί ένα αντικείμενο JSON που έχει μέσα τα στοιχεία της απάντησης και το γυρίζει πίσω.

Παρακάτω βλέπουμε το αντικείμενο που κάνει τη σύνδεση αλλά και τη διεύθυνση που θα ζητηθεί.

```
public class AsyncHttpRequest extends AsyncTask<BasicNameValuePair, Integer, JSONObject> {
        protected HttpClient httpClient; // Το αντικείμενο που κάνει τη σύνδεση
        protected String address; // Η διεύθυνση του Servlet που θα ζητήσει
```

Στη συνεχεία φτιάχνουμε ένα νέο client-μια νέα σύνδεση και βάζουμε τη διεύθυνση που θα επικοινωνεί όπως φαίνεται παρακάτω.

```
@Override
protected JSONObject doInBackground(BasicNameValuePair... params) {
        HttpClient httpClient = new DefaultHttpClient(); // Φτιάχνω τη σύνδεση

        HttpPost httpPost = new HttpPost(address);// Βάζω τη διεύθυνση
```

Έπειτα βάζουμε τις παραμέτρους που θα αποστείλουμε στη σύνδεση αυτή.

```
// Βάζω τις παραμέτρους που θα στείλω
List<NameValuePair> nameValuePairList = new ArrayList<NameValuePair>();

for (BasicNameValuePair pair : params) {
        nameValuePairList.add(pair);
}
//////////
```

Τέλος βλέπουμε τη διαδικασία κατασκευής ενός HTTP Request.

```
try {
        HttpResponse response = httpClient.execute(httpPost);

        // Παίρνω την απάντηση και τη βάζω σε ένα String
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        response.getEntity().writeTo(out);
        String responseString = out.toString();
        ////////////
```

Παίρνουμε το response-την απάντηση και την βάζουμε σε μια συμβολοσειρά.

```
// Δημιουργώ ένα αντικείμενο JSON από το παραπάνω String και το επιστρέφω.
JSONObject result = new JSONObject(responseString);
return result;
```

Δημιουργούμε ένα αντικείμενο JSON και το επιστρέφουμε.

4.4.4 Κλάση TPoint

Η κλάση TPoint είναι μία βοηθητική κλάση που περιέχει δύο ακέραιους. Χρησιμοποιείται ώστε να είναι πιο εύκολο να περάσουμε σαν παραμέτρους ή σαν τιμές επιστροφής τη θέση ενός αντικειμένου.

```
public class TPoint {
    protected int i, j;

    public TPoint(int i, int
        this.i = i;
        this.j = j;
    }

    public int getI() {
        return i;
    }

    public void setI(int i) {
        this.i = i;
    }

    public int getJ() {
        return j;
    }

    public void setJ(int j) {
        this.j = j;
    }


}
```

4.4.5 Κλάση TicTacToeGame

Η κλάση TicTacToeGame περιέχει τη λογική του παιχνιδιού. Έχει ένα πίνακα με αντικείμενα Symbol που είναι η κατάσταση του ταμπλό.

```
private Symbol board[][] = {
    { Symbol.NONE, Symbol.NONE, Symbol.NONE },
    { Symbol.NONE, Symbol.NONE, Symbol.NONE },
    { Symbol.NONE, Symbol.NONE, Symbol.NONE },
};
```

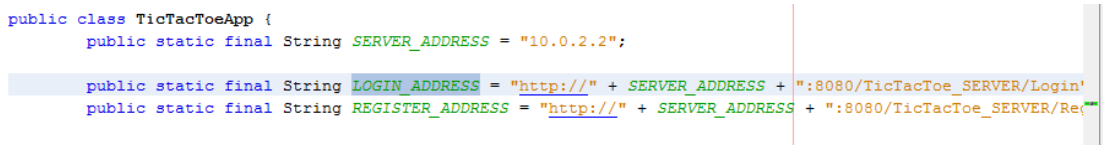
Έπειτα έχει διάφορες βοηθητικές συναρτήσεις για να επιστρέφει τι βρίσκεται σε μία συγκεκριμένη θέση ή για να αλλάζει την τιμή μίας θέσης.

```
public Symbol getSymbolAtPosition( TPoint point ) {
    return getSymbolAtPosition( point.getI(), point.getJ() );
}

public Symbol getSymbolAtPosition( int i, int j ) {
    return board[i][j];
}
```

4.4.6 *Κλάση TicTacToeApp*

Στην κλάση TicTacToeApp βλέπουμε τη διεύθυνση του server που μας δίνει το emulator, την διεύθυνση εισαγωγής LOGGIN_ADDRESS και την διεύθυνση REGISTER_ADDRESS.



Όπως βλέπουμε η πόρτα σύνδεσης είναι η 8080 by default.

4.4.7 *Κλάση Symbol*

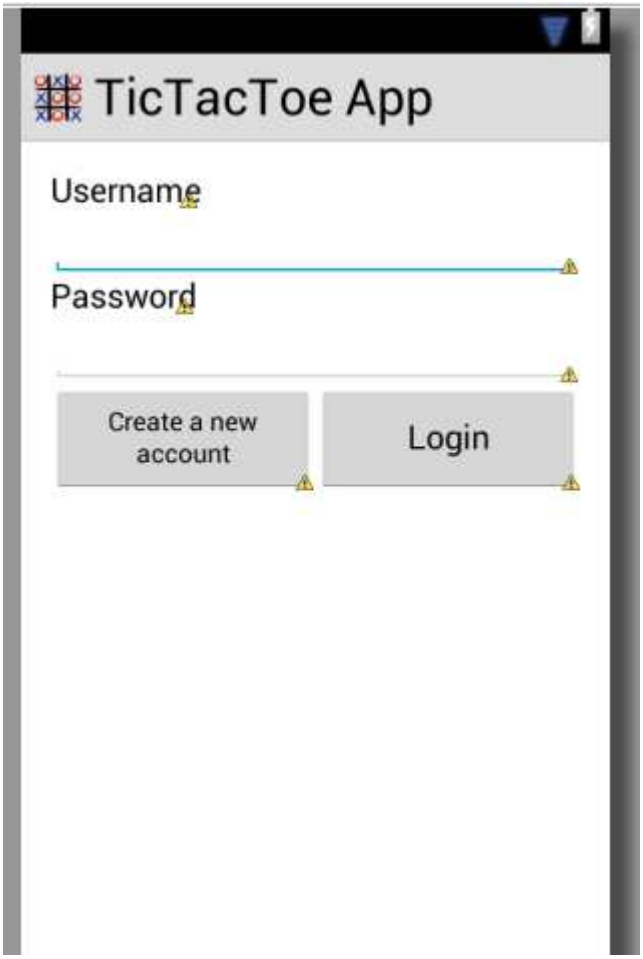
Η κλάση Symbol είναι ένα enum που περιέχει τα σύμβολα που περιέχονται στο ταμπλό του παιχνιδιού. Αυτά είναι τα X, Y και NONE αν μία θέση είναι κενή.

```
package gr.teiher.epp.tictactoeapp;

public enum Symbol {
    X,
    O,
    NONE
}
```

4.4.8 *Activity Login*

Το Activity Login είναι η οθόνη που βλέπει ο παίκτης όταν ανοίγει το παιχνίδι. Εκεί βάζει τα στοιχεία του ώστε να συνδεθεί.



4.4.8.1 Κλάση LoginActivity

4.4.9 Activity Register

Το activity register είναι η φόρμα που βλέπει ο χρήστης όταν θέλει να δημιουργήσει έναν νέο λογαριασμό.



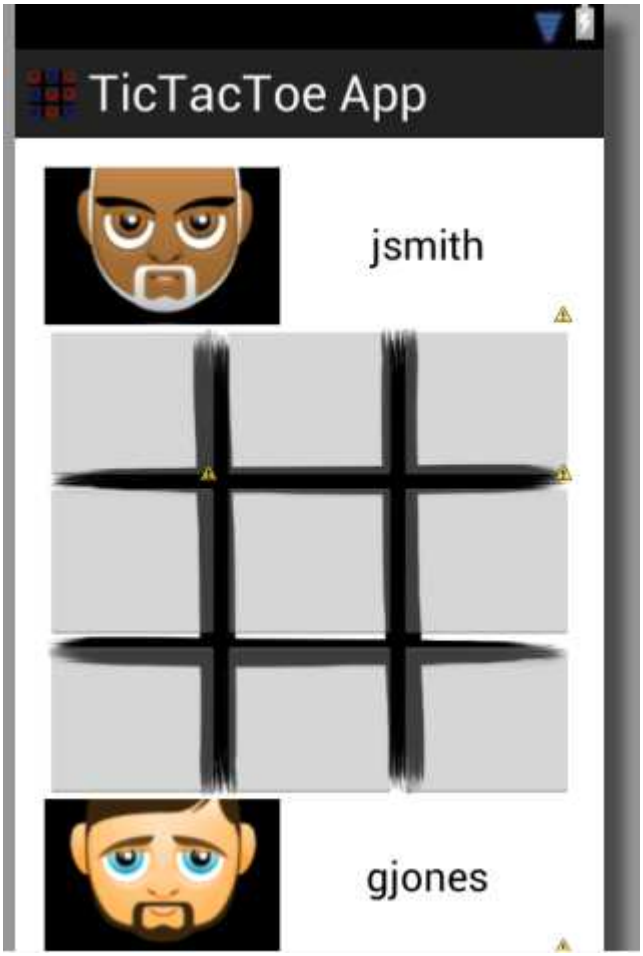
4.4.10 Activity Join

Το activity join είναι η οθόνη που βλέπει ο χρήστης αφού συνδεθεί ώστε να συνδεθεί σε ένα παιχνίδι με κάποιον άλλο χρήστη.



4.4.11 Activity Main

Το activity main είναι η κύρια οθόνη του παιχνιδιού. Αυτή δείχνει τους παίκτες και το ταμπλό της τρίλιζας.


TicTacToe App
jsmith
gjones

5 Συμπεράσματα

Για την ανάπτυξη της εφαρμογής χρειάστηκε να ασχοληθούμε με ένα μεγάλο εύρος τεχνολογιών, συγκεκριμένα με τη βάση δεδομένων MySQL, τον Apache Tomcat Server και τα εργαλεία ανάπτυξης εφαρμογών Android. Επιπλέον χρειάστηκε γνώση και κάποιων βοηθητικών εφαρμογών, όπως η βιβλιοθήκη JDBC για την επικοινωνία του server με τη βάση δεδομένων και το JSON για την αποστολή των αποτελεσμάτων σε μία κατανοητή μορφή.

Η ανάπτυξη του μέρους του προγράμματος που σχετίζεται με τη βάση δεδομένων ήταν γενικά απλή. Τόσο οι τεχνολογίες των servlets όσο και το JDBC είναι αρκετά ώριμες και εύχρηστες.

Το JSON δεν παρουσιάζει καμία δυσκολία και οι βιβλιοθήκες της Java που ασχολούνται με αυτό είναι αρκετά απλές στη χρήση.

Τα εργαλεία για ανάπτυξη εφαρμογών Android είναι τα πιο πολύπλοκα απ' όλα και τα πιο δύσκολα στη χρήση. Αν και πρακτικά πρόκειται για μία βιβλιοθήκη ανάπτυξης γραφικών περιβαλλόντων η φιλοσοφία απέχει πολύ από το Swing και χρειάζεται πολύ περισσότερος χρόνος για την κατανόηση κάποιων βασικών λειτουργιών.

Το δυσκολότερο όμως μέρος απ' όλα ήταν ο συνδυασμός όλων των παραπάνω σε μία εφαρμογή ώστε να επικοινωνούν σωστά μεταξύ τους. Αυτό συνεπάγεται γνώση του τρόπου λειτουργίας του κάθε αντικειμένου σε βάθος.

Παρόλα αυτά με την κατάλληλη ενασχόληση μπορούν να δημιουργηθούν αξιόλογες εφαρμογές με πάρα πολλές δυνατότητες.

6 Βιβλιογραφία

1. http://java.com/en/
2. http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/
3. http://tomcat.apache.org/
4. http://www.android.com/
5. http://www.eclipse.org/
6. https://netbeans.org/
7. http://www.wampserver.com/en/
8. http://www.webopedia.com/TERM/A/Android_SDK.html
9. http://www.json.org/javadoc/org/json/JSONObject.html
10. http://developer.android.com/reference/android/app/Activity.html