ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ
ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ
ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΠΟΛΥΜΕΣΩΝ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Υλοποίηση Διαδικτυακής Πλατφόρμας Δημιουργίας Στατιστικών Δεδομένων σε Ιατρικές Βάσεις Δεδομένων»

ΡΟΥΣΣΟΣ ΠΑΝΑΓΙΩΤΗΣ

Επιβλέπων Καθηγητής : Ζαχαρόπουλος Βασίλειος

ΗΡΑΚΛΕΙΟ 2007

1. Περίληψη 3
1.1 Σκοπός της Πτυχιακής Εργασίας 3
1.2 Ορισμός του Προβλήματος 3
1.3 Προτεινόμενη Λύση 3
2. Αρχιτεκτονική και Τεχνολογίες 4
2.1 Αρχιτεκτονική της εφαρμογής 4
2.1.1 Δομή της εφαρμογής – Επίπεδο Χρήστη 4
2.1.2 Δομή της Εφαρμογής - Επίπεδο Σχεδίασης 5
2.1.3 Επίπεδο Δεδομένων 5
2.1.4 Επίπεδο Διαδικασιών 6
2.1.5 Επίπεδο Παρουσίασης 7
2.1.6 Ρόλοι Χρηστών και Δικαιώματα 7
2.1 Τεχνολογίες 8
2.1.1 ASP.NET 8
2.1.1.1 Εισαγωγή στην ASP.NET 8
2.1.1.2 Παρουσίαση της C# 8
2.1.1.3 Εξέλιξη της C# 10
2.1.2 ADO.NET 11
2.1.3 Βάσεις Δεδομένων 11
2.1.3.1 Microsoft SQL Server 12
2.1.3.2 Microsoft Access 12
2.1.4 Internet Information Services (IIS) 13
3. Λειτουργική Περιγραφή της Εφαρμογής 14
3.1 Εισαγωγή 14
3.2 Εισαγωγική Σελίδα 14
3.3 Κύρια Σελίδα 15
3.4 Menu Διαχείριση 16
3.4.1 Βάσεις Δεδομένων 16
3.4.2 Ερωτήματα 17
3.4.2.1 Λίστα Group Ερωτημάτων 17
3.4.2.2 Λίστα ερωτημάτων 18
3.4.2.3 Φόρμα Δημιουργίας και Επεξεργασίας Ερωτήματος 19
3.4.2.4 Φόρμα Εκτέλεσης Ερωτημάτων 23
3.5.3 Διαχείριση Χρηστών 24
3.5.3.1 Λίστα Χρηστών 24
3.5.3.2 Ρόλοι Χρηστών και Δικαιώματα 25
4. Τεχνική Περιγραφή της Εφαρμογής 26
4.1 Εισαγωγή 26
4.2 Εισαγωγή στα User Controls 27
4.2 Εισαγωγική Σελίδα 28
4.3 Κύρια Σελίδα 30
4.4 Κατηγορίες Σελίδων 30
4.4.1 Σελίδα με Απλή Λίστα 31
4.4.2 Σελίδα με Σύνθετη Λίστα 32
4.4.3 Σελίδα Επεξεργασίας Ερωτήματος 34
4.4.4 Σελίδα Εκτέλεσης Ερωτήματος 42
4.5 Εκτέλεση Ερωτήματος και User Controls 45
4.5.1 Απλό – SimpleStat User Control 45
4.5.2 Πολλαπλό – MultiStat User Control 46
4.5.3 Γράφημα – PieChartStat User Control 54
5. Βελτιστοποίηση της Εφαρμογής 55
5.1 Υποστήριξη νέων τύπων Βάσεων Δεδομένων 56
5.2 Υποστήριξη Web Services 56
5.3 Διεργασία Αντιγράφων Ασφαλείας 56
6. Βιβλιογραφικές Αναφορές 57

1. Περίληψη

1.1 Σκοπός της Πτυχιακής Εργασίας

Σκοπός της πτυχιακής εργασίας είναι να καλύψει τις ανάγκες του εργαστηρίου Ιατρικής Πληροφορικής και Τηλεϊατρικής (TCDN) του τμήματος Εφαρμοσμένης Πληροφορικής και Πολυμέσων (ΕΠΠ) της Σχολής Τεχνολογικών Εφαρμογών για την παραγωγή και προβολή στατιστικών δεδομένων από ιατρικές βάσεις δεδομένων.

1.2 Ορισμός του Προβλήματος

Το εργαστήριο Ιατρικής Πληροφορικής και Τηλεϊατρικής έχει αναπτύξει ένα πληροφοριακό σύστημα για την κλινική της Χειρουργικής Ογκολογίας του Πανεπιστημιακού Γενικού Νοσοκομείου Ηρακλείου (ΠΕ.ΠΑ.Γ.Ν.Η). Από την λειτουργία του πληροφοριακού συστήματος παρατηρήθηκε από τους χρήστες (ιατρούς – νοσηλευτικό προσωπικό) η ανάγκη προβολής στατιστικών δεδομένων για την παρακολούθηση της κλινικής. Σε τακτά χρονικά διαστήματα το εργαστήριο (TCDN) λάμβανε απαιτήσεις για τέτοια δεδομένα, γεγονός το οποίο απαιτούσε αρκετό φόρτο εργασίας απο τους συμμετέχοντες στο project αυτό.

1.3 Προτεινόμενη Λύση

Για την αντιμετώπιση του παραπάνω θέματος προτείνεται η υλοποιήση μιας εφαρμογής η οποία θα απλοποιεί τις διαδικασίες παραγωγής και προβολής στατιστικών δεδομένων. Η εφαρμογή πρέπει να είναι με τέτοιο τρόπο δομημένη, ώστε η δημιουργία των στατιστικών δεδομένων να μπορεί να γίνει από χρήστες που κατέχουν τις βασικές γνώσεις της Structured Query Language (SQL). Χρειάζεται ακόμα να ορισθούν ρόλοι χρηστών, αφού η εφαρμογή μπορεί να υποστηριξει πολλές βάσεις δεδομένων απο διαφορετικές εφαρμογές, καθώς επίσης και δικαιώματα στον κάθε ρόλο.

Επίσης για να μπορούν οι χρήστες της εφαρμογής (ιατροί – νοσηλευτικό προσωπικό) να παρακολουθούν τους δείκτες ανά πάσα στιγμή και από οποιοδήποτε χώρο, είτε εντός, είτε εκτός του νοσοκομείου, προτείνεται η εφαρμογή αυτή να βασίζεται στην τεχνολογία του διαδικτύου (Client/Server Architecture).

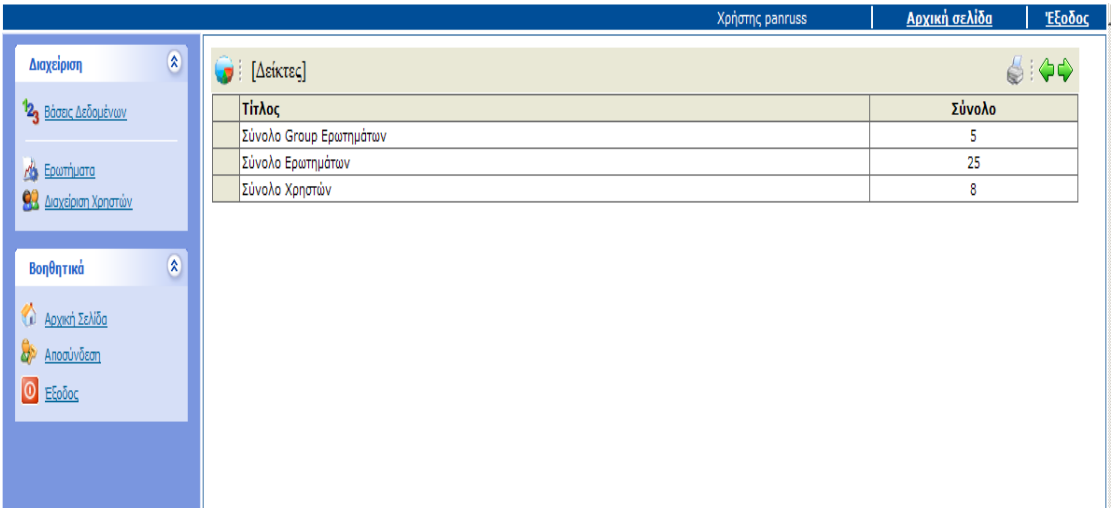
2. Αρχιτεκτονική και Τεχνολογίες

2.1 Αρχιτεκτονική της εφαρμογής

Στην ενότητα αυτή θα περιγράψουμε την λογική και τον τρόπο με τον οποίο έχει υλοποιηθεί η εφαρμογή αυτή, όπως για παράδειγμα η δομή της και τα επίπεδα της.

2.1.1 Δομή της εφαρμογής – Επίπεδο Χρήστη

Η εφαρμογή έχει σχεδιαστεί με τέτοιο τρόπο ώστε να διευκολύνει τον χρήστη για την λειτουργία της. Όπως φαίνεται και από την *Εικόνα 1. Κεντρική Σελίδα Εφαρμογής*, αφού ο χρήστης πιστοποιηθεί οτι ανήκει στην εφαρμογή, παραπέμπεται στην βασική οθόνη.



Εικόνα 1. Κεντρική Σελίδα Εφαρμογής

Όπως μπορούμε να παρατηρήσουμε από την εικόνα, η εφαρμογή χωρίζεται σε τρία τμήματα. Το πρώτο τμήμα βρίσκεται στο πάνω μέρος της εφαρμογής και είναι η επικεφαλίδα της εφαρμογής. Η επικεφαλίδα περιέχει πληροφορίες για τον χρήστη ο οποίος είναι συνδεδεμένος τη δεδομένη στιγμή καθώς επίσης δίνει την δυνατότητα στον χρήστη να μετακινηθεί στην αρχική σελίδα και την δυνατότητα να αποσυνδεθεί απο την εφαρμογή.

Το δεύτερο τμήμα της εφαρμογής βρίσκεται στο αριστερό της μέρος και αποτελεί το menu για την μετάβαση ανάμεσα στις λειτουργίες. Όπως μπορούμε να δούμε υπάρχουν δύο βασικά menu, η «Διαχείριση» και τα «Βοηθητικά». Το κάθε menu έχει την δική του λειτουργία, την οποία και θα αναλύσουμε στα επόμενα κεφάλαια.

Το τρίτο τμήμα της εφαρμογής βρίσκεται στο κέντρο και αποτελεί την κεντρική της σελίδα. Στο μέρος αυτό εμφανίζονται όλα τα υποσυστήματα και οι λειτουργίες της εφαρμογής. Συνήθως οι σελίδες που εμφανίζονται στο μέρος αυτό αποτελούνται από μία δική τους επικεφαλίδα η οποία περιέχει πληροφορία την λειτουργία της. Η λειτουργία δηλώνεται μέσω του τίτλου της, όπως για παράδειγμα «Λίστα Ερωτημάτων». Ο τίτλος αυτός δηλώνει οτι βρισκόμαστε σε μια σελίδα που περιέχει μια λίστα με ερωτήματα.

2.1.2 Δομή της Εφαρμογής - Επίπεδο Σχεδίασης

Σε επίπεδο σχεδίασης η εφαρμογή μπορούμε να πούμε οτι αποτελείται από τρία μέρη επίσης. Το επίπεδο των δεδομένων (Data layer), το επίπεδο των διαδικασίων(Application/Business layer) και το επίπεδο της χρήσης ή παρουσίασης (Presentation Layer). Το επίπεδο των δεδομένων είναι το επίπεδο στο υπάρχουν τα δεδομένα της εφαρμογής και αυτό δεν είναι άλλο απο την Βάση Δεδομένων. Στην Βάση Δεδομένων υπάρχουν όλες οι πληροφορίες που χρειάζονται για να λειτουργήσει η εφαρμογή.

Στο επίπεδο των διαδιακασίων βρίσκονται όλες οι κλάσεις της εφαρμογής που έχουν υλοποιηθεί ώστε να λειτουργήσει η εφαρμογή. Το επίπεδο αυτό αποτελεί ουσιαστικά την βιβλιοθήκη της εφαρμογής.

Το επίπεδο χρήσης είναι αυτό που λαμβάνει ο τελικός χρήστης, η διεπαφή δηλαδή της εφαρμογής από την οποία μπορούμε να θέσουμε σε λειτουργία όλες αυτές τις επιλογές που μας προσφέρει η εφαρμογή.

2.1.3 Επίπεδο Δεδομένων

Για την λειτουργία της διαδικτυακής εφαρμογής χρησιμοποιείται μία βάση δεδομένων. Στην συγκεκριμένη εφαρμογή χρησιμοποιούμε τον Microsoft SQL Server (MS SQL Server 2000) για τον οποίο θα μιλήσουμε αναλυτικά στην πορεία. Θα μπορούσαμε κάλλιστα να χρησιμοποιήσουμε σαν βάση δεδομένων την Microsoft Access, καθώς υποστηρίζεται από την εφαρμογή. Ο βασικός λόγος που επιλέξαμε να χρησιμοποιήσουμε τον MS SQL Server για βάση δεδομένων ήταν για να εξοικειωθούμε με το περιβάλλον του και τις σχεσιακές βάσεις δεδομένων. Η βάση δεδομένων χρησιμοποιείται για την αποθήκευση της πληροφορίας που επιλέγει ο χρήστης.

Οι πληροφορίες που θα αποθηκεύονται στην βάση δεδομένων της εφαρμογής είναι οι ακόλουθες:

- Στατιστικά Δεδομένα

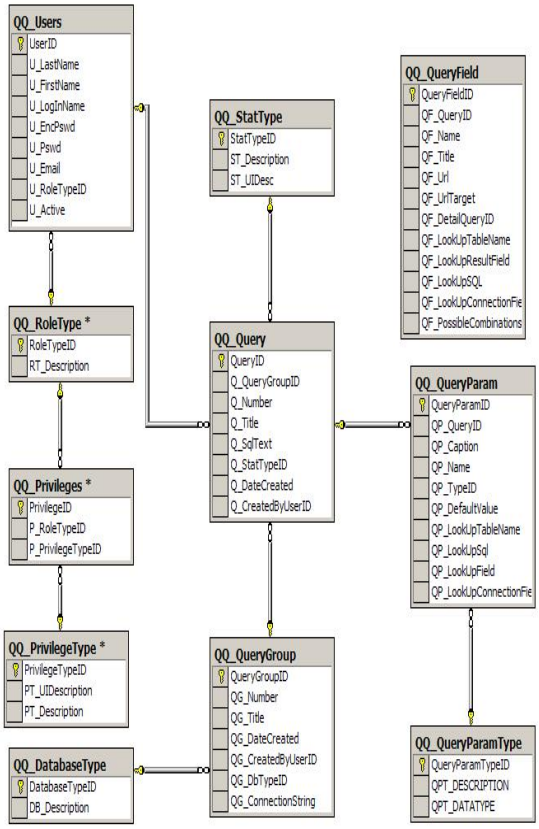
Οι πληροφορίες που περιέχουν τα στατιστικά δεδομένα αφορούν πληροφορίες σχετικά με τις βάσεις δεδομένων απο τις οποίες θα αντληθούν τα στατιστικά αυτά, καθώς επίσης και τα ερωτήματα που τίθονται.

- Χρήστες, ομάδες χρηστών και δικαιώματα πρόσβασης

Στην βάση δεδομένων αποθηκεύονται οι χρήστες, οι ομάδες χρηστών και τα δικαιώματα πρόσβασης τους στην εφαρμογή.

Η Βάση δεδομένων που χρησιμοποιούμε αποτελείται από βασικές και βοηθητικές οντότητες όπως φαίνεται και από την

Εικόνα 2. Βάση Δεδομένων.



Βασικές οντότητες:

- QQ_Query
- QQ_QueryGroup
- QQ_QueryParam
- QQ_QueryField
- QQ_Users
- QQ_RoleType
- QQ_Privileges

Βοηθητικές Οντότητες:

- QQ_StatType
- QQ_QueryParamType
- QQ_PrivilegeType
- QQ_DataBaseType

Εικόνα 2. Βάση Δεδομένων

Όπως μπορούμε να δούμε το σχήμα της βάσης δεδομένων που χρησιμοποιούμε είναι αρκετά απλό και καταννοητό. Έχουμε προσπαθήσει να χρησιμοποιήσουμε όλους τους κανόνες που έχουμε διδαχθεί στα σχετικά μαθήματα για τον σχεδιασμό μιας βάσης δεδομένων και την κανονικοποίηση της.

2.1.4 Επίπεδο Διαδικασιών

Το επίπεδο των διαδικασιών είναι το επίπεδο στο οποίο έχει υλοποιηθεί όλη η επιχειρησιασκή λογική που διέπει την εφαρμογή, δηαλδή όλοι οι λογικοι κανόνες που αφορούν την λειτουργία της. Μπορούμε να πούμε οτι βρίσκεται στο ενδιάμεσο επίπεδο της εφαρμογής καθώς δέχεται δεδομένα απο το επίπεδο των δεδομένων και βάσει των

κανόνων που έχει τα προωθεί με τον κατάλληλο τρόπο στο ανώτερο επίπεδο, το επίπεδο της χρήσης ή παρουσίασης.

2.1.5 Επίπεδο Παρουσίασης

Το επίπεδο της παρουσίασης είναι το ανώτερο επίπεδο μιας εφαρμογής και είναι αυτό που ουσιαστικά βλέπει ένας χρήστης. Ο χρήστης με την διεπαφή της εφαρμογής μπορεί να εκτελέσει τις εργασίες που θέλει και να μην το απασχολεί τι ακριβώς εκτελείται

2.1.6 Ρόλοι Χρηστών και Δικαιώματα

Στην εφαρμογή έχουν οριστεί *δικαιώματα* και *ρόλοι χρηστών*. Δικαιώματα είναι οι ενέργειες που μπορεί να κάνει ένας εγγεγραμμένος στην εφαρμογή χρήστης, όπως για παράδειγμα επεξεργασία ερωτήματος. Ρόλοι χρηστών είναι τύποι χρηστών στους οποίους έχουν οριστεί κάποια δικαιώματα.

Τα δικαιώματα που μπορούν να οριστούν στου ρόλους χρηστών είναι τα εξής:

1. Δημιουργία Ερωτήματος / Group Ερωτημάτων
2. Επεξεργασία Ερωτήματος / Group Ερωτημάτων
3. Διαγραφή Ερωτήματος / Group Ερωτημάτων
4. Δημιουργία Χρηστών
5. Επεξεργασία Χρηστών
6. Διαγραφή Χρηστών

Η εφαρμογή έχει δημιουργηθεί με τρεις βασικούς ρόλους με τα αντίστοιχα δικαιώματα τους, οι οποίοι είναι:

- Διαχειριστής Συστήματος
  - Δημιουργία Ερωτήματος / Group Ερωτημάτων
  - Επεξεργασία Ερωτήματος / Group Ερωτημάτων
  - Διαγραφή Ερωτήματος / Group Ερωτημάτων
  - Δημιουργία Χρηστών
  - Επεξεργασία Χρηστών
  - Διαγραφή Χρηστών
- Συντάκτης Ερωτημάτων
  - Δημιουργία Ερωτήματος / Group Ερωτημάτων
  - Επεξεργασία Ερωτήματος / Group Ερωτημάτων
  - Διαγραφή Ερωτήματος / Group Ερωτημάτων
  - Δημιουργία Χρηστών
  - Επεξεργασία Χρηστών
- Χρήστης Ερωτημάτων
  - Δεν έχουν οριστεί δικαιώματα για τον ρόλο αυτό.

Αυτό σημαίνει οτι ο ρόλος αυτός έχει δικαίωμα να βλέπει τα ερωτήματα που του αναλογούν βάσει του ρόλου του και να τα εκτελεί.

Πέρα από τους βασικούς ρόλους της εφαρμογής μπορούν να δημιουργηθούν νέοι ρόλοι χρηστών βάσει των δικαιωμάτων που έχουμε αναφέρει.

Στο σημείο αυτό να σημειώσουμε οτι τα ερωτήματα που μπορούμε να δούμε είναι πάντα βάσει του ρόλου μας. Για να γίνει πιο καταννοητό να φέρουμε σαν παράδειγμα τον ρόλο «Διαχειριστή Συστήματος» και τον ρόλο «Συντάκτης Ερωτημάτων». Ο ρόλος «Διαχειριστή Συστήματος» είναι αυτοννόητο ότι μπορεί και βλέπει όλα τα ερωτήματα και όλους τους χρήστες που υπάρχουν στην εφαρμογή, σε αντίθεση με τον ρόλο

«Συντάκτης Ερωτημάτων», οποίος μπορεί να δει μόνο τα ερώτημα και τους χρήστες που έχει δημιουργήσει.

2.1 Τεχνολογίες

2.1.1 ASP.NET

2.1.1.1 Εισαγωγή στην ASP.NET

Η τεχνολογία ASP.NET είναι μια τεχνολογία που χρησιμοποιείται για την ανάπτυξη και υλοποιήση διαδικτυακών εφαρμογών. Έχει υλοποιηθεί από την εταιρεία Microsoft και είναι μια αναβαθμισμένη έκδοση της τεχνολογίας ASP(Active Server Pages). Στόχος της ASP.NET είναι να προσφέρει ένα τρόπο ανάπτυξης διαδικτυακών εφαρμογών εφάμιλο με την ανάπτυξη εφαρμογών επιτραπέζιων υπολογιστών. Η ASP.NET πέρα από τα βασικά HTML Control περιέχει δικά της Controls που δεν ταιριάζουν στα κλασικά, αλλά έχουν δημιουργηθεί με γνώμονα τα απλά HTML Controls. Για παράδειγμα το System.Web.UI.WebControls.DataGrid είναι μια κλάση η οποία αποτυπώνει τα δεδομένα από μία βάση δεδομένων σαν μια λίστα. Η λίστα αυτή αποτελείται από HTML πίνακες, αλλά ο μηχανικός προγραμματίζει το Control αυτό απο ένα υψηλότερο επίπεδο προσφέροντας του περισσότερες δυνατότητες. Όλες οι κλάσεις για τις διαδικτυακές εφαρμογές υποστηρίζουν ένα μοντέλο γεγονότων το οποίο διευκολύνει την αλληλεπίδραση με τον τελικό χρήστη. Το μοντέλο γεγονότων αυτό είναι παρόμοιο με αυτό το μοντέλο που χρησιμοποείται για την ανάπτυξη εφαρμογών επιτραπέζιων υπολογιστών.

Οι κλάσεις για τις εφαρμογές διαδικτύου(WebForms) παρέχουν πλέον πολλές υπηρεσίες οι οποίες κάνουν το σχεδιασμό και την δημιουργία τέτοιων εφαρμογών πιο εύκολη. Φόρμες σχεδιασμού, ιδιότητες σε Controls και φόρμες και ο καθορισμός διαχείρισης των γεγονότων σε φόρμες και Controls είναι μερικές απο αυτές τις υπηρεσίες.

Σήμερα μπορούμε να πούμε οτι η τεχνολογία αυτή έχει γίνει ευρέως αποδεκτή καθώς είναι πιο σταθερή και πιο γρήγορη από την προκάτοχο της, παρόλη την δυσκολία της.

2.1.1.2 Παρουσίαση της C#

Η γλώσσα προγραμματισμού C#(προφέρεται ως «C sharp») αποτελεί την λύση της εταιρείας Microsoft σε αριθμό θεμάτων που αντιμετώπιζαν οι Μηχανικοί Λογισμικού κατά την ανάπτυξη εφαρμογών με την χρήση γλωσσών προγραμματισμού όπως η C και η C++. Η C# είναι μια μοντέρνα αντικειμενο-στραφής γλώσσα προγραμματισμού η οποία προσφέρει γρήγορη και εύκολη ανάπτυξη μιας μεγάλης γκάμας εφαρμογών για την Microsoft .Net πλατφόρμα, η οποία παρέχει μια σειρά από εργαλεία και υπηρεσίες.

Πάνω από όλα η C# έχει σχεδιαστεί για να φέρει στον προγραμματιστή της C++ την ταχύτητα ανάπτυξης, χωρίς να θυσιάζει την δύναμη και τον έλεγχο που υπάρχει στις γλώσσες προγραμματισμού C και C++. Λόγω της κληρονομιάς της, η C# έχει μεγάλο βαθμό συγγένειας με την C και την C++, κάνοντας τους προγραμματιστές με τις σχετικές γνώσεις, να μεταπηδήσουν εύκολα σε αυτήν την νέα γλώσσα προγραμματισμού.

Η κεντρική ιδέα σχεδιασμού της C# είναι η απλότητα παρά η ταχύτητα. Η C# προσφέρει αρκετά οφέλη για τους προγραμματιστές που την χρησιμοποιούν, όπως:

- Απλότητα

- Συνοχή
- Εκσυγχρονισμός
- Αντικειμενοστράφεια
- Ασφάλεια Τύπων Μεταβλητών
- Κλιμάκωση
- Συμβατότητα

Ας αναλύσουμε τώρα το κάθε ένα σημείο.

Απλότητα:

Σύμφωνα με την πλειοψηφία των προγραμματιστών το πιο ενοχλητικό στην γλώσσα προγραμματισμου C είναι η χρήση των δεικτών «->», η χρήση «::» για τα μέλη κλάσης και η χρήση της τελείας «.». Για τον λόγο αυτό η C# αναγνωρίζει αυτόματα το κάθε ένα από τα παραπάνω και όλα δηλώνονται πλέον μόνο με την χρήση της τελείας «.».

Ένα άλλο θέμα είναι με τους τύπους των μεταβλητών. Στην C# πλέον ένας 64-bit ακέραιος είναι long και όχι __int64, ένας χαρακτήρας(char) είναι χαρακτήρας(char). Δεν υπάρχουν πλέον unsigned char, signed char και wchar_t.

Ένα τρίτο θέμα είναι οτι στην C και C++, οι ακέραιοι χρησιμοποιούνται ως λογικοί έλεγχοι (Boolean), προκαλώντας αρκετά λάθη. Η C# διαχωρίζει τα δύο αυτά είδη, παρέχοντας ένα τύπο μεταβλητής τύπου Bool. Οι τιμές της μεταβλητής αυτής μπορεί να είναι «true» ή «false» και μπορεί να μετατραπεί σε άλλους τύπους μεταβλητών.

Συνοχή:

Η C# εννοποιεί τους τύπους συστήματος επιτρέποντας στον χρήστη να δει τον κάθε τύπο σαν ένα αντικείμενο (object). Είτε ο χρήστης χρησιμοποιεί κλάση, είτε δομή, είτε πίνακα μπορεί να το αντιμετοπίσει σαν ένα αντικείμενο. Τα αντικείμενα συνδέονται με namespaces, πράγμα το οποίο επιτρέπει την χρήση τους προγραμματιστηκά. Αυτό σημαίνει οτί αν θέλουμε να χρησιμοποιήσουμε μια βιβλιοθλήκη αντί για το γνωστό τρόπο που δηλώναμε στην C και στην C++

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

Χρησιμοποιούμε αυτόν τρον τρόπο.

```
using System;
```

Εκσυγχρονιμός:

Όπως οι γλώσσες προγραμματισμού έτσι και οι απαιτήσεις των προγραμματισμών εξελίσσονται και αυξάνονται. Η C και η C++ είχανε μείνει πίσω τεχνολογικά σε κάποια θέματα και έτσι σχεδιάστηκε από την αρχή με τις απαραίτητες προσθήκες, όπως ο Garbage Collector ο οποίος καθαρίζει την μνήμη από οτιδήποτε δεν χρησιμοποιείται.

Η C# έχει επίσης ένα πλουσιότερο εσωτερικό μοντέλο για την διαχείριση σφαλμάτων από οτι η C++. Η C# βελτιώνει το θέμα αυτό παρέχοντας ολοκληρωμένη υποστήριξη για την εμφάνιση (throw), try…catch, try…finally σαν μηχανισμούς ελέγχου σφαλμάτων.

Ένα άλλο σημείο στο οποίο έχει εκσυγχρονιστεί η C# είναι η χρήση για οικονομικούς και ημερολογιακούς τύπων δεδομένων. Δανειζόμενη από άλλες γλώσσες όπως η SQL, η C# υποστηρίζει τέτοιους τύπους δεδομένων με την χρήση decimal και string.

Έχει βελτίωσει επίσης και την διαδικασία ελέγχου του κώδικα για λάθη (debugging). Μέχρι τώρα για να μπορέσει ο χρήστης να κάνει debug στην εφαρμογή που έχει αναπτύξει έπρεπε να υλοποιήσει ένα ξεχωριστό πρόγραμμα για το σκοπό αυτό. Στο τέλος καταλήγαμε με δύο υλοποιήσεις, μία για debug και μία για πώληση.

Αντικειμενοστράφεια (Object Oriented):

Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού όπως και οι προκάτωχοι της, με την διαφορά οτι έχει καταργήσει αρκετά σημαντικά σημεία τα οποία δυσκόλευαν την ανάπτυξη εφαρμογών, όπως η κληρονομηκότητα από πολλές κλάσεις παράλληλα. Η C# σαν αντικειμενοστραφής γλώσσα προγραμματισμού υποστηρίζει όλα τα βασικά χαρακτηριστικά όπως η ενθυλάκωση (encapsulation), πολυμορφισμός (polymorphism) και κληρονομηκότητα (inheritance). Καταργεί επίσης της global συναρτήσεις, μεταβλητές και σταθερές(constants) και αντί για αυτές μπορούν να δημιουργηθούν στατικά μέλη κλάσης, κάνοντας έτσι τον κώδικα πιο καταννοητό.

Ασφάλεια Τύπων Μεταβλητών (Type-Safety):

Έχει επικρατήσει η λογική οτι η ασφαλής χρήση των μεταβλητών δημιουργεί σωστά και δυνατά προγράμματα. Πολλά χαρακτηριστικά που προάγουν την σωστή εκτέλεση του κώδικα που υπάρχουν στην γλώσσα προγραμματισμού Visual Basic έχουν συμπεριληφθεί στην C#. όπως για παράδειγμα όλα τα δυναμικά στην μνήμη αντικείμενα και πίνακες αρχικοποιούνται αυτομάτως με μηδενική τιμή. Παρόλο που η C# δεν αρχικοποιεί τοπικές μεταβλητές, ο μεταγλωτιστής(compiler) θα μας ενημερώσει για το θέμα αυτό.

Στην C# δεν μπορείς να δημιουργήσεις λάθος δηλώσεις. Ο Grabage Collector φροντίζει να μην αφήνουμε δηλώσεις που δεν στέκουν στον κώδικα μας. Αριθμητικές πράξεις και μετατροπές δεν επιτρέπονται αν δεν υποστηρίζονται απο την μεταβλητή που χρησιμοποιείται.

Κλιμάκωση:

Όταν ένα έργο αυξηθεί αρκετά σε μέγεθος η C# επιτρέπει την διάσπαση του σε μικρότερα αρχεία χωρίς περιορισμούς για το που θα βρίσκονται ή πως θα ονομάζονται. Κατά την μεταγλώτιση(compile) της C#, μπορείτε να φανταστείτε οτι όλα τα αρχεία αυτά ενώνονται σε ένα κοινό αρχείο.

Συμβατότητα:

Τέσσερα είδη τύπων από APIs(Application Protocol Interface) είναι κοινά στις πλατφόρμες Windows και η C# τα υποστηρίζει όλα.

2.1.1.3 Εξέλιξη της C#

Η C# από τότε που δημιουργήθηκε (έκδοση C# 1.0) μέχρι σήμερα έχει αναβαθμιστεί άλλες τρείς φορές. Η αμέσως επόμενη έκδοση από την αρχική ήταν η C# 1.1 η οποία δεν είχε πολλές σοβαρές διαφορές από την προηγούμενη.

Η επόμενη αναβάθμιση της C# είναι η C# 2.0 όπου αποτελεί μια σηματνική έκδοση καθώς είχαν ενσωματωθεί αρκετά νέα χαρακτηριστικά. Μαζί με μερικές μικρές βελτιώσεις η C# 2.0 έφερε τα εξής 14 νέα χαρακτηριστικά:

- Generics
- Nullable Types
- Iterators
- Partial Class Definition
- Anonymous methods
- The :: operator

- Static classes
- Covariance and contravariance
- Fixed-size buffers
- Friend assemblies
- Extern aliases
- Method group conversions
- Accessor access control
- New #pragma directives

Από αυτά τα νέα χαρακτηριστικά αυτό που μπορούμε να πούμε οτι είναι το πιο σημαντικό και το οποίο έχει άμεσο αντίκτυπο στους προγραμματιστές είναι τα Generics.

Άλλη μια αναβάθμιση που έχει γίνει στην C# είναι η C# 3.0 (προφέρεται ως «C sharp Orkas»), η οποία περιέχει αρκετές προεκτάσεις βασισμένες στην προηγούμενη έκδοση, οι οποίες και είναι:

- Implicitly typed local variables
- Extension methods
- Lambda expressions
- Object initializers
- Anonymous types
- Implicitly typed arrays
- Query expressions
- Expression trees

Δεν θα αναφερθούμε περαιτέρω στο ζήτημα των εκδόσεων της C# καθώς δεν είναι αυτός ο σκοπός της εργασίας.

2.1.2 ADO.NET

Σχεδόν κάθε εφαρμογή βασίζεται σε δεδομένα, τα οποία συνήθως συγκεντρώνονται σε κάποια Βάση δεδομένων όπως ο SQL Server, Oracle ή DB2. Με την τεχνολογία του .NET μπορούμε να έχουμε πρόσβαση στην πληροφορία αυτή χρησιμοποιώντας την τελευταία τεχνολογία της Microsoft, το ADO.NET.

Όπως και με τις υπόλοιπες τεχνολογίες του .NET, έτσι και αυτή η τεχνολογία έχει βασιστεί στον προκάτοχο της, το ADO, με την διαφορά οτι το ADO.NET περιέχει κάποιες σημαντικές αλλαγές και μερικές καινοτομίες. Περιέχει ένα αποσυνδεδεμένο προγραμματιστικό μοντέλο (disconnected programming model) προσαρμοσμένο για κατανεμημένες εφαρμογές και για το Διαδίκτυο, ενσωματωμένη υποστήριξη XML serialization, πρακτική συνδεσιμότητα δεδομένων και ένα εκτεταμένο πακέτο από διασυνδέσεις (interfaces) που επιτρέπουν την δημιουργία κατά παραγγελία παρόχους δεδομένων.

2.1.3 Βάσεις Δεδομένων

Στην ενότητα αυτή θα μιλήσουμε για τους τύπους των βάσεων δεδομένων που έχουμε χρησιμοποιήσει. Για την δομή της βάσης δεδομένων έχουμε ήδη αναφερθεί σε προηγούμενη ενότητα οπότε τώρα θα αναφερθούμε στα τεχνικά χαρακτηριστικά τους.

2.1.3.1 Microsoft SQL Server

Ο Microsoft SQL Server είναι μια σχεσιακή βάση δεδομένων που έχει δημιουργηθεί από την εταιρεία Microsoft. Η βασική γλώσσα ερωτημάτων που χρησιμοποιεί είναι η Transact-SQL, μια υλοποιήση της ANSI/ISO standard Structured Query Language (SQL). Ο SQL Server χρησιμοποιείται ευρέως από επιχειρήσεις μικρές και μεσαίες εφαρμογές, αλλά τα τελευταία χρόνια έχει παρατηρηθεί η χρήση του και σε μεγαλύτερης κλίμακας εφαρμογές.

Εργαλεία του SQL Server που χρησιμοποιούνται συχνά είναι ο Enterprise Manager, ο Query Analyzer, ο SQL Profiler, ο Service Manager και τα Data Transformation Services. Ας αναλύσουμε ένα ενα τα εργαλεία αυτά:

- Enterprise Manager

Είναι το βασικό εργαλεία διαχείρισης του SQL Server, καθώς παρέχει ένα γραφικό περιβάλλον από το οποίο μπορούμε να διαχειριστούμε τις βάσεις δεδομένων που υπάρχουν. Μέσω του εργαλείου αυτού μπορούμε να προβούμε σε υψηλού επιπέδου ενέργειες που αφορούν έναν μόνο ή περισσότερους servers.

- Query Analyzer

Προσφέρει ένα γρήγορο τρόπο εκτέλεσης ερωτημάτων σε οποιαδήποτε βάση δεδομένων του SQL Server. Με το εργαλείο αυτό μπορείς να ανακτήσεις πολύ γρήγορα την πληροφορία που θες ή να δοκιμασεις αν λειτουργεί σωστά ένα ερώτημα πριν το δημιουργήσεις.

- SQL Profiler

Είναι ένα εργαλείο το οποίο μας παρέχει την δυνατότητα παρακολούθησεις των ενεργειών που γίνονται στην βαση δεδομένων που θέλουμε να παρακολου-θήσουμε. Μπορούμε να παρακολουθήσουμε διάφορα είδη γεγονότων σε πραγματικό χρόνο. Είναι ένα πολύ καλό εργαλείο για την βελτιστοποίηση της Βάσης δεδομένων μας.

- Service Manager

Είναι εργαλείο το οποίο διαχειρίζεται τις διεργασίες του MSSQLSERVER(η βασική του διαδικασία), την διεργασία MSDTC (Microsoft Distributed Transaction Coordinator) και την διεργασία SQLServerAgent. Συνήθως έχουμε πρόσβαση στο εργαλείο αυτό από ένα εικονίδιο που εμφανίζεται στο system tray του λειτουργικού συστήματος. Από εκεί μπορούμε να ξεκινήσουμε ή να διακόψουμε την διεργασία του SQL Server.

- Data Transformation Services

Παρέχει ένα πολύ καλό τρόπο για την εισαγωγή και εξαγωγή δεδομένων ανάμεσα στον SQL Server και σε μια μεγάλη ποικιλία από τύπους αρχείων.

2.1.3.2 Microsoft Access

Η Microsoft Office Access γνωστή και ως Microsoft Access, είναι μια σχεσιακή βάση δεδομένων που έχει υλοποιηθεί από την εταιρεία Microsoft. Συνδυάζει την μηχανή βάσης Microsoft Jet με ένα γραφικό περιβάλλον και ανήκει στο σύστημα Microsoft Office. Η Microsoft Access μπορεί να χρησιμοποιηθεί στην ανάπτυξη εφαρμογών, αλλά και στην απλή καταγραφή δεδομένων. Υποστηρίζει κάποιες αντικειμενοστραφής τεχνικές, αλλα δεν μπορούμε να πούμε οτι ανήκει σε αυτή τη κατηγορία. Η Microsoft Access χρησιμοποιείται από μικρές επιχειρήσεις, από τμήματα μεγάλων επιχειρήσεων και από χομπίστες προγραμματιστές για να δημιουργήσουν εφαρμογές εκείνη τη στιγμή(Ad-hoc).

Ένα από τα πλεονεκτήματα της Access είναι η σχέση της με την SQL, όπου τα ερωτήματα που περιέχει μπορούν να προβληθούν και να επεξεργαστούν σας δηλώσεις SQL. Οι χρήστες της μπορούν να χρησιμοποιήσουν την VBA και τα Macros για να προγραμματίσουν φόρμες και λογική.

Αντίθετα με σχεσιακή βάση δεδομένων(RDBMS), η Access δεν υποστηρίζει χαρακτηριστικά όπως triggers και stored procedures. Μόνο στην έκδοση της Access 2000(Jet 4.0) υπάρχει σύνταξη για την δημιουργία ερωτημάτων με παραμέτρους, με έναν τρόπο ο οποίος μοιάζει με την δημιουργία stored procedure.

Όλα τα εργαλεία που χρησιμοποιούμε για την Access είναι ενσωματομένα και μερικά από αυτά είναι ο Query Designer Grid και ο SQL Editor.

2.1.4 Internet Information Services (IIS)

Ο Microsoft Internet Information Services, γνωστός και ως IIS για λόγους συντομίας, είναι ένα σύνολο από διαδικτυακές διεργασίες για υπολογιστές τύπου server και οι οποίοι χρησιμοποιούν λειτουργικά συστήματα της εταιρείας Microsoft. Οι υπηρεσίες που περιέχει ένας IIS είναι FTP, SMTP, NNTP και HTTP/HTTPS. Η εφαρμογή που έχουμε υλοποιήσει χρησιμοποιεί τον HTTP server του IIS. Η εφαρμογή αναπτύχθηκε σε υπολογιστή με λειτουργικό σύστημα Windows XP Professional, το οποίο σημαίνει οτι χρησιμοποιήσαμε τον IIS 5.1. Η έκδοση του IIS που χρησιμοποιήσαμε δεν δημιουργεί κάποιο θέμα και η εφαρμογή μας μπορεί να φιλοξενηθεί και από άλλες εκδόσεις του IIS.

Ο IIS έχει ρυθμιστεί για μέγιστη ασφάλεια του διακομιστή web και με κάθε έκδοση του βελτιώνεται. Οι υπηρεσίες IIS παρέχουν την πιο αξιόπιστη, παραγωγική, συνδεδεμένη και ενοποιημένη λύση διακομιστή Web με ανοχή σφαλμάτων, ουρά αιτήσεων, παρακολούθηση κατάστασης εφαρμογών, αυτόματη ανακύκλωση εφαρμογών, προσωρινή αποθήκευση και άλλα.

3. Λειτουργική Περιγραφή της Εφαρμογής

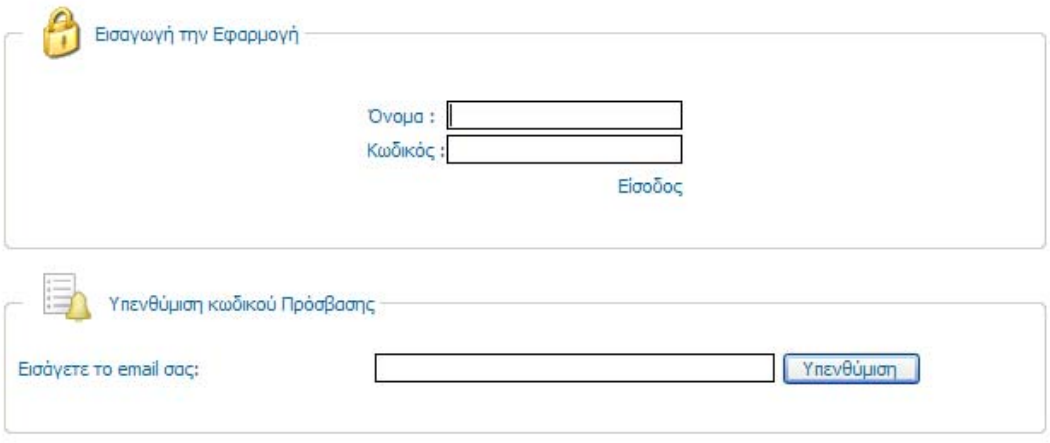
3.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφονται οι λειτουργίες της εφαρμογής για την παραγωγή και προβολή των στατιστικών δεδομένων.

Η διεπαφή που έχει δημιουργηθεί σχεδιάστηκε με γνώμονα τους χρήστες που θα την χρησιμοποιούν. Στο τομέα της Ανάπτυξης Λογισμικού, η εμπείρια των ανθρώπων έχει δείξει οτι ένα από τα βασικά προβλήματα που έχουμε να αντιμετωπίσουμε από την χρήση μιας νέας εφαρμογής, είναι η αποδοχής της από τους χρήστες. Όπως έχουμε αναφέρει η εφαρμογή αυτή δημιουργήθηκε για να υποστηρίξει την απαίτηση του ιατρικού και νοσηλευτικού προσωπικού για την παρακολούθηση στατιστικών δεδομένων (δεικτών) της κλινικής της Χειρουργικής Ογκολογίας. Το προσωπικό αυτό έχει οριστεί σαν βασικοί χρήστες της εφαρμογής. Οι χρήστες αυτοί, λόγω του χώρου εργασίας (ΠΕ.ΠΑ.Γ.Ν.Η), έχουν μάθει την χρήση του ηλεκτρονικού υπολογιστή μέσα από λειτουργικά συστήματα της Microsoft. Για τον λόγο αυτό θέλαμε το περιβάλλον εργασίας της εφαρμογής να θυμίζει αυτό που έχουν μάθει ως τώρα.

3.2 Εισαγωγική Σελίδα

Η εισαγωγική σελίδα είναι η σελίδα που παρατίθεται παρακάτω (Εικόνα 1) και στην οποία καλείται ο χρήστης να δώσει τα στοιχεία του για την εισαγωγή του στην εφαρμογή. Τα στοιχεία αυτά είναι το όνομα χρήστη (username) και τον κωδικό πρόσβασης (password). Πατώντας της επιλογή «Είσοδος» εκτελείται μια διαδικασία που ελέγχει τα στοιχεία του χρήστη και επιστρέφει ένα αποτέλεσμα για το αν ο χρήστης γίνεται αποδεκτός απο την εφαρμογή καθώς επίσης τον ρόλο του και τα δικαιώματα που έχει.



Εικόνα 3. Εισαγωγική Σελίδα

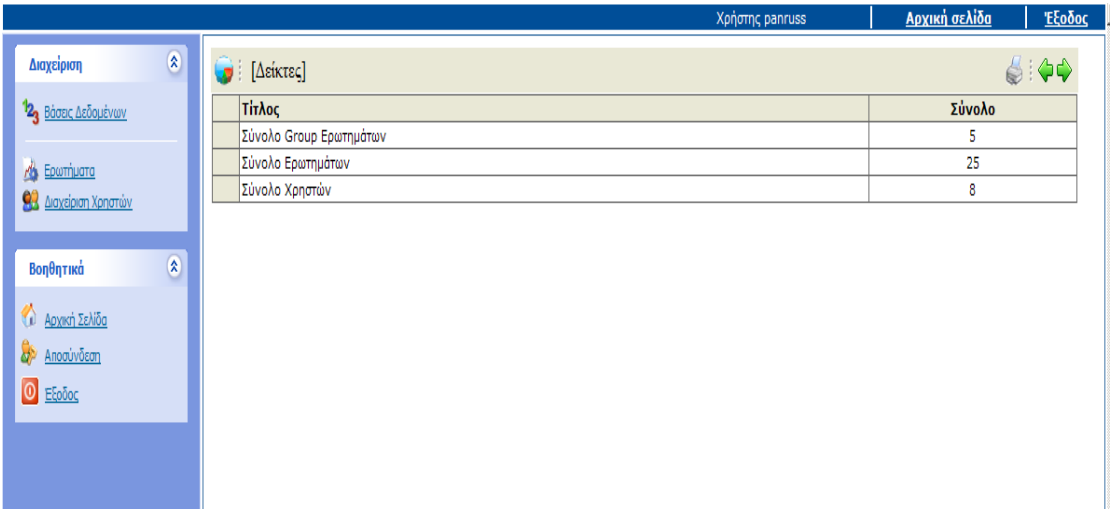
Αν ο χρήστης γνωρίζει την διεύθυνση οποιαδήποτε σελίδα της εφαρμογής και την πληκτρολογήσει κατευθείαν, αν δεν έχει πιστοποιηθεί οτι ανήκει σε αυτόν τον δικτυακό τόπο, τότε δεν θα του επιτραπεί η προβολή της σελίδας αυτής και θα του ζητηθούν τα στοιχεία εισόδου του.

Όπως μπορούμε να δούμε από την εικόνα, παρέχεται άλλη μία υπηρεσία η οποία είναι η «*Υπενθύμιση του Κωδικού Πρόσβασης*». Με την επιλογή αυτή, δίνοντας το email που έχουμε καταχωρήσει κατά την εγγραφή μας στην εφαρμογή, μας αποστέλλει η εφαρμογή ενημερωτικό email με τα στοιχεία είσόδου μας. Όπως καταλαβαίνετε ο

τρόπος αυτός υπενθύμισης του κωδικού πρόσβασης είναι αρκετά ασφαλής, καθώς τα στοιχεία εισόδου αποστέλλονται στον λογαριασμό του ηλεκτρονικού μας ταχυδρομείου, στο οποίο και εκεί απαιτείται πιστοποίηση του χρήστη.

3.3 Κύρια Σελίδα

Αφού πληκτρολογήσουμε τα στοιχεία πρόσβασης για την είσοδο μας στην εφαρμογή και πιστοποιηθούμε σαν χρήστες του δικτυακού χώρου, η εφαρμογή μάς παραπέμπει στην αρχική της σελίδα, η οποία και φαίνεται στην εικόνα που ακολουθεί (Εικόνα 2)



Εικόνα 4. Κύρια Σελίδα

Η επικεφαλίδα της κύριας σελίδας περιλαμβάνει πληροφορίες σχετικά με το όνομα χρήστη (username) με το οποίο έχουμε συνδεθεί, καθώς επίσης περιέχει και δύο συνδέσμους οι οποίοι μας παραπέμπουν στην αρχική σελίδα και στην έξοδο μας από την εφαρμογή.

Στο κεντρικό μέρος της κύριας σελίδας περιλαμβάνει κάποιες πληροφορίες σχετικά με την λειτουργία της εφαρμογής, όπως για παράδειγμα το σύνολο των Group Ερωτημάτων, το σύνολο των Ερωτημάτων και στο σύνολο χρηστών που έχουν δημιουργηθεί, οι έννοιες των οποίων θα αναλυθούν στην πορεία. Οι παραπάνω πληροφορίες εμφανίζονται πάντα βάσει του ρόλου και του επιπέδου πρόσβασης του χρήστη. Ο διαχειριστής της εφαρμογής για παράδειγμα μπορεί να δει το σύνολο όλων των ερωτημάτων που έχουν δημιουργηθεί στην εφαρμογή, σε αντίθεση με τον συντάκτη ερωτημάτων ο οποίος μπορεί να δεί το σύνολο των ερωτημάτων τα οποία έχει συντάξει.

Όπως αναφέραμε στην εισαγωγή, θέλαμε το περιβάλλον εργασίας να είναι όσο το δυνατόν πιο οικείο στους χρήστες της εφαρμογής και όπως μπορείτε να παρατηρήσετε θυμίζει το περιβάλλον εργασίας του λειτουργικού συστήματος Microsoft Windows XP.

Στο αριστερό της κύριας σελίδας μέρος βρίσκεται το menu με το οποίο μπορούμε να περιηγηθούμε μέσα στην εφαρμογή επιλέγοντας τους κατάλληλους συνδέσμους. Υπάρχουν δύο βασικά Menu τα οποία είναι:

- Διαχείριση

Από το menu «Διαχείριση» ουσιαστικά γίνοται όλες οι λειτουργίες της εφαρμογής, οι οποίες είναι:

- Βάσεις Δεδομένων
- Ερωτήματα
- Διαχείριση Χρηστών

Θα αναφερθούμε αναλυτικά σε επόμενες ενότητες.

- Βοηθητικά

Στο menu «Βοηθητικά» υπάρχουν κάποιες βασικές ενέργειες, οι οποίες είναι:

- Μετάβαση στην αρχική σελίδα
- Αποσύνδεση από την εφαρμογή
- Έξοδος από την εφαρμογή

Σε αυτό το σημείο να διευκρινήσουμε μόνο την διαφορά της «Αποσύνδεσης» και της «Εξόδου» από το σύστημα. Κατά τα άλλα θεωρούμε οτι οι λειτουργίες αυτές είναι καταννοητές. Με την αποσύνδεση ενημερώνουμε τον server οτι θέλουμε να λήξουμε τις εργασίες με τον συγκεκριμένο χρήστη με τον οποίο έχουμε συνδεθεί και να μας μεταβιβάσει στην αρχική σελίδα της εφαρμογής σε αντίθεση με την έξοδο με την οποία ενημερώνουμε τον server οτι θέλουμε να λήξουμε τις εργασίες με τον συγκεκριμένο χρήστη με τον οποίο έχουμε συνδεθεί και οτι θέλουμε να κλείσουμε το παράθυρο της εφαρμογής.

3.4 Menu Διαχείριση

3.4.1 Βάσεις Δεδομένων

Επιλέγοντας τον σύνδεσμο «Βάσεις Δεδομένων» εμφανίζεται μια λίστα με τις βάσεις δεδομένων που έχουν οριστεί, όπως φαίνεται στην εικόνα 3 και από τις οποίες αντλούνται τα στοιχεία για να παραχθούν τα στατιστικά δεδομένα.

[Λίστα Group Ερωτημάτων - Επεξεργασία]

Εξαγωγή Δεδομένων | Προσθήκη Νέας Γραμμής

Α/Α	Τίτλος Group	Ημ/νία Δημιουργίας	Τύπος Β.Δ.	Επεξεργασία
1	Β.Δ. StatOnLine	05/03/2007	MS SQL Server	
2	Β.Δ. ΜΗΤΡΩΟ ΚΛΙΝΙΚΗΣ ΧΟΓΚ	05/03/2007	MS Access	
3	..	03/09/2007	MS SQL Server	
4	Β.Δ. ΧΟΓΚ	12/03/2007	MS Access	
5	TCDN Xogk	03/12/2007	MS SQL Server	

Εικόνα 5. Βάσεις Δεδομένων

Στην εφαρμογή οι έννοιες «Group Ερωτημάτων» και «Βάσεις Δεδομένων» ταυτίζονται, δημιουργώντας δηλαδή μία εγγραφή για μία βάση δεδομένων, αυτομάτως δημιουργούμε το group ερωτημάτων.

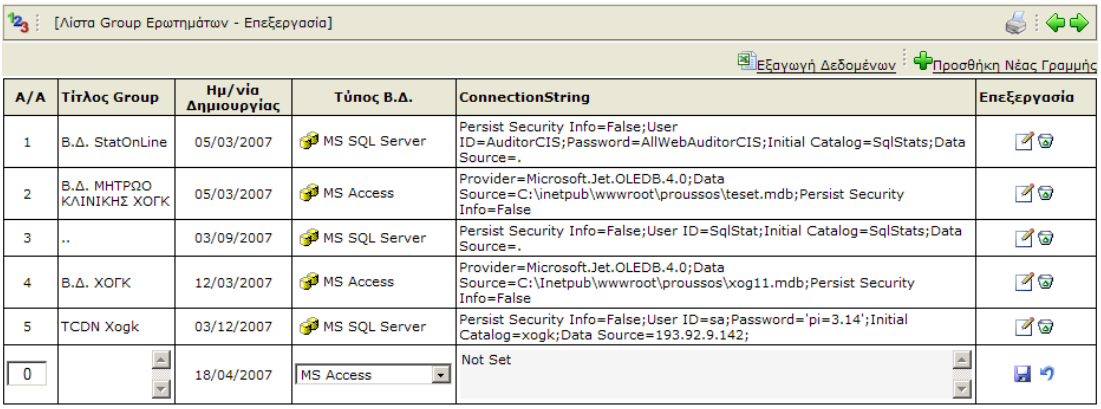
Οι ενέργειες που μπορούν να γίνουν μέσα από την φόρμα αυτή είναι οι εξής:

- Προσθήκη Νέας Βάσης Δεδομένων / Νέου Group ( )

Με την επιλογή αυτή, προστίθεται μία νέα γραμμή στην λίστα στην οποία καλείται ο χρήστης να συμπληρώσει τον τίτλο του Group με το οποίο θα εμφανίζεται, ο τύπος της Βάσης Δεδομένων (MS Access, MS SQL Server) και την τοποθεσία της Β.Δ. με τα απαραίτητα στοιχεία πρόσβασης, γνωστό και ως «ConnectionString».

- Επεξεργασία Βάσης Δεδομένων / Group ( )

Με την επιλογή αυτή ο χρήστης μπορεί να τροποποιήσει τα στοιχεία που αναφέραμε παραπάνω. Όταν βρισκόμαστε σε κατάστασης επεξεργασίας, στην συγκεκριμένη γραμμή στην οποία βρισκόμαστε εμφανίζονται οι επιλογές «Αποθήκευση» και «Αναίρεση», όπως φαίνεται στην εικόνα 4.



Εικόνα 6. Επεξεργασία Βάσης Δεδομένων / Group

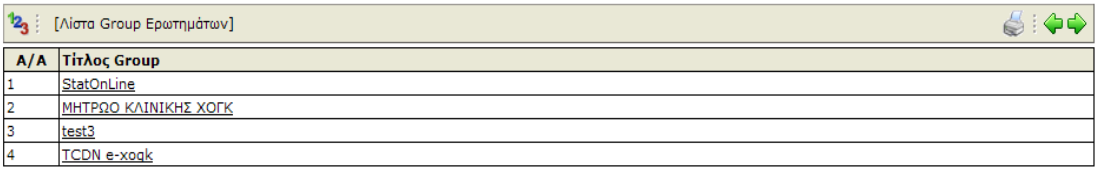
- Διαγραφή Βάσης Δεδομένων / Group ( )

Με την επιλογή αυτή ο χρήστης μπορεί να διαγράψει το group ερωτημάτων καθώς επίσης και όλα τα ερωτήματα τα οποία ανήκουν στο group αυτό.

3.4.2 Ερωτήματα

3.4.2.1 Λίστα Group Ερωτημάτων

Επιλέγοντας τον σύνδεσμο «Ερωτήματα» εμφανίζεται μία λίστα με τα groups των ερωτημάτων που είναι διαθέσιμα, όπως φαίνεται στην Εικόνα 5, πάντα βάσει του ρόλου μας στην εφαρμογή. Η λίστα αυτή είναι ακριβώς ίδια με την λίστα των βάσεων δεδομένων που περιγράψαμε στην προηγούμενη ενότητα με την διαφορά οτι εδώ δεν μπορούμε να επεξεργαστούμε τα στοιχεία της.



Εικόνα 7. Λίστα Group Ερωτημάτων

Επιλέγοντας τον τίτλο του group μεταφερόμαστε στην λίστα των ερωτημάτων που ανήκουν στο group αυτό.

3.4.2.2 Λίστα ερωτημάτων

Η λίστα ερωτημάτων περιέχει τα ερωτήματα που ανήκουν στο group που επιλέξαμε. Από την εικόνα 6 μπορούμε να δούμε οτι η λίστα περιλαμβάνει κάποιες βασικές πληροφορίες για το ερώτημα, όπως τον τίτλο του ερωτήματος, την ημερομηνία δημιουργίας του και τον χρήστη δημιουργίας του ερωτήματος.

[Λίστα Ερωτημάτων]

Εξαγωγή Δεδομένων | Εκτέλεση Group | Προσθήκη Νέας Γραμμής

Α/Α	Τίτλος Ερωτήματος	Ημ/νία Δημιουργίας	Χρήστης Δημιουργίας	Επεξεργασία
1	Πληροφοριακά Στοιχεία	03/12/2007	Admin	
2	Αριθμός και ποσοστό ασθενών που έχουν κλικαρισμένο το πεδίο "πραγματοποιήθηκε συμπληρωματική θεραπεία" στη φόρμα συμπληρωματική θεραπεία	03/12/2007	Admin	
3	Αριθμός και ποσοστό ασθενών που έχουν κλικαρισμένο το πεδίο "ορθό" στη φόρμα γενικά στοιχεία και το πεδίο "υποτροπή" στις επιπλοκές	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
4	ή/και σε κάποιο follow-up	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
5	Σύνολο	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
6	Πόσος ήταν ο χρόνος και σε τι ποσοστό κάθε φορά από την ημερομηνία στη φόρμα χειρουργείο μέχρι την ημερομηνία της υποτροπής στη φόρμα επιπλοκές ή το πεδίο υποτροπή στο Follow-up	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
7	Σε ασθενείς που έχουν κλικαρισμένο το πεδίο Υποτροπή στη φόρμα επιπλοκές ή/και το πεδίο υποτροπή στα Follow-up να βρεθεί ο αριθμός και το ποσοστό κάθε φορά στο πεδίο "περιφερικό-απόσταση από το χειρουργικό όριο" στη φόρμα παθολογοανατομική εξέταση	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
8	Τι ποσοστό ασθενών ήταν άντρες-γυναίκες	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	

Εικόνα 8. Λίστα Ερωτημάτων

Οι ενέργεις που μπορούν να γίνουν σε αυτή την φόρμα είναι:

- Προσθήκη νέου ερωτήματος( )

Με την επιλογή αυτή η εφαρμογή μας μεταφέρει στην φόρμα του ερωτήματος όπου εκεί μπορούμε να δημιουργήσουμε και να επεξεργαστούμε το ερώτημα. Την φόρμα αυτήν θα την αναλύσουμε παρακάτω.

- Επεξεργασία ερωτήματος( )

Με την επιλογή αυτή η εφαρμογή μας μεταφέρει στην φόρμα του ερωτήματος όπου εκεί μπορούμε να δημιουργήσουμε και να επεξεργαστούμε το ερώτημα.

- Εκτέλεση ερωτήματος( )

Με την επιλογή αυτή η εφαρμογή μας μεταφέρει στην φόρμα του ερωτήματος όπου εκεί μπορούμε να εκτελέσουμε το ερώτημα, την οποία θα αναλύσουμε στην συνέχεια.

- Εκτέλεση group ερωτημάτων

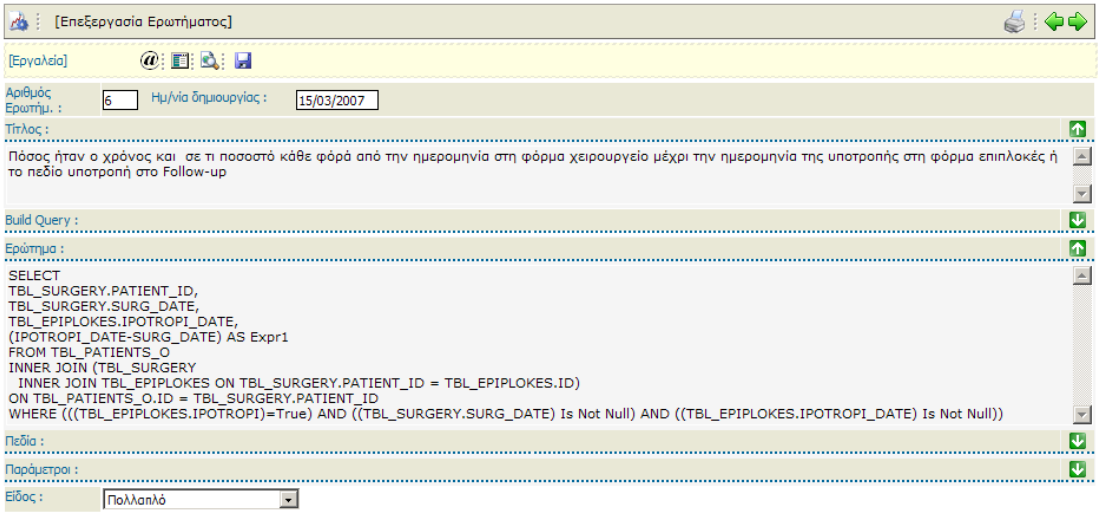
Με την επιλογή αυτή η εφαρμογή εκτελεί όλα τα ερωτήματα που ανήκουν στο group αυτό και εμφανίζει τα αποτελέσματα τους. Αν κάποιο ερώτημα έχει παραμέτρους, τότε αυτό εκτελείται με τις αρχικές παραμέτρους που έχουν οριστεί από τον χρήστη δημιουργίας του ερωτήματος.

- Διαγραφή ερωτήματος( )

Με την επιλογή αυτή ο χρήστης μπορεί να διαγράψει το ερώτημα που έχει επιλέξει.

3.4.2.3 Φόρμα Δημιουργίας και Επεξεργασίας Ερωτήματος

Η φόρμα δημιουργίας και επεξεργασίας ερωτημάτων μπορούμε να πούμε οτι είναι μία από τις πιο βασικές φόρμες της εφαρμογής καθώς έχει ένα σημαντικό ρόλο. Απλοποιεί την διαδικασία παραγωγής ερωτημάτων. Ένα τυπικό παράδειγμα της φόρμας φαίνεται στην Εικόνα 7.



Εικόνα 9. Φόρμα Δημιουργίας και Επεξεργασίας Ερωτήματος

Η φόρμα στην εικόνα 7 δεν είναι σε πλήρη ανάπτυξη για να είναι πιο φιλική στον χρήστη, καθώς περιέχει αρκετή πληροφορία. Η φόρμα αποτελείται από τα εξής μέρη:

- Εργαλεία

Η εργαλειοθήκη περιέχει τα εξής εργαλεία:

  - Αρχικοποίηση Παραμέτρων ( )

Με την επιλογή αυτή ξεκινάει μια διαδικασία αναζήτησης λέξεων μέσα στο κείμενο του ερωτήματος οι οποίες το πρώτο τους γράμμα είναι ο χαρακτήρας «@». Αφου ολοκληρωθεί η αναζήτηση αυτή, τότε η λέξεις που έχουν βρεθεί προστίθονται στην λίστα των παραμέτρων. Στην γλώσσα υπολογιστή SQL και συγκεκριμένα στον MS SQL Server, οι παραμέτροι δηλώνονται έχοντας σαν πρώτο χαρακτήρα, τον χαρακτήρα που αναφέραμε.

  - Αρχικοποίηση Πεδίων ( )

Με την επιλογή αυτή ξεκινάει μια διαδικασία αναζήτησης των λέξεων μέσα στο κείμενο του ερωτήματος οι οποίες βρίσκονται στο μέρος όπου δηλώνονται τα πεδία σε ένα ερώτημα SQL.

  - Δοκιμαστική εκτέλεση του ερωτήματος ( )

Με την επιλογή αυτή μας παραπέμπει η εφαρμογή στην φόρμα εκτέλεσης του ερωτήματος για να δοκιμάσουμε αν το ερώτημα που έχουμε δημιουργήσει λειτουργεί.

  - Αποθήκευση ( )

Αποθηκεύει τις πληροφορίες του ερωτήματος.

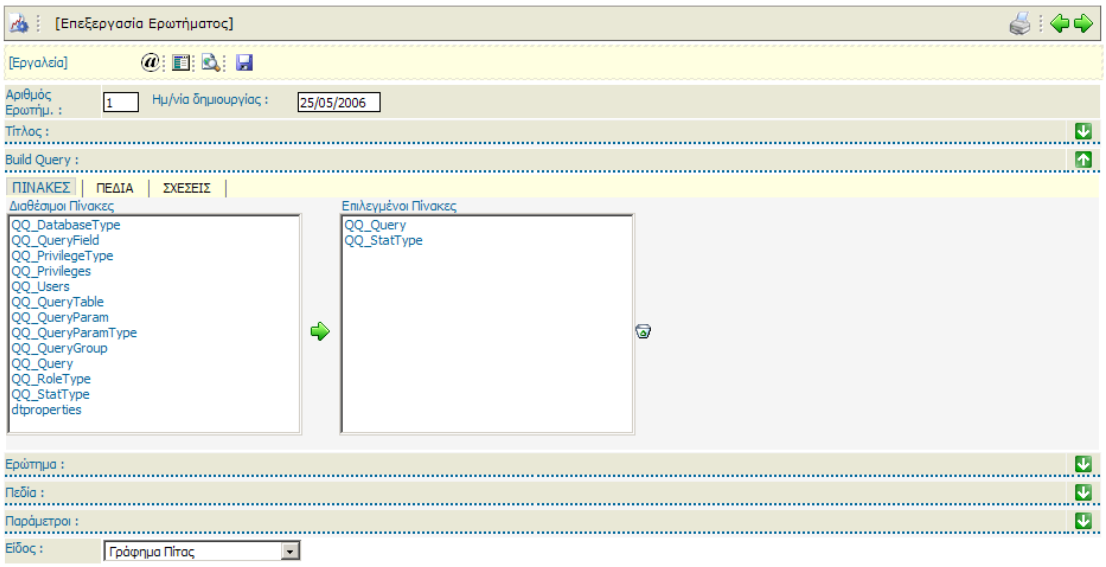
- Αριθμός ερωτήματος και Ημερομηνία δημιουργίας

Περιέχει την πληροφορία του αύξοντος αριθμού του ερωτήματος που έχουμε δηλώσει κατά την δημιουργία του και την ημερομηνία δημιουργίας του ερωτήματος.

- Τίτλος ερωτήματος
- Τον οδηγό δημιουργίας του ερωτήματος (Query Builder)

Το τμήμα αυτό είναι ένα αρκετά σημαντικό σημείο της φόρμας καθώς αυτό είναι που απλοποιεί την διαδικασία δημιουργίας του ερωτήματος SQL. Όπως φαίνεται και από

την Εικόνα 8 αποτελείται από τρία(3) μέρη, τους πίνακες, τα πεδία και τις σχέσεις των πεδίων όπου υπάρχουν.

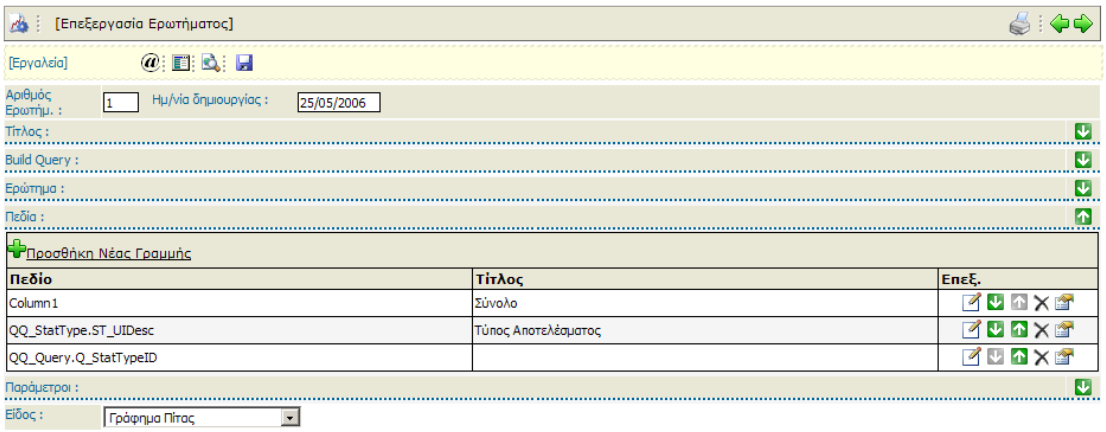


Εικόνα 10. Query Builder

Σε κάθε μέρος στο αριστερό του τμήμα βρίσκονται οι διαθέσιμες οντότητες για επιλογή και στο δεξιά τμήμα βρίσκοντα οι επιλεγμένες οντότητες. Η προσθήκη της οντότητας, η μετάβαση της δηλαδή από το «διαθέσιμο» στο «επιλεγμένο», γίνεται μέσω της επιλογής «Εισαγωγή»( ) που φαίνεται στην εικόνα 8. Με αυτόν τον τρόπο αν θέλουμε να προσθέσουμε ένα πίνακα στο ερώτημα επιλέγουμε τον πίνακα που θέλουμε από την λίστα των διαθέσιμων πινάκων και στην συνέχεια επιλέγουμε την εισαγωγή του πίνακα μέσω της επιλογής( ). Ομοίως λειτουργεί η διαδικασία της προσθήκης πεδίων και σχέσεων των πεδίων.

- Το ερώτημα (σε γλώσσα υπολογιστή - SQL)
- Λίστα πεδίων που ανήκουν στο ερώτημα.

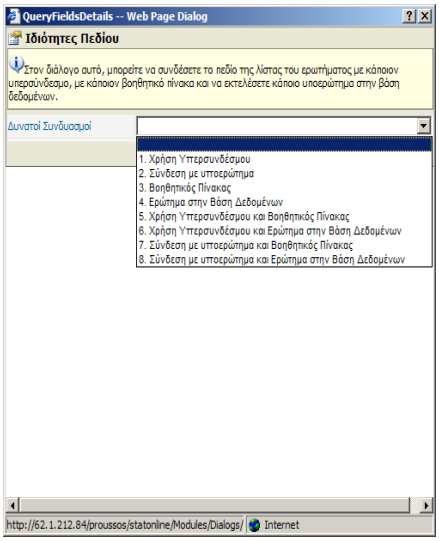
Η λίστα πεδίων περιέχει τα πεδία που έχουν δηλωθεί στο ερώτημα. Ο λόγος για τον οποίο έχει δημιουργηθεί ξεχωριστή λίστα είναι επειδή μπορούμε προσθέσουμε κάποιες ιδιότητες τις οποίες θα αναλύσουμε στην πορεία. Ένα παράδειγμα λίστας πεδίων φαίνεται στην Εικόνα 9.



Εικόνα 11. Λίστα Πεδίων

Όπως φαίνεται και από την εικόνα 9, η λίστα πεδίων περιέχει πληροφορίες για την ονομασία του πεδίου που έχει δηλωθεί και για τον τίτλο του. Μπορούμε να επεξεργαστούμε τα στοιχεία αυτά με την επιλογή «Επεξεργασία» ( ). Επίσης

μπορούμε να αλλάξουμε την σειρά εμφάνισης του πεδίου με τις επιλογές «Μετακίνηση Πάνω» () και «Μετακίνηση Κάτω» () καθώς και να διαγράψουμε το πεδίο από την λίστα με την επιλογή «Διαγραφή» (). Οι ιδιότητες που μπορούμε να προσθέσουμε στο πεδίο γίνεται μέσω της επιλογής «Ιδιότητες» (). Με την επιλογή αυτή η εφαρμογή μας παραπέμπει σε έναν διάλογο, ο οποίος φαίνεται στην εικονα 10 με τις εξής ιδιότητες:



1. Χρήση Υπερσυνδέσμου
Με την επιλογή αυτή εμφανίζονται δύο πεδία, ο «Σύνδεσμος» και ο «Στόχος». Στο πεδίο «Σύνδεσμος» δηλώνουμε την διεύθυνση που θέλουμε και στο πεδίο «Στόχος», γνωστό και ως target, το σημείο που θέλουμε να ανοίξει η νέα διεύθυνση μετά την επιλογή του. Αποτέλεσμα της επιλογής αυτής είναι να δημιουργηθεί ένας σύνδεσμος με τις παραμέτρους που έχουμε ορίσει.

2. Σύνδεση με υποερώτημα(Master/Detail)
Με την επιλογή αυτή εμφανίζονται δύο πεδία, το «Υποερώτημα» και το «Πεδίο Σύνδεσης». Σκοπός της επιλογής αυτής είναι να συνδέσει τα δύο αυτά ερωτήματα και σε κάθε γραμμή της λίστας να εμφανίσει τα δεδομένα του

Εικόνα 12. Διάλογος Ιδιοτήτων Πεδίων

υποερωτήματος που αφορούν την γραμμή. Δημιουργούμε δηλαδή μία λίστα με master/detail δεδομένα. Στο πεδίο «Υποερώτημα» δηλώνουμε το ερώτημα που θέλουμε να συνδέσουμε και στο «Πεδίο Σύνδεσης» δηλώνουμε το πεδίο με το οποίο θα συσχετιστεί το πεδίο του ερωτήματος μας.

3. Βοηθητικός Πίνακας
Με την επιλογή αυτή εμφανίζονται τρία πεδία, ο «Βοηθητικός Πίνακας», το «Πεδίο Σύνδεσης» και το «Πεδίο Επιστροφής». Σκοπός της επιλογής αυτής είναι να εμφανίσει τιμές από ένα ορισμένο άλλο πεδίο, άλλου πίνακα βάσει της τιμής που έχει. Για να γίνει πιο καταννοητό να φέρουμε το εξής παράδειγμα. Λόγω της κανονικοποίησης που εφαρμόζεται στον σχεδιασμό μίας βάσης δεδομένων μπορεί ένας πίνακας να αποτελείται μόνο από πεδία τύπου ακεραίου και οι τιμές αυτών των πεδίων να βρίσκονται σε βοηθητικούς πίνακες. Ας υποθέσουμε οτι έχουμε τον πίνακα των ασθενών(PATIENTS) και πεδίο του πίνακα είναι το ταμείο ασφάλισης τους(INSURANCE_TYPE). Το πεδίο INSURANCE_TYPE είναι τύπου ακεραίου και οι πραγματικες του τιμές βρίσκονται στον βοηθητικό πίνακα «INSURANCES». Για να αποφύγουμε την δημιουργία σχέσεων πινάκων στο ερώτημα μας, μπορούμε να χρησιμοποιήσουμε την ιδιότητα αυτήν για να προβάλουμε τις τιμές αυτές.

4. Ερώτημα στην Βάση Δεδομένων
Η επιλογή αυτή είναι ίδια με αυτήν του βοηθητικού πίνακα με την διαφορά οτι τώρα αντί για βοηθητικό πίνακα χρησιμοποιούμε ένα ερώτημα που γράφουμε προς την βάση δεδομένων.

5. Χρήση Υπερσυνδέσμου και Βοηθητικός Πίνακας
Η επιλογή αυτή είναι ένας συνδυασμός των παραπάνω. Προβάλουμε τιμές δηλαδή από έναν βοηθητικό πίνακα και επίσης δημιουργούμε και έναν σύνδεσμο.

6. Χρήση Υπερσυνδέσμου και Ερώτημα στην Βάση Δεδομένων
Ομοίως με το παραπάνω

7. Σύνδεση με υποερώτημα και Βοηθητικός Πίνακας
Η επιλογή αυτή λειτουργεί ως εξής: Προβάλουμε τιμές από έναν βοηθητικό πίνακα και δημιουργουμε έναν σύνδεσμο στο υποερώτημα που έχουμε δηλώσει.

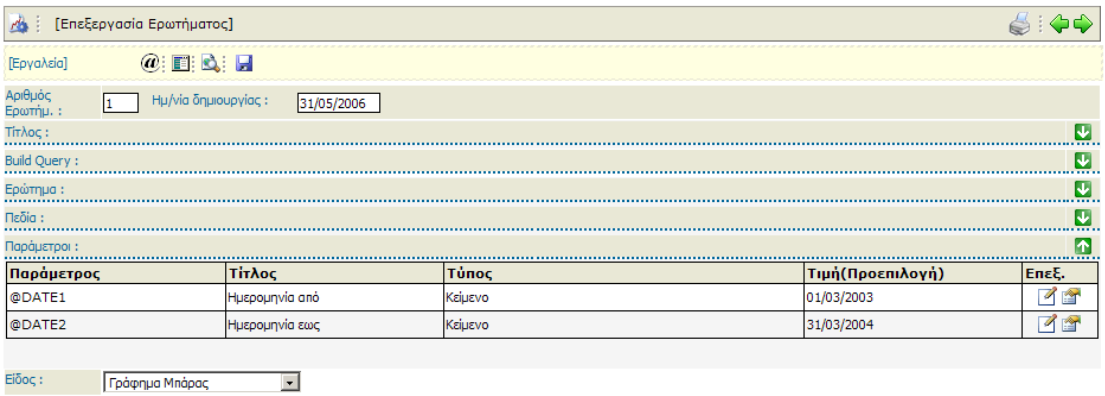
8. Σύνδεση με υποερώτημα και Ερώτημα στην Βάση Δεδομένων

Ομοίως με το παραπάνω.

Στο σημείο αυτό να σημειώσουμε οτι οι παραπάνω ιδιότητες έχουν νόημα μόνο στην προβολή του αποτελέσματος του ερωτήματος σαν πολλαπλό(Λίστα), στην οποία θα αναφερθούμε στην πορεία.

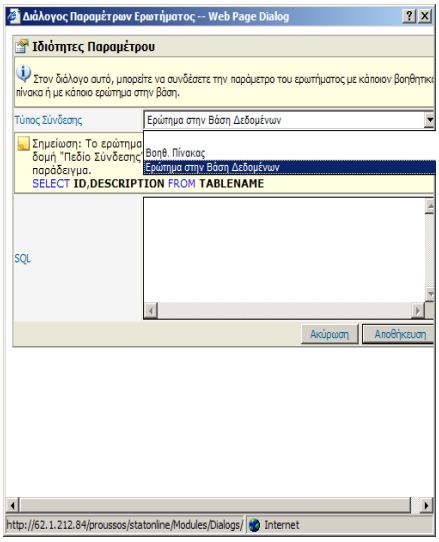
- Λίστα με τις παραμέτρους

Η λίστα παραμέτρων περιέχει τις παραμετρους που έχουν δηλωθεί στο ερώτημα. Από την εικόνα 11 φαίνεται οτι η λίστα των παραμέτρων περιέχει τις πληροφορίες όπως το όνομα της παραμέτρου, ο τίτλος της , ο τύπος και η αρχική της τιμή που θέτει ο χρήστης. Όλα τα παραπάνω πεδία μπορούν επεξεργαστούν με την επιλογή «Επεξεργασία» ( ).



Εικόνα 13. Λίστα Παραμέτρων

Όπως στην λίστα των πεδίων έτσι και εδώ οι παράμετροι έχουν κάποιες ιδιότητες. Οι ιδιότητες που μπορούμε να προσθέσουμε στις παραμέτρους γίνεται μέσω της επιλογής «Ιδιότητες» ( ) και ανοίγει ένας διάλογος όπως φαίνεται στην εικόνα 12 με τις εξής επιλογές:



1. Βοηθητικός Πίνακας

Η λειτουργία της επιλογής αυτή είναι όμοια με την λειτουργία που αναφέραμε στην λίστα πεδίων.

2. Ερώτημα στην βάση δεδομένων

Η λειτουργία της επιλογής αυτή είναι όμοια με την λειτουργία που αναφέραμε στην λίστα πεδίων.

Εικόνα 14. Ιδιότητες Παραμέτρου

- Το είδος της απεικόνισης του αποτελέσματος

Η εφαρμογή μπορεί ουσιαστικά να απεικονίσει τα στατιστικά δεδομένα σε τρεις μορφές, οι οποίες είναι:

  - Πολλαπλό

Το πολλαπλό αποτέλεσμα είναι ουσιαστικά μια λίστα με το αποτέλεσμα του ερωτήματος, βασει των επιλογών που έχουμε κάνει κατά την δημιουργία του ερωτήματος.

- Γράφημα

Η εφαρμογή μπορεί να προβάλει τριών ειδών γραφήματα

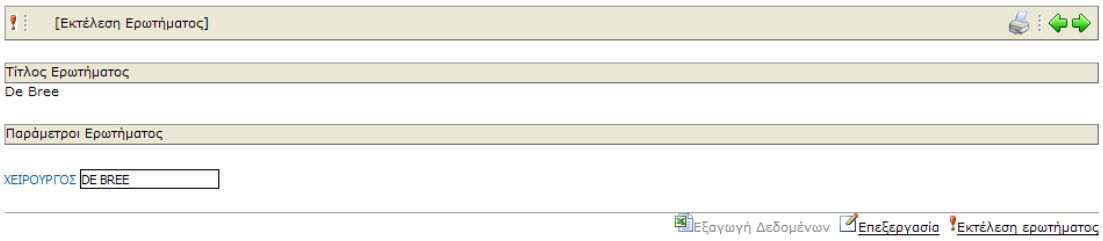
- ❖ Γράφημα Πίτας
- ❖ Γράφημα Μπάρας
- ❖ Γράφημα Καμπύλης

- Απλό

Το απλό αποτέλεσμα είναι η προβολή μίας μόνο τιμής του αποτελέσματος, όπως για παράδειγμα ένα αριθμητικό αποτέλεσμα.

3.4.2.4 Φόρμα Εκτέλεσης Ερωτημάτων

Η φόρμα εκτέλεσης ερωτημάτων είναι η φόρμα στην οποία προετοιμαζόμαστε για την εκτέλεση του ερωτήματος, όπως φαίνεται στην εικόνα 13.



Εικόνα 15. Φόρμα Εκτέλεσης Ερωτημάτων

Η φόρμα περιέχει πληροφορίες όπως τον τίτλο του ερωτήματος και τις παραμέτρους του, αν έχουν τεθεί. Αν δεν έχουν τεθεί παράμετροι εμφανίζεται μήνυμα που ενημερώνει το χρήστη οτι δεν υπάρχουν. Οι παράμετροι εμφανίζονται με τις αρχικές τους τιμές που έχει θέσει ο χρήστης κατά την δημιουργία του ερωτήματος. Σε αυτήν την φόρμα μπορούμε να τροποποιήσουμε τις αρχικές τιμές των παραμέτρους πριν εκτελέσουμε το ερώτημα.

Η ενέργειες που μπορούν να γίνουν από την φόρμα είναι:

- Εκτέλεση ερωτήματος

Εκτελείται το ερώτημα. Αν υπάρχουν παράμετροι στο ερώτημα, λαμβάνονται υπόψη αυτοί που έχουν δηλωθεί στα πεδία των παραμέτρων και όχι οι αρχικοί παράμετροι που έχει θέσει ο χρήστης κατά την δημιουργία του.

- Επεξεργασία

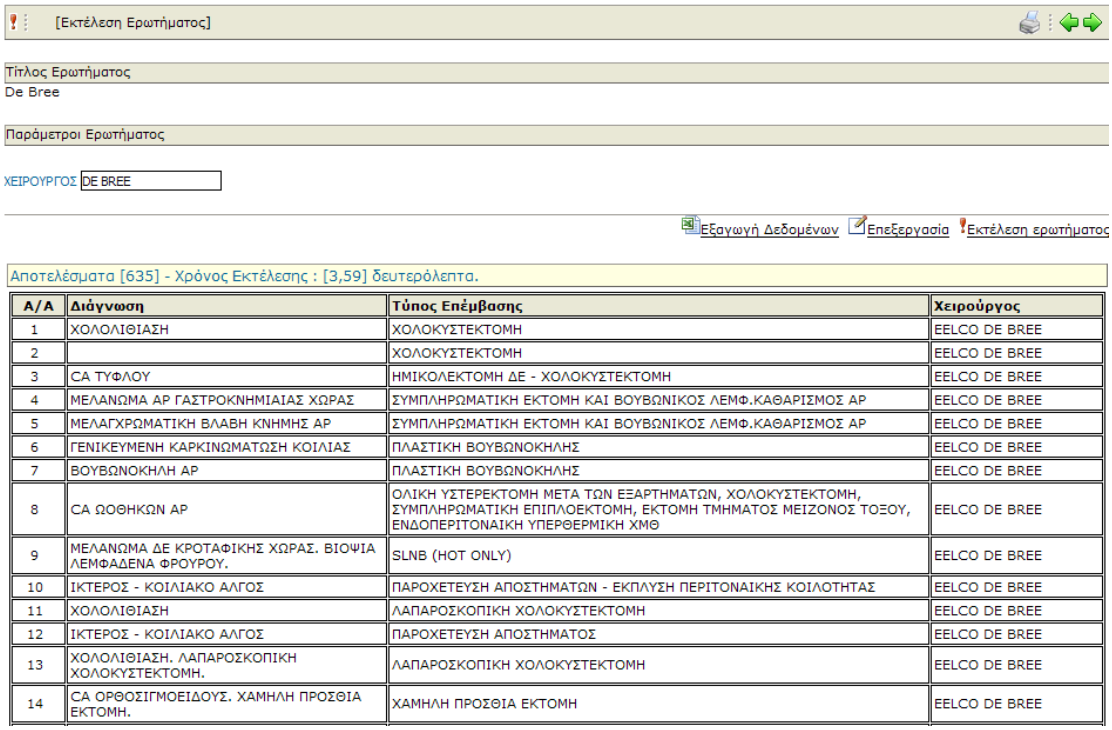
Η εφαρμογή μας μεταφέρει στην φόρμα επεξεργασίας του ερωτήματος που περιγράψαμε στην προηγούμενη ενότητα.

- Εξαγωγή δεδομένων

Η ενέργεια αυτή ενεργοποιείται μόνο σε περίπτωση που το αποτέλεσμα έχει δηλωθεί να εμφανίζεται σε λίστα. Τότε μπορούμε να εξάγουμε τα δεδομένα στο Microsoft Excel.

Ένα χαρακτηριστικό το οποίο έχουμε προσθέσει είναι η απεικόνιση του αριθμού των αποτελεσμάτων του ερωτήματος και ο χρόνος εκτέλεσης του. Το χαρακτηριστικό αυτό εμφανίζεται μόνο σε αποτελέσμα τύπου λίστας(πολλαπλό) και τύπου απλού. Ο σκοπός του είναι να μας ενημερώσει για το πλήθος των αποτελέσματων του ερωτήματος, αλλά να μας δώσει και μια εικόνα για το βαθμό δυσκολίας εκτέλεσης του συγκεκριμένου ερωτήματος.

Ένα παράδειγμα εκτέλεσης ερωτήματος φαίνεται στην εικόνα 14.


[Εκτέλεση Ερωτήματος]

Τίτλος Ερωτήματος
De Bree

Παράμετροι Ερωτήματος

ΧΕΙΡΟΥΡΓΟΣ DE BREE

Εξαγωγή Δεδομένων | Επεξεργασία | Εκτέλεση ερωτήματος

Αποτελέσματα [635] - Χρόνος Εκτέλεσης : [3,59] δευτερόλεπτα.

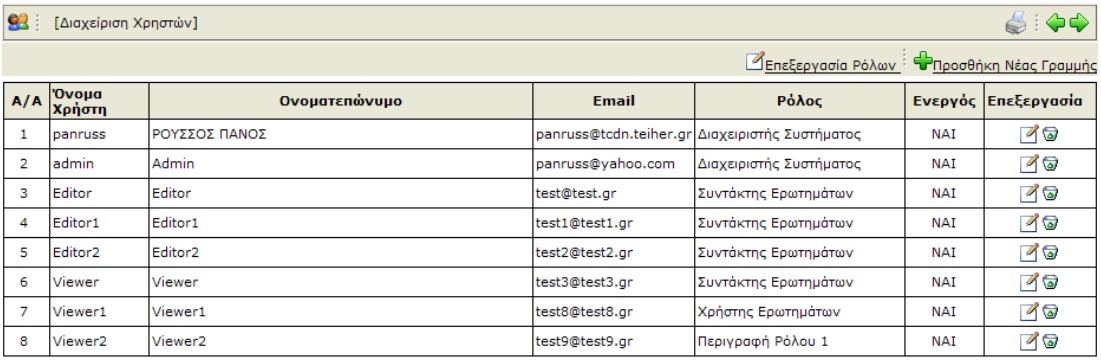
Α/Α	Διάγνωση	Τύπος Επέμβασης	Χειρούργος
1	ΧΟΛΟΛΙΘΙΑΣΗ	ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ	EELCO DE BREE
2		ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ	EELCO DE BREE
3	CA ΤΥΦΛΟΥ	ΗΜΙΚΟΛΕΚΤΟΜΗ ΔΕ - ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ	EELCO DE BREE
4	ΜΕΛΑΝΩΜΑ ΑΡ ΓΑΣΤΡΟΚΝΗΜΙΑΙΑΣ ΧΩΡΑΣ	ΣΥΜΠΛΗΡΩΜΑΤΙΚΗ ΕΚΤΟΜΗ ΚΑΙ ΒΟΥΒΩΝΙΚΟΣ ΛΕΜΦ.ΚΑΘΑΡΙΣΜΟΣ ΑΡ	EELCO DE BREE
5	ΜΕΛΑΓΧΡΩΜΑΤΙΚΗ ΒΛΑΒΗ ΚΝΗΜΗΣ ΑΡ	ΣΥΜΠΛΗΡΩΜΑΤΙΚΗ ΕΚΤΟΜΗ ΚΑΙ ΒΟΥΒΩΝΙΚΟΣ ΛΕΜΦ.ΚΑΘΑΡΙΣΜΟΣ ΑΡ	EELCO DE BREE
6	ΓΕΝΙΚΕΥΜΕΝΗ ΚΑΡΚΙΝΩΜΑΤΩΣΗ ΚΟΙΛΙΑΣ	ΠΛΑΣΤΙΚΗ ΒΟΥΒΩΝΟΚΗΛΗΣ	EELCO DE BREE
7	ΒΟΥΒΩΝΟΚΗΛΗ ΑΡ	ΠΛΑΣΤΙΚΗ ΒΟΥΒΩΝΟΚΗΛΗΣ	EELCO DE BREE
8	CA ΩΟΘΗΚΩΝ ΑΡ	ΟΛΙΚΗ ΥΣΤΕΡΕΚΤΟΜΗ ΜΕΤΑ ΤΩΝ ΕΞΑΡΤΗΜΑΤΩΝ, ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ, ΣΥΜΠΛΗΡΩΜΑΤΙΚΗ ΕΠΙΠΛΟΕΚΤΟΜΗ, ΕΚΤΟΜΗ ΤΜΗΜΑΤΟΣ ΜΕΙΖΟΝΟΣ ΤΟΞΟΥ, ΕΝΔΟΠΕΡΙΤΟΝΑΙΚΗ ΥΠΕΡΘΕΡΜΙΚΗ ΧΜΘ	EELCO DE BREE
9	ΜΕΛΑΝΩΜΑ ΔΕ ΚΡΟΤΑΦΙΚΗΣ ΧΩΡΑΣ. ΒΙΟΨΙΑ ΛΕΜΦΑΔΕΝΑ ΦΡΟΥΡΟΥ.	SLNB (HOT ONLY)	EELCO DE BREE
10	ΙΚΤΕΡΟΣ - ΚΟΙΛΙΑΚΟ ΑΛΓΟΣ	ΠΑΡΟΧΕΤΕΥΣΗ ΑΠΟΣΤΗΜΑΤΩΝ - ΕΚΠΛΥΣΗ ΠΕΡΙΤΟΝΑΙΚΗΣ ΚΟΙΛΟΤΗΤΑΣ	EELCO DE BREE
11	ΧΟΛΟΛΙΘΙΑΣΗ	ΛΑΠΑΡΟΣΚΟΠΙΚΗ ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ	EELCO DE BREE
12	ΙΚΤΕΡΟΣ - ΚΟΙΛΙΑΚΟ ΑΛΓΟΣ	ΠΑΡΟΧΕΤΕΥΣΗ ΑΠΟΣΤΗΜΑΤΟΣ	EELCO DE BREE
13	ΧΟΛΟΛΙΘΙΑΣΗ. ΛΑΠΑΡΟΣΚΟΠΙΚΗ ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ.	ΛΑΠΑΡΟΣΚΟΠΙΚΗ ΧΟΛΟΚΥΣΤΕΚΤΟΜΗ	EELCO DE BREE
14	CA ΟΡΘΟΣΙΓΜΟΕΙΔΟΥΣ. ΧΑΜΗΛΗ ΠΡΟΣΘΙΑ ΕΚΤΟΜΗ.	ΧΑΜΗΛΗ ΠΡΟΣΘΙΑ ΕΚΤΟΜΗ	EELCO DE BREE

Εικόνα 16. Παράδειγμα εκτέλεσης ερωτήματος

3.5.3 Διαχείριση Χρηστών

3.5.3.1 Λίστα Χρηστών

Επιλέγοντας τον σύνδεσμο «Διαχείριση Χρηστών» από το menu «Διαχείριση» εμφανίζεται μία λίστα με τους εγγεγραμένους στην εφαρμογή χρήστες, όπως φαίνεται στην εικόνα 15. Δικαίωμα διαχείρισης των χρηστών έχουν οι ρόλοι στους οποίους έχει οριστεί το δικαίωμα αυτό.


[Διαχείριση Χρηστών]

Επεξεργασία Ρόλων | Προσθήκη Νέας Γραμμής

Α/Α	Όνομα Χρήστη	Ονοματεπώνυμο	Email	Ρόλος	Ενεργός	Επεξεργασία
1	panruss	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	panruss@tcdn.teiher.gr	Διαχειριστής Συστήματος	ΝΑΙ	
2	admin	Admin	panruss@yahoo.com	Διαχειριστής Συστήματος	ΝΑΙ	
3	Editor	Editor	test@test.gr	Συντάκτης Ερωτημάτων	ΝΑΙ	
4	Editor1	Editor1	test1@test1.gr	Συντάκτης Ερωτημάτων	ΝΑΙ	
5	Editor2	Editor2	test2@test2.gr	Συντάκτης Ερωτημάτων	ΝΑΙ	
6	Viewer	Viewer	test3@test3.gr	Συντάκτης Ερωτημάτων	ΝΑΙ	
7	Viewer1	Viewer1	test8@test8.gr	Χρήστης Ερωτημάτων	ΝΑΙ	
8	Viewer2	Viewer2	test9@test9.gr	Περιγραφή Ρόλου 1	ΝΑΙ	

Εικόνα 17. Λίστα χρηστών

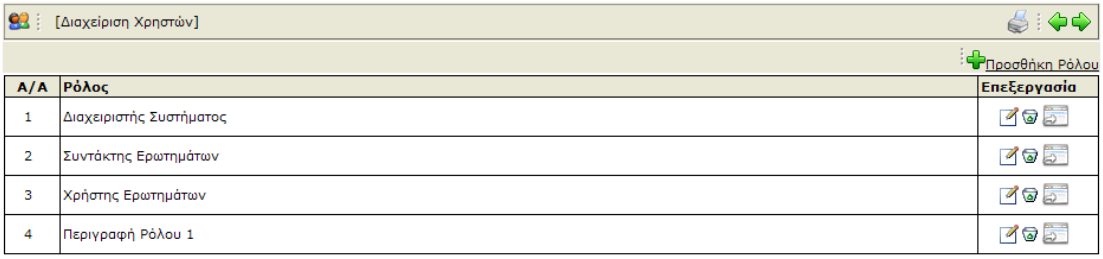
Η λίστα των χρηστών περιέχει πληροφορίες όπως το όνομα χρήστη(username), τον κωδικό πρόσβασης(password – μόνο σε κατάσταση επεξεργασίας), το ονοματεπώνυμο του, το email, τον ρόλο του στην εφαρμογή και αν είναι ενεργοποιημένος. Οι ενέργειες που μπορούν να γίνουν από την φόρμα αυτή είναι:

- Προσθήκη νέου χρήστη
- Επεξεργασία Χρήστη
- Διαγραφή Χρήστη
- Επεξεργασία Ρόλων Χρηστών

Η εφαρμογή μας παραπέμπει σε μία φόρμα διαχείρισης των ρόλων χρηστών

3.5.3.2 Ρόλοι Χρηστών και Δικαιώματα

Όπως αναφέραμε στην προηγούμενη ενότητα επιλέγοντας τον σύνδεσμο «Επεξεργασία Ρόλων Χρηστών» η εφαρμογή μας παραπέμπει στην φόρμα διαχείρισης των ρόλων, η οποία ουσιαστικά είναι μια λίστα με τους υπάρχοντες ρόλους όπως φαινεται στην εικόνα 16.


[Διαχείριση Χρηστών]

Προσθήκη Ρόλου

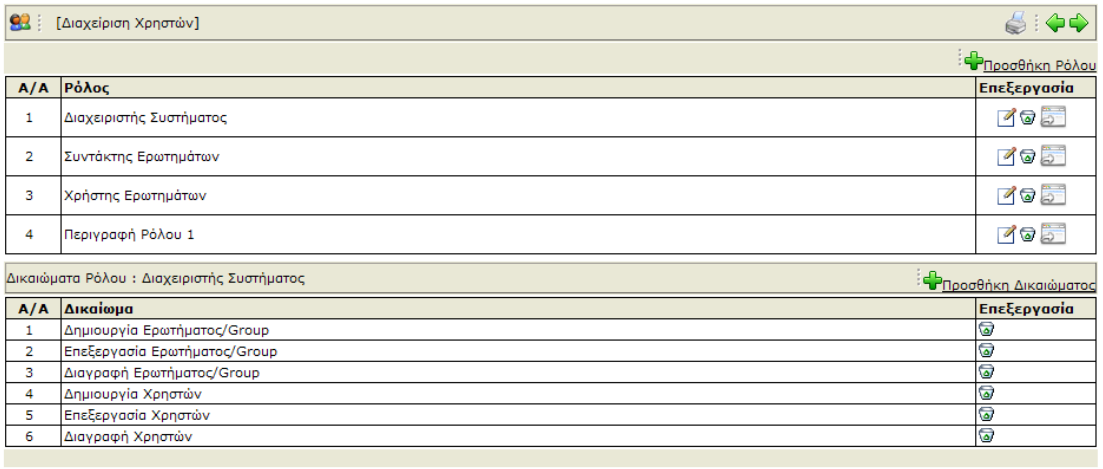
Α/Α	Ρόλος	Επεξεργασία
1	Διαχειριστής Συστήματος	
2	Συντάκτης Ερωτημάτων	
3	Χρήστης Ερωτημάτων	
4	Περιγραφή Ρόλου 1	

Εικόνα 18. Λίστα Ρόλων Χρηστών

Η λίστα αυτή περιέχει την περιγραφή του ρολου και μια εργαλειοθήκη. Οι ενέργειες που μπουρούν να γίνουν από την φόρμα αυτή είναι:

- Προσθήκη ρόλου( )
- Επεξεργασία ρόλου( )
- Διαγραφή ρόλου( )
- Ορισμός δικαιωμάτων ρόλου( )

Επιλέγοντας την ενέργεια «Ορισμός δικαιωμάτων ρόλου» ( ) εμφανίζεται μια λίστα με τα δικαιώματα που έχουν οριστεί στον ρόλο αυτό, όπως φαίνεται στην εικόνα 17.


[Διαχείριση Χρηστών]

Προσθήκη Ρόλου

Α/Α	Ρόλος	Επεξεργασία
1	Διαχειριστής Συστήματος	
2	Συντάκτης Ερωτημάτων	
3	Χρήστης Ερωτημάτων	
4	Περιγραφή Ρόλου 1	

Δικαιώματα Ρόλου : Διαχειριστής Συστήματος — Προσθήκη Δικαιώματος

Α/Α	Δικαίωμα	Επεξεργασία
1	Δημιουργία Ερωτήματος/Group	
2	Επεξεργασία Ερωτήματος/Group	
3	Διαγραφή Ερωτήματος/Group	
4	Δημιουργία Χρηστών	
5	Επεξεργασία Χρηστών	
6	Διαγραφή Χρηστών	

Εικόνα 19. Ρόλος και Δικαιώματα

Οι ενέργειες που μπορούν να γίνουν απο την λίστα αυτή είναι:

- Προσθήκη Δικαιώματος( )
- Διαγραφή Δικαιώματος( )

4. Τεχνική Περιγραφή της Εφαρμογής

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούμε στον τρόπο υλοποίησης διάφορων διαδικασιών.

Αρχικά να αναφέρουμε οτι κάθε εφαρμογή η οποία έχει υλοποιηθεί με την χρήση της τεχνολογίας ASP.NET περιέχει ένα παραμετροποιήσιμο αρχείο, το «*web.config*», το οποίο περιέχει πληροφορίες για την λειτουργία της εφαρμογής, όπως την διεύθυνση της Βάσης Δεδομένων(«ConnectionString»), κατασκευασμένα μηνύματα λάθους(«Custom Error Messages») και τοπικές ρυθμίσεις γλώσσας. Το αρχείο αυτό είναι ένα XML αρχείο και περιέχει σωστά δομημένα XML τμήματα, με μοναδικό περιορισμό οτι το αρχικό τμήμα πρέπει να είναι το «*<configuration>*». Μέσα σε αυτό το τμήμα μπορούν να προστεθούν πολλά άλλα υποτμήματα.

Το τμήμα «*<configuration>*» περιέχει ένα μόνο παιδί το οποίο είναι το «*<system.web>*», μέσα στο οποίο περιέχονται πληροφορίες που αφορούν βασικές λειτουργίες της εφαρμογής, όπως τα τμήματα `compilation`, `customErrors`, `authentication`, `authorization`, κτλ. Πέρα από το τμήμα «<system.web>» μπορούμε να προσθέσουμε δικά μας τμήματα που αφορούν την λειτουργία της εφαρμογής, τα οποία λέγονται «*<appSettings>*».

Υπάρχουν πολλές ρυθμίσεις που μπορούμε να συμπεριλάβουμε στο web.config, αλλά ας αναλύσουμε τα τμήματα που έχουμε χρησιμοποιήσει και επεξεργαστεί:

- system.web

Το τμήμα αυτό, όπως έχουμε αναφέρει, περιέχει ρυθμίσεις που αφορούν την λειτουργία της εφαρμογής. Στην εποχή πριν του .NET, οι ρυθμίσεις αυτές θα περιέχονταν στον IIS. Οι ρυθμίσεις αυτές είναι:

- <authentication>

Το τμήμα αυτό ελέγχει τον τροπό αυθεντικοποίησης των χρηστών της εφαρμογής. Αν δεν θέλουμε να υπάρχει έλεγχος για την πρόσβαση στην εφαρμογή μας, επιλέγουμε την τιμή «None». Αν απαιτείται αυθεντικοποίηση μπορούμε να επιλέξουμε ανάμεσα στις τιμές «Windows», «Forms» ή «Passport». Στην εφαρμογή μας χρησιμοποιούμε το εξής:

```
<authentication mode="Forms">
  <forms name=".ASPXCOOKIEDEMO" protection="All" timeout="60"
loginUrl="System/Login.aspx" />
</authentication>
```

- <authorization>

  Το τμήμα αυτό χρησιμοποιείται για να επιτρέψουμε ή αποτρέψουμε την είσοδο συγκεκριμένων ρόλων. Αυτό γίνεται χρησιμοποιώντας τα υποτμήματα «Deny» και «Allow». Στην εφαρμογή μας ουσιαστικά δεν χρησιμοποιούμε το τμήμα αυτό και επιτρέπουμε σε όλους τους χρήστες την πρόσβαση, όπως φαίνεται στο παράδειγμα.

```
<authorization>
    <deny users="?"/> <!-- Allow all users -->
</authorization>
```

- Custom Errors

Στο τμήμα αυτό μπορούμε, αν θέλουμε δικές μας σελίδες, στις οποίες θα προβάλουμε τα τυχόν λάθη που θα εμφανιστούν κατά την λειτουργία της εφαρμογής με έναν πιο φιλικό και καταννοητό προς τον χρήστη τρόπο. Οι τιμές που παίρνει αυτή η ρύθμιση

είναι «On» και «RemotelyOnly». Η διαφορά στις δύο τιμές είναι, οτι στη μεν πρώτη, όλοι οι χρήστες βλέπουν τις σελίδες που έχουμε ορίσει, σε αντίθεση με την άλλη τιμή με την οποία οι τοπικοί χρήστες μπορούν να δουν τα πραγματικά μηνύματα της εφαρμογής που παράγει το ASP.NET. Εμείς το χρησιμοποιούμε με τον δεύτερο τρόπο ως εξής:

```
<customErrors mode="RemoteOnly"/>
```

- Globalization

Το τμήμα αυτό περιέχει πληροφορίες σχετικά με την κωδικοποίηση που χρησιμοποιούμε στην εφαρμογή. Από μόνο του το θέμα αυτό μπορεί να μας απασχολήσει πάρα πολύ, αλλά εμείς θα αναφερθούμε περιληπτικά. Εν ολίγοις το τμήμα αυτό μας επιτρέπει να ρυθμίσουμε την κωδικοποίηση με την οποία ο εξυπηρετητής(server) θα αποστείλει τα δεδόμενα στον πελάτη(client), καθώς επίσης και τις τοπικές ρυθμίσεις όπως την προβολή ημερομηνιών και νομίσματος. Στην εφαρμογή μας το χρησιμοποιούμε ως εξής:

```
<globalization fileEncoding="utf-8" requestEncoding="utf-8" responseEncoding="utf-8"/>
```

- appSettings

Πέρα από της ρυθμίσεις της εφαρμογής υπάρχει το τμήμα αυτό στο οποίο ο μηχανικός ανάπτυξης λογισμικού μπορεί να χρησιμοποίησει για να αποθηκεύση κάποιες σταθερές μεταβλητές, όπως για παράδειγμα η διεύθυνση της Βάσης Δεδομένων(ConnectionString). Το τμήμα αυτό το χρησιμοποιούμε και εμείς για να αποθηκεύσουμε την τοποθεσία της Β.Δ. ως εξής:

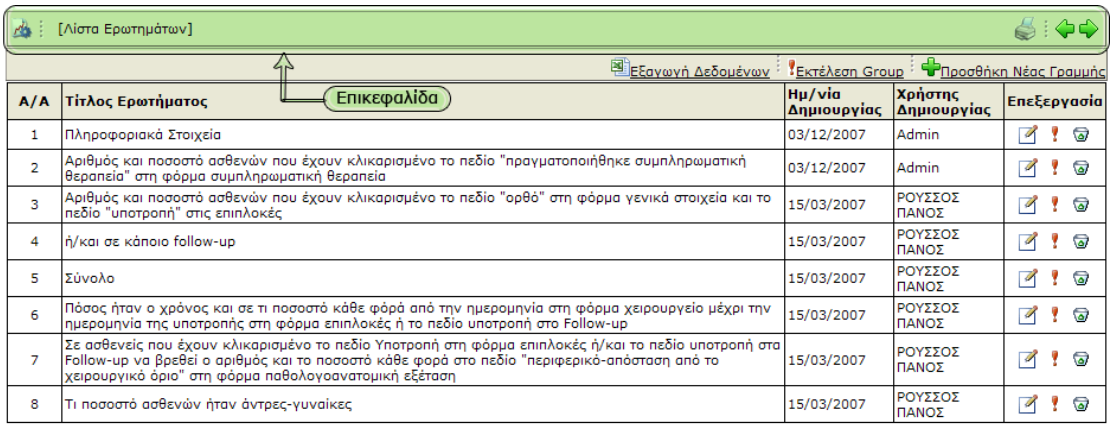
```
<appSettings>
 <add key="SqlStatCon" value="Persist Security Info=False;User ID=SqlStat;Initial Catalog=SqlStats;Data Source=." />
</appSettings>
```

Στο σημείο αυτό να αναφέρουμε οτι μπορούμε να έχουμε πολλά τέτοια παραμετροποιήσιμα αρχεία μέσα στην εφαρμογή μας σε διαφορετικά επίπεδα(φακέλους). Οι ρυθμίσεις του κάθε αρχείου εφαρμόζονται στον φάκελο στον οποίο βρίσκεται το αρχείο αυτό. Το αρχείο web.config προστατεύεται απο τον εξυπηρετητή (Web Server - IIS) και δεν μπορεί να το δει κάποιος πελάτης. Αν προσπαθήσει κάποιος να δει το αρχείο αυτό του εμφανίζεται ένα μήνυμα οτι δεν του επιτρέπεται η πρόσβαση.Ο εξυπηρετητής (Web Server - IIS) παρακολουθεί το αρχείο αυτό γιατί το έχει αποθηκευμένο στην μνήμη του και όταν τροποποιηθεί το αντιλαμβάνεται και το αναννεώνει, χωρίς να απαιτείται επανεκκίνηση.

4.2 Εισαγωγή στα User Controls

Υπάρχουν αρκετοί ορισμοί οι οποίοι προσπαθούν να εξηγήσουν τι είναι User Control, αλλά εμείς θα προσπαθήσουμε να το εξηγήσουμε όσο πιο απλά γίνεται. User Control είναι ουσιαστικά σελίδες που περιέχουν Server Controls ή HTML Controls και μπορούν να χρησιμοποιηθούν από άλλες σελίδες. Με τον τρόπο αυτόν αυξάνεται η χρηστικότητα του κώδικα που έχουμε δημιουργήσει και μειώνεται ο χρόνος υλοποίησης της εφαρμογής. Τα Server Controls παρέχουν μεγάλη χρηστικότητα, αλλά δεν μπορούν καλύψουν κάθε περίπτωση. Για τον λόγο αυτό δημιουργήθηκαν τα User Controls. Τα User Controls μοιάζουν με μια απλή σελίδα, αλλά μπορούν να χρησιμοποιηθούν σαν Server Controls.

Ο λόγος στον οποίο αναφερόμαστε σε αυτός το είδος των εργαλείων είναι επειδή τα χρησιμοποιούμε αρκετά στην εφαρμογή μας. Ένα απλό παράδειγμα ενός User Control είναι η επικεφαλίδα που υπάρχει σε κάθε σελίδα, όπως φαίνεται στην εικόνα.



Α/Α	Τίτλος Ερωτήματος	Ημ/νία Δημιουργίας	Χρήστης Δημιουργίας	Επεξεργασία
1	Πληροφοριακά Στοιχεία	03/12/2007	Admin	
2	Αριθμός και ποσοστό ασθενών που έχουν κλικαρισμένο το πεδίο "πραγματοποιήθηκε συμπληρωματική θεραπεία" στη φόρμα συμπληρωματική θεραπεία	03/12/2007	Admin	
3	Αριθμός και ποσοστό ασθενών που έχουν κλικαρισμένο το πεδίο "ορθό" στη φόρμα γενικά στοιχεία και το πεδίο "υποτροπή" στις επιπλοκές	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
4	ή/και σε κάποιο follow-up	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
5	Σύνολο	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
6	Πόσος ήταν ο χρόνος και σε τι ποσοστό κάθε φόρά από την ημερομηνία στη φόρμα χειρουργείο μέχρι την ημερομηνία της υποτροπής στη φόρμα επιπλοκές ή το πεδίο υποτροπή στο Follow-up	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
7	Σε ασθενείς που έχουν κλικαρισμένο το πεδίο Υποτροπή στη φόρμα επιπλοκές ή/και το πεδίο υποτροπή στα Follow-up να βρεθεί ο αριθμός και το ποσοστό κάθε φορά στο πεδίο "περιφερικό-απόσταση από το χειρουργικό όριο" στη φόρμα παθολογοανατομική εξέταση	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	
8	Τι ποσοστό ασθενών ήταν άντρες-γυναίκες	15/03/2007	ΡΟΥΣΣΟΣ ΠΑΝΟΣ	

Εικόνα 20. Επικεφαλίδα

Η επικεφαλίδα είναι ένα user control και αποτελείται από εναν πίνακα που περιέχει 3 εικόνες και ένα κείμενο. Η πρώτη εικόνα απο τα αριστερά και το κείμενο έχουν οριστεί σαν ιδιότητες στο User Control, καθώς σε κάθε σελίδα τροποποιούνται, οι οποίες είναι:

- Η Εικόνα

Όπως παρατηρούμε από τα αριστερά την εικόνα, βλέπουμε ένα εικονίδιο. Αυτό σε κάθε σελίδα μπορεί να είναι διαφορετικό και έχει οριστεί από εμας. Η ιδιότητα αυτή περιέχει την πληροφορία της διεύθυνσης της εικόνας για να δείξει.

- Ο Τίτλος

Τον τίτλο της σελίδας που θα εμφανιστεί.

Τον εικονίδιο της εκτύπωσης() εμφανίζεται πάντα και η λειτουργία του είναι να εκτυπώνει τα περιεχόμενα της σελίδας. Ομοίως τα εικονίδια της προηγούμενης () και επόμενης() σελίδας εμφανίζονται πάντα και έχουν την ίδια λειτουργία, την μετάβαση, βάσει ιστορικού, στην επόμενη και προηγούμενη σελίδα.

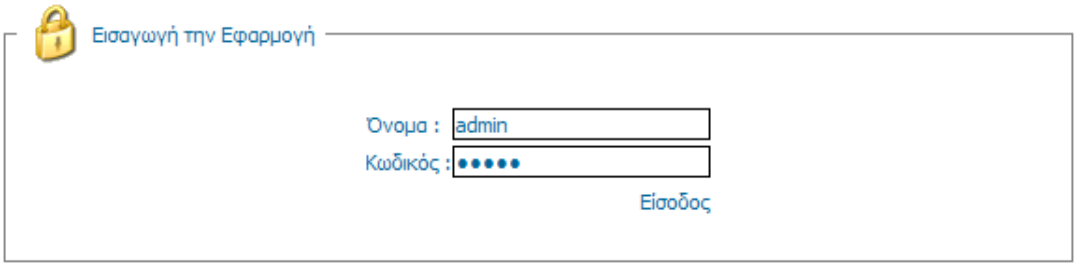
Αυτό ήταν ένα μικρό παράδειγμα για το είναι User Control και πως μπορεί να χρησιμοποιηθεί. Κάθε σελίδα έχει αυτο το User Control και το μόνο που κάνουμε κάθε φορά είναι να γράφουμε την εικόνα και τον τίτλο της σελίδας. Οι υπόλοιπες λειτουργίες κληρονομούνται αυτομάτως.

Άλλα User Controls που χρησιμοποιούμε είναι για την προβολή των αποτελεσμάτων μετά την εκτέλεση ενός ερωτήματος. Είναι τα MultiStat, SimpleStat και pieChartStat User Controls που χρησιμοποιούμε. Το καθένα έχει το δικό του ρόλο, όπως για παράδειγμα το MultiStat είναι υπεύθυνο για την παραγωγή αποτελεσμάτων σε μορφή λίστας, το SimpleStat είναι υπεύθυνο για την παραγωγή του απλού αποτελέσματος και το pieChartStat είναι υπεύθυνο για την παραγωγή των γραφημάτων.

4.2 Εισαγωγική Σελίδα

Στο κεφάλαιο της λειτουργικής περιγραφής της εφαρμογής(κεφάλαιο 2.3) παρουσιάσαμε τον τρόπο με τον οποίο ο χρήστης μπορεί να συνδεθεί στην εφαρμογή και να προχωρήσει στις ενέργειες που θέλει. Στο κεφάλαιο αυτό θα αναλύσουμε σε τεχνικό επίπεδο, τις διαδικασίες τις οποίες αναφέραμε οτι εκτελούνται κατά την ενέργεια της αυθεντικοποίησης του χρήστη.

Αφού ο χρήστης συμπληρώσει τα στοιχεία εισόδου του στην εφαρμογή επιλέγει το πλήκτρο «Είσοδος» για να αυθεντικοποιηθεί ο χρήστης στην εφαρμογή, όπως φαίνεται στην εικόνα 18.



Εικόνα 21. Είσοδος

Ο κώδικας που εκτελείται στην εφαρμογή είναι ο εξής:

```
private void lbLogIn Click(object sender, System.EventArgs e)
{
        // Attempt to Validate User Credentials using UsersDB
        UsersDB accountSystem = new UsersDB();
        int userId = accountSystem.Login(txtUserName.Text, txtPassword.Text);
        if (userId > 0 ){
        // Use security system to set the UserID within a client-side Cookie
        FormsAuthentication.RedirectFromLoginPage(Convert.ToString(userId),
RememberCheckbox.Checked);
        }
        else {
                this.Label3.Text = "Η είσοδος στο σύστημα για τον χρήστη " +
txtUserName.Text + " απέτυχε. Προσπαθήστε ξανά!";
        }
}
```

Η μέθοδος *lbLogIn_Click* είναι μια μέθοδος της εφαρμογής που έχει ισχεί μόνο για την εισαγωγική σελίδα και δεν επιστρέφει κάποια τιμή ύστερα από την εκτέλεση της. Η μέθοδος αυτή εκτελείται όταν ενεργοποιηθεί το κουμπί «Είσοδος», όταν δηλαδή το επιλέξει ο χρήστης. Όπως φαίνεται από τον κώδικα που έχουμε παραθέσει για την αυθεντικοποίηση του χρήστη δημιουργείται ένα αντικείμενο τύπου *UsersDB* και στην συνέχεια από το αντικείμενο αυτό, καλείται μια μέθοδος που ελέγχει αν χρήστης ανήκει στην εφαρμογή. Η μέθοδος αυτή δέχεται σαν όρισμα το όνομα χρήστη(username) και το κωδικό χρήστη(password), εκτελεί ένα ερώτημα στην Β.Δ. για να διαπιστώσει την ύπαρξη του χρήστη και επιστρέφει ενα νούμερο που αντιστοιχεί στο νούμερο με το οποίο έχει καταχωρηθεί ο χρήστης αυτός στην Β.Δ. Αν το νούμερο αυτό είναι μεγαλύτερο του μηδενός, σημαίνει οτι ο χρήστης υπάρχει καταχωρημένος στην εφαρμογή και η εφαρμογή τον προωθεί στις επόμενες σελίδες της εφαρμογής. Σε διαφορετική περίπτωση εμφανίζεται μήνυμα αποτυχίας εισόδου. Η μέθοδος που ελέγχει τα στοιχεία του χρήστη είναι αυτή που παραθέτουμε στην συνέχεια.

```
public int Login(string username, string EncUserPswd)
{
        SqlConnection con = null ;
        int userId = 0 ;
        try{
         con = new SqlConnection( ConfigurationManager.DatabaseConnectionString ) ;
         con.Open() ;
         SqlCommand cmd = new SqlCommand();
         // -- add parameters
         cmd.CommandText = "QQ_USERS_SELECTBYKEY_SP";
         cmd.CommandType = CommandType.StoredProcedure ;
         cmd.Connection = con ;
         cmd.Parameters.Add("@U LoginName", username ) ;
         cmd.Parameters.Add("@EncPswd", EncUserPswd ) ;
         cmd.Parameters.Add("@U_Pswd", EncUserPswd ) ;

          userId = Convert.ToInt32(cmd.ExecuteScalar() ) ;
        }
        catch(Exception ex){
             con.Close() ;
             throw new Exception("Method Login failed. Reason : " + ex.Message , ex ) ;
        }
        finally{
```

```
                con.Close() ;
        }
        return userId ;
}
```

4.3 Κύρια Σελίδα

Τώρα πλέον που ο χρήστης έχει αυθεντικοποιηθεί, η εφαρμογή τον έχει προωθήσει στην αρχική σελίδα. Όπως έχουμε αναφέρει η αρχική σελίδα περιέχει στατιστικές πληροφορίες(δείκτες) σχετικά με την χρήση της εφαρμογής. Ουσιαστικά αυτό που γίνεται είναι κατά την εμφάνιση της σελίδας βάσει του ρόλου του χρήστη εκτελείται ένα ερώτημα στην Β.Δ. και επιστρέφει μια λίστα με τα αποτελέσματα, όπως φαίνεται στο κώδικα που παραθέτουμε.

```
private void Page_Load(object sender, System.EventArgs e)
{
        // Put user code to initialize the page here
        int userId = int.Parse(this.User.Identity.Name);
        string command = string.Empty;
        DataTable StatDataTable;
        SqlParameter[] sqlParams=null;
        if(!IsPostBack){
                //set select command based on role
                if( StatSecurity.IsAdministrator(userId) ){
                command = string.Format(@"
                SELECT 'Σύνολο Ερωτημάτων' AS Title, COUNT(*) AS 'Counts' FROM QQ_Query
                UNION
                SELECT 'Σύνολο Group Ερωτημάτων' AS Title, COUNT(*) AS 'Counts' FROM
QQ QueryGroup
                UNION
                SELECT 'Σύνολο Χρηστών' AS Title, COUNT(*) AS 'Counts' FROM QQ_Users");
                }
               else {
                command = string.Format(@"
                SELECT 'Σύνολο Ερωτημάτων' AS Title, COUNT(*) AS 'Counts' FROM QQ Query
                WHERE Q CreatedByUserID = @UserID OR Q CreatedByUserID IN ( SELECT
UserID FROM QQ_Users WHERE U_ParentUserID = @UserID )
                UNION
                SELECT 'Σύνολο Group Ερωτημάτων' AS Title, COUNT(*) AS 'Counts' FROM
QQ QueryGroup
                WHERE QG CreatedByUserID = @UserID OR QG CreatedByUserID IN ( SELECT
UserID FROM QQ Users WHERE U ParentUserID = @UserID )
                UNION
                SELECT 'Σύνολο Χρηστών' AS Title, COUNT(*) AS 'Counts' FROM QQ_Users
WHERE UserID = @UserID OR UserID IN ( SELECT UserID FROM QQ Users WHERE U ParentUserID
= @UserID )");
                sqlParams = new SqlParameter[1];
                sqlParams[0] = new SqlParameter("@UserID", userId);
                }

                StatDataTable =
DataManager.FillDataGrid(ConfigurationManager.DatabaseConnectionString, command,
CommandType.Text, sqlParams);
                dgStats.DataSource = StatDataTable;
                dgStats.DataBind();
        }
}
```

4.4 Κατηγορίες Σελίδων

Η εφαρμογή αποτελείται από έναν αριθμό σελίδων, η λογική των οποίων επαναλαμβάνεται τακτικά. Λόγω της επανάληψης αυτής θα αποφύγουμε να περιγράψουμε τεχνικά την κάθε σελίδα, αλλά θα ομαδοποιήσουμε της σελίδας αυτές και θα περιγράψουμε το κάθε είδος. Από την εφαρμογή διαπιστώνουμε οτι υπάρχουν τα εξής είδη σελίδων, τα οποία και θα περιγράψουμε αναλυτικά στην συνέχεια:

- Σελίδα με Απλή Λίστα

- Σελίδα με Σύνθετη Λίστα
- Σελίδα Επεξεργασίας Ερωτήματος
- Σελίδα Εκτέλεσης Ερωτήματος

4.4.1 Σελίδα με Απλή Λίστα

Σελίδα με απλή λίστα έχουμε χαρακτηρίσει τις σελίδες εκείνες οι οποίες αποτελούνται από μία απλή λίστα δεδομένων και συνήθως μια επικεφαλίδα η οποία περιέχει πληροφορίες για την σελίδα, όπως για παράδειγμα ο τίτλος. Ένα τυπικό παράδειγμα μιας τέτοιας σελίδας είναι η σελίδα των ερωτημάτων - [Λίστα Group Ερωτημάτων], όπως φαίνεται στην Εικόνα 20.

[Λίστα Group Ερωτημάτων]

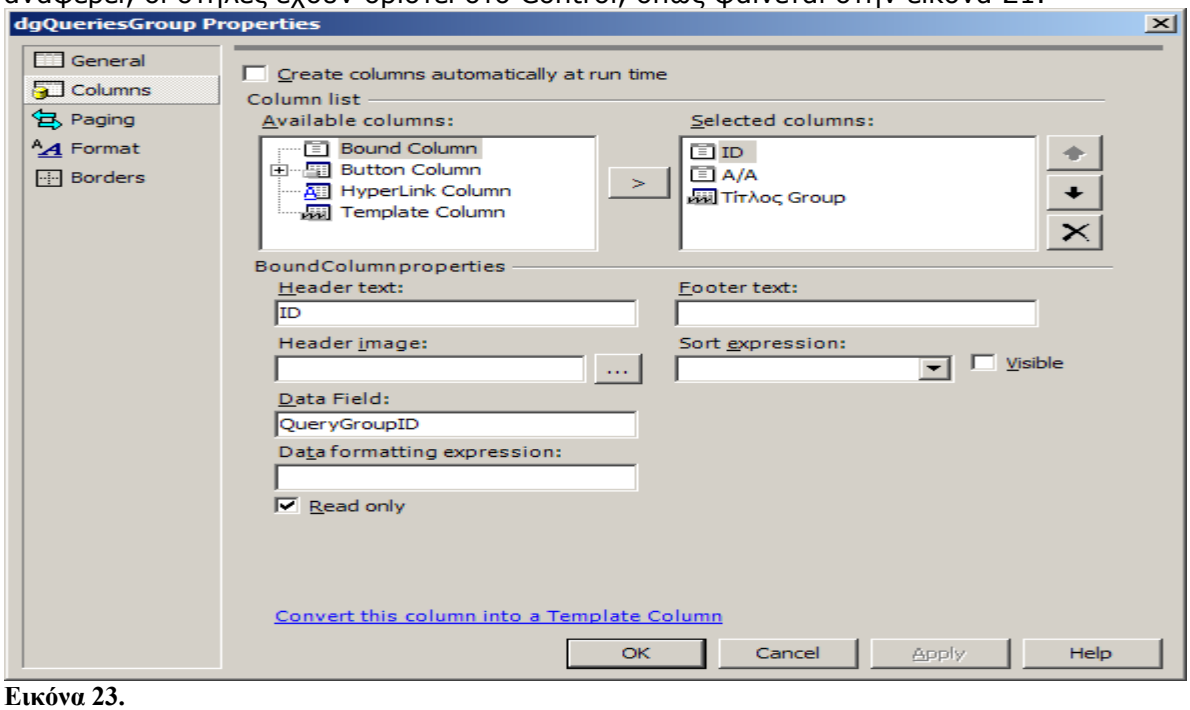
Α/Α	Τίτλος Group
1	StatOnLine
2	ΜΗΤΡΩΟ ΚΛΙΝΙΚΗΣ ΧΟΓΚ
3	test3
4	TCDN e-xogk

Εικόνα 22. Σελίδα με Απλή Λίστα

Όπως φαίνεται και από την Εικόνα 20, η σελίδα αποτελείται από μία επικεφαλίδα και μια λίστα με τα groups των ερωτημάτων. Η επικεφαλίδα περιέχει την πληροφορία του τίτλου της σελίδας, ένα εικονίδιο που την χαρακτηρίζει, την επιλογή «Εκτύπωση» και την δυνατότητα μετακίνησης σε προηγούμενη και επόμενη σελίδα βάσει του ιστορικού του περιηγητή που χρησιμοποιούμε, στην περίπτωση μας Internet Explorer. Η λίστα αυτή είναι ένα Control της Microsoft και η ονομασία του είναι *DataGrid*. Για να δημιουργηθεί η λίστα αυτή πρέπει να γίνουν οι εξής κινήσεις:

1. Να οριστούν οι στήλες της λίστας

Για την δημιουργία των στηλών της λίστας υπάρχουν δύο τρόποι. Ο ένας τρόπος είναι να πούμε στο DataGrid να δημιουργήσει τις στήλες που θα βρεί απο την πήγη που έχουμε ορίσει και ο άλλος τρόπος είναι να ορίσουμε εμείς στο DataGrid ποιες στήλες θα δείξει ανεξαρτήτως των στηλών που περιέχει η πηγή. Εμείς χρησιμοποιούμε και τους δύο τρόπους. Στο συγκεκριμένο παράδειγμα που έχουμε αναφέρει, οι στήλες έχουν οριστεί στο Control, όπως φαίνεται στην εικόνα 21.



Εικόνα 23.

2. Να οριστεί η πηγή της λίστας

Η πηγή ορίζεται στο Control εκτελώντας ένα ερώτημα στην βάση και το αποτέλεσμα του το θέτουμε στην ιδιότητα «DataSource» του Control, όπως φαίνεται παρακάτω.

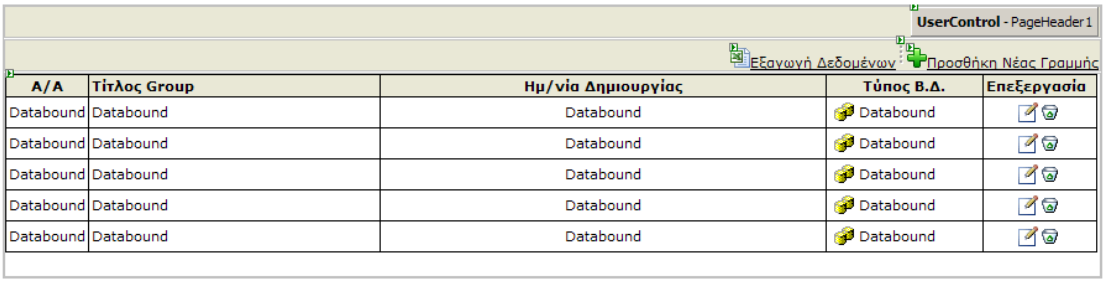
```
private void BindGrid(){
        userId = Convert.ToInt32(this.User.Identity.Name);
        //show user data grid
        DataTable dt = DataManager.showQueryGroupsByUser( userId ) ;
        dgQueriesGroup.DataSource = dt ;
        if( dt.Rows.Count > 10 )
                dgQueriesGroup.AllowPaging = true ;
        dgQueriesGroup.DataBind() ;
}
```

3. Να δωθεί η εντολή για την δημιουργία της λίστας

Η εντολή για την δημιουργία της λίστας φαίνεται στο παράδειγμα που έχουμε παραθέσει παραπάνω και είναι η μέθοδος «DataBind». Η εντολή αυτή ουσιαστικά θέτει τις τιμές από την πηγή στο Control.

4.4.2 Σελίδα με Σύνθετη Λίστα

Η σελίδα με σύνθετη λίστα είναι παρόμοια με την σελίδα με απλή λίστα με την διαφορά ότι η σύνθετη λίστα περιέχει μια στήλη ακόμα η οποία περιέχει διάφορες επιλογές όπως επεξεργασία της γραμμής, διαγραφή κλπ. Ένα παράδειγμα τέτοιας λίστας είναι η σελίδα που φαίνεται στην εικόνα 22.



Εικόνα 24. Σελίδα με Σύνθετη Λίστα

Η σελίδα αυτή αποτελείται απο μία επικεφαλίδα, την επιλογή «Προσθήκη Νέας Γραμμής» και την λίστα. Η λίστα έχει δημιουργηθεί με τον ίδιο τρόπο που περιγράψαμε νωρίτερα με την διαφορά οτι έχει οριστεί μία επιπλέον στήλη με τον τίτλο «Επεξεργασία» και περιέχει τις επιλογές «Επεξεργασία», «Διαγραφή», «Αποθήκευση» και «Αναίρεση». Ας αναλύσουμε τώρα τις επιλογές της σελίδας.

«Προσθήκη Νέας Γραμμής»: Σε όλες τις σελίδες η λογική της επιλογής αυτής είναι ίδια. Προσθέτει μια νέα κενή γραμμή στην λίστα όπως φαίνεται στο ακόλουθο παράδειγμα.

```
private void addRow Click(object source, System.EventArgs e) {
        int userId = Convert.ToInt32(this.User.Identity.Name) ;

        try{
                SqlParameter[] sqlParams = new SqlParameter[5];
                sqlParams[0] = new SqlParameter("@QG Number", string.Empty ) ;
                sqlParams[1] = new SqlParameter("@QG Title" , string.Empty ) ;
                sqlParams[2] = new SqlParameter("@QG_DateCreated" , DateTime.Now ) ;
                sqlParams[3] = new SqlParameter("@QG_CreatedByUserID" , userId ) ;
                sqlParams[4] = new SqlParameter("@QG DbTypeID" , 1 ) ;
                int affectedRows = DataManager.ExecuteNonQuery(
ConfigurationManager.DatabaseConnectionString ,
"QQ QUERYGROUP INSERT SP",CommandType.StoredProcedure, sqlParams ) ;

                //Hide connectionstring field on add
                dgQueriesGroup.Columns[5].Visible = true;
                //bind grid
                BindGrid();
        }
        catch(Exception ex){
```

```
                throw new Exception("Error!! Reason : " + ex.Message ) ;
        }
}
```

Από το παράδειγμα μπορούμε να δούμε οτι εκτελείται μία εντολή στην Β.Δ. η οποία προσθέτει μία γραμμή στον πίνακα που θέλουμε με αρχικοποιημένες τιμές και στην πορεία δημιουργεί την λίστα για φανεί η νέα γραμμή.

«Επεξεργασία Γραμμής» : Η λίστα των Β.Δ. όταν είναι σε κατάσταση επεξεργασίας εμφανίζονται κάποια controls σε κάθε κελί. Για παράδειγμα στο κελί για τον τίτλο του Group εμφανίζεται ένα control για ελεύθερο κείμενο, ενώ στο κελί για τον τύπο της Β.Δ. εμφανίζεται μια λίστα με τους τύπους των Β.Δ. Επίσης εμφανίζεται το κελί με την διεύθυνση(ConnectionString) της Β.Δ.

```
private void dgQueriesGroup_EditCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
        //bind
        tbl = DataManager.FillDataGrid( ConfigurationManager.DatabaseConnectionString ,
        "SELECT * FROM QQ_DatabaseType" , CommandType.Text ) ;

        //Show ConnectionString field on edit
        dgQueriesGroup.Columns[5].Visible = true;
        dgQueriesGroup.EditItemIndex = e.Item.ItemIndex ;
        BindGrid() ;
}
```

«Αποθήκευση»:

```
private void dgQueriesGroup UpdateCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e){
        // get Values
        string rowId = e.Item.Cells[0].Text ;

        string viewOrder = ((TextBox)e.Item.Cells[1].Controls[1]).Text ;
        string title = ((TextBox)e.Item.Cells[2].Controls[1]).Text ;
        DateTime dateCreated=DateTime.Parse(((Label)e.Item.Cells[3].Controls[1]).Text)
;
        int dbTypeID=
nt.Parse(((DropDownList)e.Item.Cells[4].Controls[1]).SelectedValue);
        string clientConnectionString = ( (TextBox)e.Item.Cells[5].Controls[1]).Text ;
        string qg_UserID= this.User.Identity.Name ;

        //add parameters
        SqlParameter[] sqlParams = new SqlParameter[7] ;
        sqlParams[0] = new SqlParameter("@QG Number", viewOrder ) ;
        sqlParams[1] = new SqlParameter("@QG TITLE", title ) ;
        sqlParams[2] = new SqlParameter("@QG_CreatedByUserID", qg_UserID ) ;
        sqlParams[3] = new SqlParameter("@QueryGroupID", rowId ) ;
        sqlParams[4] = new SqlParameter("@QG DbTypeID", dbTypeID ) ;
        sqlParams[5] = new SqlParameter("@QG ConnectionString",clientConnectionString);
        sqlParams[6] = new SqlParameter("@QG DateCreated", dateCreated ) ;

        //execute update
        UpdateRow( sqlParams ) ;
        //Hide connectionstring field on update
        dgQueriesGroup.Columns[5].Visible = false;
        //bind grid
        dgQueriesGroup.EditItemIndex = -1;
        BindGrid();
}
```

«Αναίρεση»:

```
private void dgQueriesGroup CancelCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e){
        //Hide connectionstring field on cancel
        dgQueriesGroup.Columns[5].Visible = false;
        //Bind
        dgQueriesGroup.EditItemIndex = -1 ;
        BindGrid() ;
}
```

«Διαγραφή Γραμμής»:

Με την επιλογή αυτή βάσει του κωδικού που έχει η κάθε εγγραφή διαγράφεται η αντίστοιχη εγγραφή στην βάση δεδομένων.

```
private void dgQueriesGroup_DeleteCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
        DeleteRow( int.Parse( e.Item.Cells[0].Text ) ) ;
        BindGrid() ;
}

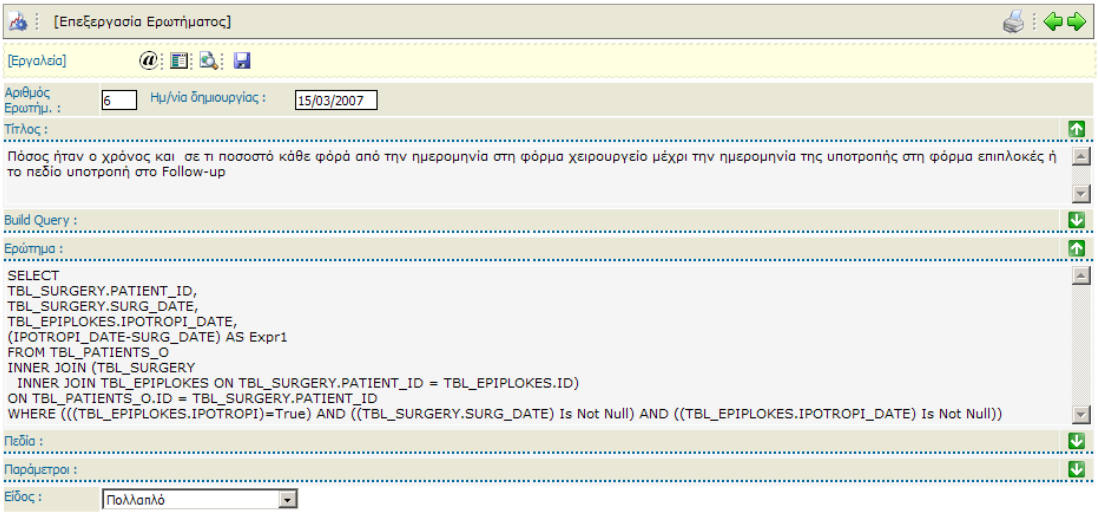
private void DeleteRow( int rowIndex){
        SqlConnection con = null ;
        SqlTransaction trans = null ;

        try{
                con=new SqlConnection(ConfigurationManager.DatabaseConnectionString);
                con.Open() ;
                trans = con.BeginTransaction() ;
                //create command
                SqlCommand cmd = new SqlCommand() ;
                cmd.CommandText = "QQ_QUERYGROUP_DELETE_SP";
                cmd.CommandType = CommandType.StoredProcedure ;
                cmd.Connection = con ;
                cmd.Transaction = trans ;
                //attach parameters
                cmd.Parameters.Add("@QueryGroupID" , rowIndex ) ;
                //execute command
                cmd.ExecuteNonQuery() ;
                //commit it
                trans.Commit();
        }
        catch(Exception ex){
                trans.Rollback();
        }
        finally{
                con.Close() ;
        }
}
```

4.4.3 Σελίδα Επεξεργασίας Ερωτήματος

Η σελίδα στην οποία δημιουργούμε ή επεξεργαζόμαστε τα ερωτήματα μας είναι αρκετά περίπλοκη και αποτελείται απο πολλά τμήματα. Η σελίδα αυτή περιέχει λίστες για τις οποίες έχουμε ήδη αναφερθεί προηγουμένως όπως η Σύνθετη Λίστα και για τον λόγο αυτό δεν θα αναφερθούμε εκτενώς σε τέτοια τμήματα, παρά μόνο αν χρειάζεται. Την λειτουργία της σελίδας την έχουμε περιγράψει σε προηγούμενο κεφάλαιο (

3.4.2.3 Φόρμα Δημιουργίας και Επεξεργασίας Ερωτήματος) όπου μπορείτε επίσης να δείτε την μορφή της στην εικόνα 7. Ας αναλύσουμε τώρα απο τεχνική άποψη τις λειτουργίες της φόρμας ξεκινώντας βάσει της εικόνας από πάνω προς τα κάτω.



Η σελίδα περιέχει αρχικά μια επικεφαλίδα, η οποία όπως έχουμε αναφέρει ήδη, δηλώνει το που βρισκόμαστε μέσω του τίτλου της. Στην συνέχεια μπορούμε να δούμε την εργαλειοθήκη η οποία αποτελείται απο τα εξής τμήματα:

- Αρχικοποιήση Παραμέτρων

Την λειτουργία της επιλογής αυτής την έχουμε περιγράψει. Ας δούμε τώρα πως έχει υλοποιηθεί. Η λογική είναι απλή. Ψάχνουμε για λέξεις που ξεκινάνε με τον χαρακτήρα «@» και τις προσθέτουμε στην κατάλληλη λίστα.

```
private void btnGetParams Click(object sender, System.Web.UI.ImageClickEventArgs e){
        //find parameters and store them in a arrayList
        bool paramChar = false;
        string paramName = "";
        ArrayList paramNames = new ArrayList();
        foreach (char ch in (this.txtSqlEdit.Text+" ").ToCharArray()){
                switch (ch) {
                        case '@' :
                                paramChar = true;
                                paramName = ""+ch;
                                break;
                        case ' ' :
                        case '=' :
                        case '>' :
                        case '<' :
                                paramChar = false;
                                if (paramName.Length>0)
                                        paramNames.Add(paramName);
                                paramName = "";
                                break;
                        default :
                                if (paramChar)
                                        paramName += ch;
                                break;
                }
        }
        //clear query params rows
        queryDataSet.QueryParam.Rows.Clear();
        //now procced. check if already exists in datagrid
        bool exists = false ;
        foreach (string pName in paramNames){
                if( queryDataSet.QueryParam.Rows.Count > 0  ){
                        foreach( DataRow row in queryDataSet.QueryParam.Rows  ){
                                if( row["QP_Name"].ToString() ==
pName.ToUpper().ToString() )
                                        exists = true ;
                                }
                        }
                        if(!exists){
```

```
                                DataRow newRow = queryDataSet.QueryParam.NewRow() ;
                                newRow["QP Name"] = pName.ToUpper();
                                newRow["QP Caption"] = pName.ToUpper();
                                newRow["QP QueryID"] =
queryDataSet.Tables["Query"].Rows[0]["QueryID"] ;
                                newRow["QP_TYPEID"] = 1;
                                newRow["QP_DefaultValue"] = "1";
                                //newRow["QPT Description"] = "Κείμενο";
                                queryDataSet.QueryParam.Rows.Add(newRow) ;
                        }
                        exists = false;
                }
                dgQParams.DataSource = queryDataSet.QueryParam ;
                dgQParams.DataBind();
}
```

- Αρχικοποιήση Πεδίων

Και για την επιλογή αυτή μπορούμε να πούμε οτι στηρίζεται σε απλή λογική. Αυτό που κάνουμε για να αρχικοποιήσουμε τα πεδία είναι να εκτελούμε το ερωτήμα με τέτοιο τρόπο ώστε να μην μας επιστρέφει αποτελέσματα παρά μόνο τις στήλες που αποτελούν το αποτέλεσμα. Αυτό επιτυγχάνετε χρησιμοποιώντας το φίλτρο στο ερώτημα το «WHERE 1 = 0». Είναι μία συνθήκη που δεν ισχύει ποτέ, οπότε και δεν μας επιστρέφει κανένα αποτέλεσμα. Αυτό το κάνουμε για να ελαχιστοποιήσουμε τον χρόνο εκτέλεσης του ερωτήματος. Στην συνεχεια όπως μπορείτε να δείτε και από τον κώδικα προσθέτουμε στον κατάλληλο πίνακα τα πεδία αυτά.

```
private void btnGetQueryFields Click(object sender, System.Web.UI.ImageClickEventArgs
e) {
        try{
        //First check the query contains any suspecious word, such as DELETE or INSERT
                string sqlText = txtSqlEdit.Text;
                if( Utils.ContainsSuspeciousCommands(sqlText) ) {
                        errorReport.Text = "Το ερώτημα που έχετε θέσει περιέχει
επικίνδυνες εντολές, όπως DELETE ή INSERT και δεν μπορεί να εκτελεστεί.";
                        errorReport.Visible = true;
                        return;
                }
                //Now we can continue with out fear...
                //format query like we want e.g. delete "where" clauses
                int whereSeat = sqlText.ToUpper().IndexOf("WHERE");
                if( whereSeat>0){
                        sqlText = sqlText.Substring(0,whereSeat);
                        sqlText += " WHERE 1 = 0 ";
                }
                //clear the query field rows
                queryDataSet.QueryField.Rows.Clear();
                //now proceed. execute sql to get columns
                int qID = QueryID;
                DataTable queryField =
DataManager.ExecuteMultiStatQuery(qID,queryDataSet);
                //now that you have the columns turn them into rows
                foreach(DataColumn col in queryField.Columns ){
                        QueryDataSet.QueryFieldRow qfRow =
(QueryDataSet.QueryFieldRow)queryDataSet.Tables["QueryField"].NewRow();

                        qfRow.QF QueryID = QueryID;
                        qfRow.QF Name = col.ColumnName;
                        qfRow.QF Title = string.Empty;
                        //qfRow.QF LookUpID  = -1;
                        queryDataSet.Tables["QueryField"].Rows.Add(qfRow);
                }

                dgQFields.DataSource = queryDataSet.Tables["QueryField"];
                dgQFields.DataBind();
        }
        catch(Exception ex){
                //clear previous error messages
                errorReport.Text = string.Empty;
                //continue
                errorReport.Text += " Μήνυμα Συστήματος : " + ex.Message ;
                errorReport.Text += "<br><br> Stack Trace : " + ex.StackTrace ;
                errorReport.Visible = true;
        }
```

```
}
```

- Δοκιμαστική Εκτέλεση Ερωτήματος

Ουσιαστικά η επιλογή μας παραπέμπει στην σελίδα Εκτέλεσης Ερωτήματος την οποία θα αναλύσουμε στην συνέχεια. Η παραπομπή αυτή ορίζεται κατά την δημιουργία της σελίδας ως εξής:

```
hpQueryPreview.NavigateUrl = String.Format("showStat.aspx?queryID={0}",QueryID);
```

- Αποθήκευση

Όπως μπορούμε να δούμε και από τον κώδικα που έχουμε παραθέσει η αποθήκευση χωρίζεται σε δύο μέρη. Το ένα είναι η δημιουργία και αποθήκευση ενός νέου ερωτήματος και το δεύτερο μέρος είναι η ανανέωση δεδομένων σε ένα υπάρχων ερώτημα. Η λογική είναι επίσης απλή και εδώ. Παίρνουμε τα δεδομένα από τα Controls που υπάρχουν στην σελίδα και τα αποθηκεύουμε στους αντίστοιχους πίνακες.

```
private void SaveData(){
        //First check the query contains any suspecious word, such as DELETE or INSERT
        if( Utils.ContainsSuspeciousCommands(txtSqlEdit.Text) ) {
                errorReport.Text = "Το ερώτημα που έχετε θέσει περιέχει επικίνδυνες
εντολές, όπως DELETE ή INSERT και δεν μπορεί να εκτελεστεί.";
                errorReport.Visible = true;
                return;
        }
        DataRow curRow ;

        #region "update"
        if( QueryID > 0 ) {
                #region "Query"
                curRow = queryDataSet.Tables["Query"].Rows[0] ;
                //update dataSet for Table Query
                curRow["Q_NUMBER"] = float.Parse(txtQUERY_NUMBER.Text) ;
                curRow["Q TITLE"] = txtTitle.Text ;
                curRow["Q SqlText"] = txtSqlEdit.Text ;
                curRow["Q StatTypeID"] = ddnRESULT CLASS.SelectedValue ;

                SqlParameter[] sqlParams = new SqlParameter[8] ;
                sqlParams[0] = new SqlParameter("@QueryID" , QueryID ) ;
                sqlParams[1] = new SqlParameter("@Q Number" , (float)curRow["Q NUMBER"]
) ;
                sqlParams[2] = new SqlParameter("@Q Title" , curRow["Q Title"] ) ;
                sqlParams[3] = new SqlParameter("@Q SqlText" , curRow["Q SqlText"] ) ;
                sqlParams[4] = new SqlParameter("@Q_StatTypeID" ,
(int)curRow["Q_StatTypeID"] ) ;
                sqlParams[5] = new SqlParameter("@Q QueryGroupID" ,
(int)curRow["Q QueryGroupID"] ) ;
                sqlParams[6] = new SqlParameter("@Q DateCreated" ,
(DateTime)curRow["Q_DateCreated"]  ) ;
                sqlParams[7] = new SqlParameter("@Q_CreatedByUserID" ,
(int)curRow["Q CreatedByUserID"]  ) ;

        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString ,
"QQ QUERY UPDATE SP",CommandType.StoredProcedure, sqlParams );
                #endregion
        #region "QueryParams"
        //now check for params.... if params in dataset are more than zero , check if
exists any params in db..
        //If exists, delete them and re-insert , else insert them...
        SqlParameter[] sqlQParams = new SqlParameter[1] ;
        sqlQParams[0] = new SqlParameter("@QP_QueryID" , QueryID ) ;
        int dbQpRowsCount =
DataManager.ExecuteScalar(ConfigurationManager.DatabaseConnectionString,"SELECT
COUNT(QueryParamID) FROM QQ QueryParam WHERE QP QueryID=@QP QueryID",CommandType.Text,
sqlQParams) ;

        if( dbQpRowsCount > 0 ){
        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"DELE
TE FROM QQ QueryParam WHERE QP QueryID=@QP QueryID",CommandType.Text, sqlQParams) ;
        }
        if( queryDataSet.QueryParam.Rows.Count > 0  ) {
                //now that we have clear params, insert the new..
```

```
                foreach( DataRow r in queryDataSet.QueryParam.Rows  ) {
                        SqlParameter[] sqlQueryParams = new SqlParameter[9] ;
                        sqlQueryParams[0] = new SqlParameter("@QP QueryID" , QueryID ) ;
                        sqlQueryParams[1] = new SqlParameter("@QP Caption" ,
r["QP Caption"] ) ;
                        sqlQueryParams[2] = new SqlParameter("@QP_Name" , r["QP_Name"] )
                        sqlQueryParams[3] = new SqlParameter("@QP_TypeID" ,
r["QP TypeID"] ) ;
                        sqlQueryParams[4] = new SqlParameter("@QP DefaultValue" ,
r["QP DefaultValue"] ) ;
                        sqlQueryParams[5] = new SqlParameter("@QP_LookUpTableName" ,
r["QP_LookUpTableName"] ) ;
                        sqlQueryParams[6] = new SqlParameter("@QP LookUpSql" ,
r["QP LookUpSql"] ) ;
                        sqlQueryParams[7] = new SqlParameter("@QP LookUpField" ,
r["QP LookUpField"] ) ;
                        sqlQueryParams[8] = new SqlParameter("@QP_LookUpConnectionField"
, r["QP_LookUpConnectionField"] ) ;

        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"QQ Q
UERYPARAM INSERT SP",CommandType.StoredProcedure, sqlQueryParams) ;
                }
        }
        #endregion
        #region "QueryFields"
        //now check for fields.... if fields in dataset are more than zero , check if
exists any fields in db..
        //If exists, delete them and re-insert , else insert them...
        if( queryDataSet.QueryField.Rows.Count > 0  ) {
                SqlParameter[] sqlQf = new SqlParameter[1];
                sqlQf[0] = new SqlParameter("@QF QueryID", QueryID );

                int dbQfRowsCount =
DataManager.ExecuteScalar(ConfigurationManager.DatabaseConnectionString,"SELECT
COUNT(QueryFieldID) FROM QQ_QueryField WHERE QF_QueryID=@QF_QueryID",CommandType.Text,
sqlQf) ;

                if( dbQfRowsCount > 0 ) {

        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"DELE
TE FROM QQ_QueryField WHERE QF_QueryID=@QF_QueryID",CommandType.Text, sqlQf) ;
                }
                //now that we have clear fields, insert the new..
                foreach( DataRow r in queryDataSet.QueryField.Rows  ) {
                        SqlParameter[] sqlQueryFields = new SqlParameter[11] ;
                        sqlQueryFields[0] = new SqlParameter("@QF_QueryID" , QueryID ) ;
                        sqlQueryFields[1] = new SqlParameter("@QF_Name" , r["QF_Name"] )
;
                        sqlQueryFields[2] = new SqlParameter("@QF Title" , r["QF Title"]
) ;
                        sqlQueryFields[3] = new SqlParameter("@QF Url" , r["QF Url"] ) ;
                        //sqlQueryFields[4] = new SqlParameter("@QF_UrlFieldName" , null
) ;
                        sqlQueryFields[4] = new SqlParameter("@QF UrlTarget" ,
r["QF UrlTarget"] ) ;
                        sqlQueryFields[5] = new SqlParameter("@QF DetailQueryID" ,
!r["QF DetailQueryID"].Equals(System.DBNull.Value)?((int)r["QF DetailQueryID"]):-1);
                        //sqlQueryFields[7] = new SqlParameter("@QF_DetailParamID" ,
null ) ;
                                                //sqlQueryFields[8] = new
SqlParameter("@QF DetailParamName" , r["QF DetailParamName"] ) ;
                        sqlQueryFields[6] = new SqlParameter("@QF LookUpTableName" ,
r["QF_LookUpTableName"] ) ;
                        sqlQueryFields[7] = new SqlParameter("@QF_LookUpSQL" ,
r["QF LookUpSQL"] ) ;
                        sqlQueryFields[8] = new SqlParameter("@QF LookUpResultField" ,
r["QF LookUpResultField"] ) ;
                        sqlQueryFields[9] = new SqlParameter("@QF LookUpConnectionField"
, r["QF_LookUpConnectionField"] ) ;
                        sqlQueryFields[10] = new SqlParameter("@QF_PossibleCombinations"
, r["QF PossibleCombinations"] ) ;


        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"QQ Q
UERYFIELD_INSERT_SP",CommandType.StoredProcedure, sqlQueryFields) ;
                                        }
        #endregion
```

```
        #region "QueryTable"
        //now check for fields.... if fields in dataset are more than zero , check if
exists any fields in db..
        //If exists, delete them and re-insert , else insert them...
        if( queryDataSet.QueryTable.Rows.Count > 0  ) {

                SqlParameter[] sqlQt = new SqlParameter[1];
                sqlQt[0] = new SqlParameter("@QT QueryID", QueryID );

                int dbQtRowsCount =
DataManager.ExecuteScalar(ConfigurationManager.DatabaseConnectionString,"SELECT
COUNT(QueryTableID) FROM QQ_QueryTable WHERE QT_QueryID=@QT_QueryID",CommandType.Text,
sqlQt) ;

                if( dbQtRowsCount > 0 ) {

        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"DELE
TE FROM QQ_QueryTable WHERE QT_QueryID=@QT_QueryID",CommandType.Text, sqlQt) ;
                }
                //now that we have clear fields, insert the new..
                foreach( DataRow r in queryDataSet.QueryTable.Rows  ) {
                        SqlParameter[] sqlQueryTables = new SqlParameter[3] ;
                        sqlQueryTables[0] = new SqlParameter("@QT_QueryID" , QueryID ) ;
                        sqlQueryTables[1] = new SqlParameter("@QT_TableName" ,
r["QT TableName"] ) ;
                        sqlQueryTables[2] = new SqlParameter("@QT Relations" ,
r["QT Relations"] ) ;


        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"QQ_Q
UERYTABLE INSERT SP",CommandType.StoredProcedure, sqlQueryTables) ;
                }
        }
        #endregion
        }
}
        #endregion
        #region "insert"
        else {
                #region "Query"
                curRow = queryDataSet.Tables["Query"].NewRow() ;
                //update dataSet for Table Query
                curRow["Q NUMBER"] = float.Parse(txtQUERY NUMBER.Text) ;
                curRow["Q DateCreated"] = DateTime.Parse( txtDATE.Text ) ;
                curRow["Q TITLE"] = txtTitle.Text ;
                curRow["Q_SqlText"] = txtSqlEdit.Text ;
                curRow["Q_StatTypeID"] = ddnRESULT_CLASS.SelectedValue ;
                curRow["Q QueryGroupID"] = QueryGroupID ;
                curRow["Q CreatedByUserID"] = this.Page.User.Identity.Name ;

                queryDataSet.Tables["Query"].Rows.Add(curRow) ;
                SqlParameter[] sqlParams = new SqlParameter[7] ;
                sqlParams[0] = new SqlParameter("@Q_DateCreated" , DateTime.Parse(
txtDATE.Text ) ) ;
                sqlParams[1] = new SqlParameter("@Q Number" ,
float.Parse(txtQUERY NUMBER.Text) ) ;
                sqlParams[2] = new SqlParameter("@Q Title" , txtTitle.Text ) ;
                sqlParams[3] = new SqlParameter("@Q_SqlText" , txtSqlEdit.Text ) ;
                sqlParams[4] = new SqlParameter("@Q_StatTypeID" ,
ddnRESULT CLASS.SelectedValue ) ;
                sqlParams[5] = new SqlParameter("@Q QueryGroupID" , QueryGroupID ) ;
                sqlParams[6] = new SqlParameter("@Q CreatedByUserID" ,
int.Parse(this.User.Identity.Name) ) ;

                int insertedID =
DataManager.ExecuteScalar(ConfigurationManager.DatabaseConnectionString ,
"QQ QUERY INSERT SP",CommandType.StoredProcedure, sqlParams );
                //update this public variable
                QueryID = insertedID;

                //now that we have the inserted queryID, insert params...if exists
                if( queryDataSet.QueryParam.Rows.Count > 0  ) {
                        foreach( DataRow r in queryDataSet.QueryParam.Rows  ) {
                                SqlParameter[] sqlQueryParams = new SqlParameter[9] ;
                                sqlQueryParams[0] = new SqlParameter("@QP_QueryID" ,
insertedID ) ;
                                sqlQueryParams[1] = new SqlParameter("@QP_Caption" ,
```

```
r["QP_Caption"] ) ;
                                sqlQueryParams[2] = new SqlParameter("@QP Name" ,
r["QP Name"] ) ;
                                sqlQueryParams[3] = new SqlParameter("@QP TypeID" ,
r["QP TypeID"] ) ;
                                sqlQueryParams[4] = new SqlParameter("@QP_DefaultValue"
, r["QP_Value"] ) ;
                                sqlQueryParams[5] = new
SqlParameter("@QP LookUpTableName" , r["QP LookUpTableName"] ) ;
                                sqlQueryParams[6] = new SqlParameter("@QP LookUpSql" ,
r["QP_LookUpSql"] ) ;
                                sqlQueryParams[7] = new SqlParameter("@QP_LookUpField" ,
r["QP LookUpField"] ) ;
                                sqlQueryParams[8] = new
SqlParameter("@QP LookUpConnectionField" , r["QP LookUpConnectionField"] ) ;

        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"QQ_Q
UERYPARAM_INSERT_SP",CommandType.StoredProcedure, sqlQueryParams) ;
                        }
                }
                #endregion
                #region "QueryTable"
                if( queryDataSet.QueryTable.Rows.Count > 0  ) {
                        foreach( DataRow r in queryDataSet.QueryTable.Rows  ) {
                                SqlParameter[] sqlQueryTables = new SqlParameter[3] ;
                                sqlQueryTables[0] = new SqlParameter("@QT QueryID" ,
QueryID ) ;
                                sqlQueryTables[1] = new SqlParameter("@QT TableName" ,
r["QT_TableName"] ) ;
                                sqlQueryTables[2] = new SqlParameter("@QT_Relations" ,
r["QT Relations"] ) ;


        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"QQ Q
UERYTABLE_INSERT_SP",CommandType.StoredProcedure, sqlQueryTables) ;
                        }
                }
                #endregion
                #region "QueryFields"
                //now that we have  inserted queryID, insert the new..
                if( queryDataSet.QueryField.Rows.Count > 0  ) {
                        foreach( DataRow r in queryDataSet.QueryField.Rows  ) {
                                SqlParameter[] sqlQueryFields = new SqlParameter[11] ;
                                sqlQueryFields[0] = new SqlParameter("@QF QueryID" ,
QueryID ) ;
                                sqlQueryFields[1] = new SqlParameter("@QF_Name" ,
r["QF_Name"] )
                                sqlQueryFields[2] = new SqlParameter("@QF Title" ,
r["QF Title"] ) ;
                                sqlQueryFields[3] = new SqlParameter("@QF Url" ,
r["QF Url"] ) ;
                                sqlQueryFields[4] = new SqlParameter("@QF_UrlTarget" ,
r["QF_UrlTarget"] ) ;
                                sqlQueryFields[5] = new SqlParameter("@QF DetailQueryID"
, r["QF DetailQueryID"] ) ;
                                sqlQueryFields[6] = new
SqlParameter("@QF LookUpTableName" , r["QF LookUpTableName"] ) ;
                                sqlQueryFields[7] = new SqlParameter("@QF_LookUpSQL" ,
r["QF_LookUpSQL"] ) ;
                                sqlQueryFields[8] = new
SqlParameter("@QF LookUpResultField" , r["QF LookUpResultField"] ) ;
                                sqlQueryFields[9] = new
SqlParameter("@QF_LookUpConnectionField" , r["QF_LookUpConnectionField"] ) ;
                                sqlQueryFields[10] = new
SqlParameter("@QF PossibleCombinations" , r["QF PossibleCombinations"] ) ;


        DataManager.ExecuteNonQuery(ConfigurationManager.DatabaseConnectionString,"QQ Q
UERYFIELD_INSERT_SP",CommandType.StoredProcedure, sqlQueryFields) ;
                        }
                }
                #endregion
        }

        #endregion
}
```

Ένα άλλο σημείο που αξίζει να αναφερθούμε στο σημείο αυτό έχει να κάνει με την προβολή των διαθέσιμων πινάκων πεδίων και σχέσεων για την Β.Δ. όπως μπορούμε και από την παραπάνω εικόνα, υπάρχει ένα τμήμα το οποίο μας βοηθάει να δημιουργήσουμε το ερώτημα μας. Αυτό το τμήμα είναι ο Query Builder. Όπως έχουμε αναφέρει αποτελείται απο ένα υποτμήμα που είναι οι διαθέσιμοι και οι επιλεγμένοι πίνακες, ένα άλλο που είναι τα διαθέσιμα πεδία, βάσει του επιλεγμένου πίνακα και τα επιλεγμένα πεδία και οι διαθέσιμες και επιλεγμένες σχέσεις μεταξύ πινάκων.

Ας εξετάσουμε τώρα πως εμφανίζονται οι διαθέσιμοι πίνακες. Αυτό που κάνουμε ουσιαστικά είναι οτι λαμβάνουμε το σχήμα της Β.Δ. όπως φαίνεται στο κώδικα που παραθέτουμε.

```
private void CustomBindAvailableTables(){
        DataTable schemaTable = null;
        //QueryGroupID =
(int)queryDataSet.Query.Select("QueryID="+QueryID)[0]["Q_QueryGroupID"];
        #region "OleDbSchema"
        if( DBTypeID == 1 ) {
                //get table list from database
                OleDbConnection oleDb = new OleDbConnection( DBConn );
                oleDb.Open();
                schemaTable = oleDb.GetOleDbSchemaTable( OleDbSchemaGuid.Tables, new
object[] {null,null,null,"TABLE"});
        }
        #endregion
        #region "SqlDbSchema"
        else if( DBTypeID == 2 ) {
                schemaTable = DataManager.FillDataGrid( DBConn, "SELECT ID,NAME AS
'TABLE NAME' FROM dbo.sysobjects WHERE xtype = 'U'", System.Data.CommandType.Text);
        }
        #endregion
        //now Available Table ListBox
        BindListBox( lbAvailableTables, schemaTable, "TABLE_NAME", "TABLE_NAME");
}
```

Αναλόγως τον τύπο της Β.Δ. που έχουμε μπορούμε να πάρουμε το σχήμα της με τον αντίστοιχο τρόπο. Αν για παράδειγμα είναι τύπου MS Access, τότε μέσω της κλάσης OleDB μπορούμε να ανακτήσουμε την πληροφορία αυτή.

Με παρόμοιο τρόπο, αλλά θέτοντας διαφορετικούς παραμέτρους, γίνεται η ανάκτηση της πληροφορίας για τα διαθέσιμα πεδία ενός πίνακα όπως φαίνεται παρακάτω.

```
private void CustomBindAvailableFields(){
        DataTable schemaTable = new DataTable();
        DataColumn dCol = new DataColumn("COLUMN NAME");
        schemaTable.Columns.Add(dCol);

        #region "OleDbSchema"
        if( DBTypeID == 1 ) {
                //get table list from database
                OleDbConnection oleDb = new OleDbConnection( DBConn );
                oleDb.Open();
                DataTable curDataTable=null;
                foreach( DataRow dRow in queryDataSet.QueryTable.Rows ){
                        curDataTable = oleDb.GetOleDbSchemaTable(
OleDbSchemaGuid.Columns, new object[]
{null,null,dRow["QT TableName"].ToString(),null});
                        foreach( DataRow curRow in curDataTable.Rows ){
                                DataRow schemaRow = schemaTable.NewRow();
                                schemaRow["COLUMN_NAME"] =
dRow["QT_TableName"].ToString()+"."+curRow["COLUMN_NAME"].ToString();
                                schemaTable.Rows.Add(schemaRow);
                        }
                }
        }
        #endregion
        #region "SqlDbSchema"
        else if( DBTypeID == 2 ) {
                string tableNames=string.Empty;
                if( queryDataSet.QueryTable.Rows.Count>0 ){
                        for( int j=1; j<queryDataSet.QueryTable.Rows.Count; j++ ){
                                tableNames += string.Format(" OR dbo.sysobjects.name =
```

```
'{0}' ", queryDataSet.QueryTable.Rows[j]["QT_TableName"]);
                        }
                        string sql = "SELECT
dbo.syscolumns.id,dbo.sysobjects.NAME+'.'+dbo.syscolumns.name AS 'COLUMN NAME' FROM
dbo.sysobjects ";
                        sql += " INNER JOIN dbo.syscolumns ON dbo.sysobjects.id =
dbo.syscolumns.id ";
                        sql += " WHERE dbo.sysobjects.xtype = 'U' and (
dbo.sysobjects.name = '" + queryDataSet.QueryTable.Rows[0]["QT TableName"].ToString()
+ "' " + tableNames + ")";
                        schemaTable = DataManager.FillDataGrid( DBConn, sql,
System.Data.CommandType.Text);
                }
        }
        #endregion

        //now bind
        BindListBox( lbAvailableFields, schemaTable, "COLUMN_NAME", "COLUMN_NAME");
}
```

Η λίστα των πεδίων και των παραμέτρων βασίζεται στην λογική της Σύνθετης λίστας που έχουμε αναφέρει προηγουμένως και δεν θεωρούμε οτι χρειάζεται να πούμε κάτι παραπάνω. Θα μπορούσαμε επίσης να αναφερθούμε σε πολλά άλλα πράματα που έχει η σελίδα αυτή, αλλά θεωρούμε οτι τα πιο σημαντικά είναι αυτά τα οποία έχουμε ήδη αναφέρει.

4.4.4 Σελίδα Εκτέλεσης Ερωτήματος

Την λειτουργία της εκτέλεσης ερωτήματος την έχουμε περιγράψει στο κεφάλαιο 3.4.2.4 Φόρμα Εκτέλεσης Ερωτημάτων. Θα περιγράψουμε τώρα πως έχουν υλοποιηθεί οι λειτουργίες αυτές. Η σελίδα αποτελείται από τα εξής μέρη:

- Επικεφαλίδα

Έχουμε αναφερθεί ήδη αρκετές φορές.

- Τίτλος ερωτήματος

Είναι ένα σημείο στο οποίο εμφανίζουμε το τίτλο του ερωτήματος

- Παράμετροι Ερωτήματος

Αν το ερώτημα απαρτίζεται απο παραμέτρους τότε αυτοί εμφανίζονται δυναμικά όπως φαίνεται παρακάτω. Ουσιαστικά διαβάζει τις παραμέτρους από τον πίνακα των παραμέτρων και αναλόγως των ιδιοτήτων που έχει κάθε παράμετρος δημιουργεί το αντίστοιχο control.

```
private void InitializeParamControls(){
        if( qDs.Tables["QueryParam"].Rows.Count > 0 ){
                foreach(DataRow r in qDs.Tables["QueryParam"].Rows){
                        CreateControl(r);
                }
        }
}

private void CreateControl(DataRow row){
        Label lbl = null;
        TextBox txt = null;
        DropDownList ddl = null;

        HtmlGenericControl br = new HtmlGenericControl("br");

        #region "Boolean DropDownList"
        DataTable BooleanDataTable = new DataTable();
        BooleanDataTable.TableName = "BooleanDataTable";
        BooleanDataTable.Columns.Add("ID");
        BooleanDataTable.Columns.Add("Description");

        DataRow dRow = null;
        //1
        dRow = BooleanDataTable.NewRow();
        dRow["ID"] = "1";
        dRow["Description"] = "NAI";
```

```
        BooleanDataTable.Rows.Add(dRow);
        //2
        dRow = BooleanDataTable.NewRow();
        dRow["ID"] = "2";
        dRow["Description"] = "OXI";
        BooleanDataTable.Rows.Add(dRow);
        #endregion
        //Now check if this param has any properies(lookUp properties)
        //1. Bind DropDownList with a table from DB.
        #region "Bind DropDownList with a table from DB."
        if( !row["QP_LookUpTableName"].Equals("") &&
!row["QP_LookUpTableName"].Equals(System.DBNull.Value)){
                lbl = new Label();
                lbl.Text = row["QP Caption"].ToString() + " " ;
                lbl.CssClass = "lblTitle";
                string selectCmd = String.Format("SELECT * FROM
{0}",row["QP_LookUpTableName"].ToString());
                DataTable dt =
DataManager.FillDataGrid(ConfigurationManager.DatabaseConnectionString, selectCmd,
CommandType.Text);
                dt.TableName = "tempTable";
                //create lookUp field
                ddl = new DropDownList();
                ddl.ID = row["QP_Name"].ToString();
                ddl.DataSource = dt;
                ddl.DataValueField = row["QP LookUpConnectionField"].ToString();
                ddl.DataTextField = row["QP LookUpField"].ToString();
                ddl.DataBind();
                //set default value
                if( dt.Rows.Count>0 ){
                        string selectExp =
string.Format("{0}={1}",row["QP LookUpConnectionField"].ToString(),
row["QP DefaultValue"].ToString());
                        DataRow[] rows = dt.Select(selectExp);
                        if( rows.Length>0 ){
                                ddl.SelectedValue = rows[0][0].ToString();
                        }
                }
                //now add them
                ParamPanel.Controls.Add(lbl);
                ParamPanel.Controls.Add(ddl);
                ParamPanel.Controls.Add(br);

        }
        #endregion
        //2. Bind DropDownList with an executable sql query
        #region "Bind DropDownList with an executable sql query"
        else if( !row["QP LookUpSql"].Equals("") &&
!row["QP LookUpSql"].Equals(System.DBNull.Value)){
                lbl = new Label();
                lbl.Text = row["QP Caption"].ToString() + " " ;
                lbl.CssClass = "lblTitle";
                //Now it's time to get the fields from query. The logic is:
                // 1. Seperate fields from the rest "select statement", based on words
SELECT and FROM
                // 2. Split fields based on ","
                string originalSql = row["QP LookUpSql"].ToString().ToUpper();
                string editSql = originalSql.Substring(
originalSql.IndexOf("SELECT")+"SELECT".Length,originalSql.IndexOf("FROM")-
"SELECT".Length );
                string[] middleSql = editSql.Split(new char[]{','});
                //continue as usual...
                string selectCmd = row["QP_LookUpSql"].ToString();
                DataTable dt =
DataManager.FillDataGrid(ConfigurationManager.DatabaseConnectionString, selectCmd,
CommandType.Text);
                dt.TableName = "tempTable";
                //create lookUp field
                ddl = new DropDownList();
                ddl.ID = row["QP_Name"].ToString();
                ddl.DataSource = dt;
                ddl.DataValueField = middleSql[0].Trim();
                ddl.DataTextField = middleSql[1].Trim();
                ddl.DataBind();
                //set default value
                if( dt.Rows.Count>0 ){
                        string selectExp = string.Format("{0}={1}",middleSql[0],
```

```
row["QP_DefaultValue"].ToString());
                        DataRow[] rows = dt.Select(selectExp);
                        if( rows.Length>0 ){
                                ddl.SelectedValue = rows[0][0].ToString();
                        }
                }
                //now add them
                ParamPanel.Controls.Add(lbl);
                ParamPanel.Controls.Add(ddl);
                ParamPanel.Controls.Add(br);

        }
        #endregion
        //3. If it doesn't have any lookup properties
        #region "Add query param control based on his type"
        else{
                switch( (int)row["QP_TypeID"] ){
                        //initialize

                        //text
                        case 1:
                        case 2:
                        case 3:
                        case 4:
                                lbl = new Label();
                                lbl.Text = row["QP Caption"].ToString() + " " ;
                                lbl.CssClass = "lblTitle";
                                txt = new TextBox();
                                txt.ID = row["QP_Name"].ToString();
                                txt.Text = row["QP_DefaultValue"].ToString();
                                txt.CssClass = "txtStyle2";

                                ParamPanel.Controls.Add(lbl);
                                ParamPanel.Controls.Add(txt);
                                ParamPanel.Controls.Add(br);

                                break;
                        case 5:
                                lbl = new Label();
                                lbl.Text = row["QP_Caption"].ToString() + " " ;
                                lbl.CssClass = "lblTitle";

                                ddl = new DropDownList();
                                ddl.ID = row["QP Name"].ToString();
                                ddl.DataSource = BooleanDataTable;
                                ddl.DataTextField = "Description";
                                ddl.DataValueField = "ID";
                                ddl.DataBind();
                                ddl.SelectedValue = row["QP DefaultValue"].ToString();
                                ddl.CssClass = "txtStyle2";

                                ParamPanel.Controls.Add(lbl);
                                ParamPanel.Controls.Add(ddl);
                                ParamPanel.Controls.Add(br);

                                break;
                }
                #endregion
        }
}
```

- Επιλογή Επεξεργασία

Ουσιαστικά μάς παραπέμπει στην σελίδα επεξεργασία του ερωτήματος

- Επιλογή Εκτέλεση Ερωτήματος

Όπως φαίνεται από τον κώδικα που έχουμε παραθέσει, κατά την εκτέλεση του ερωτήματος παίρνουμε τις τιμές απο τις παραμέτρους και βάσει του τύπου του αποτελέσματος που θέλουμε καλούμε το αντίστοιχο UserControl για να εκτελέσει το ερώτημα και να μας επιστρέψει το αποτέλεσμα. Για τον τρόπο λειτουργίας του κάθε UserControl θα αναφερθούμε στην πορεία.

```
private void exec( string statType ){
        try{
        //get values from params into a dataTable
          DataTable paramValuesTbl = new DataTable();
          paramValuesTbl.Columns.Add("QP Name");
          paramValuesTbl.Columns.Add("QP_DefaultValue");
          //paramValuesTbl.Columns.Add("paramType");
          DataRow paramRow=null;
          foreach(Control ctrl in ParamPanel.Controls){
                if( (ctrl.GetType()==typeof(TextBox)) ||
(ctrl.GetType()==typeof(DropDownList)) ){
                        paramRow = paramValuesTbl.NewRow();
                        paramRow["QP Name"] = ctrl.ID;
                        if( ctrl.GetType()==typeof(TextBox) )
                                paramRow["QP DefaultValue"] = ((TextBox)ctrl).Text;
                        if(ctrl.GetType()==typeof(DropDownList))
                                paramRow["QP_DefaultValue"] =
((DropDownList)ctrl).SelectedValue;

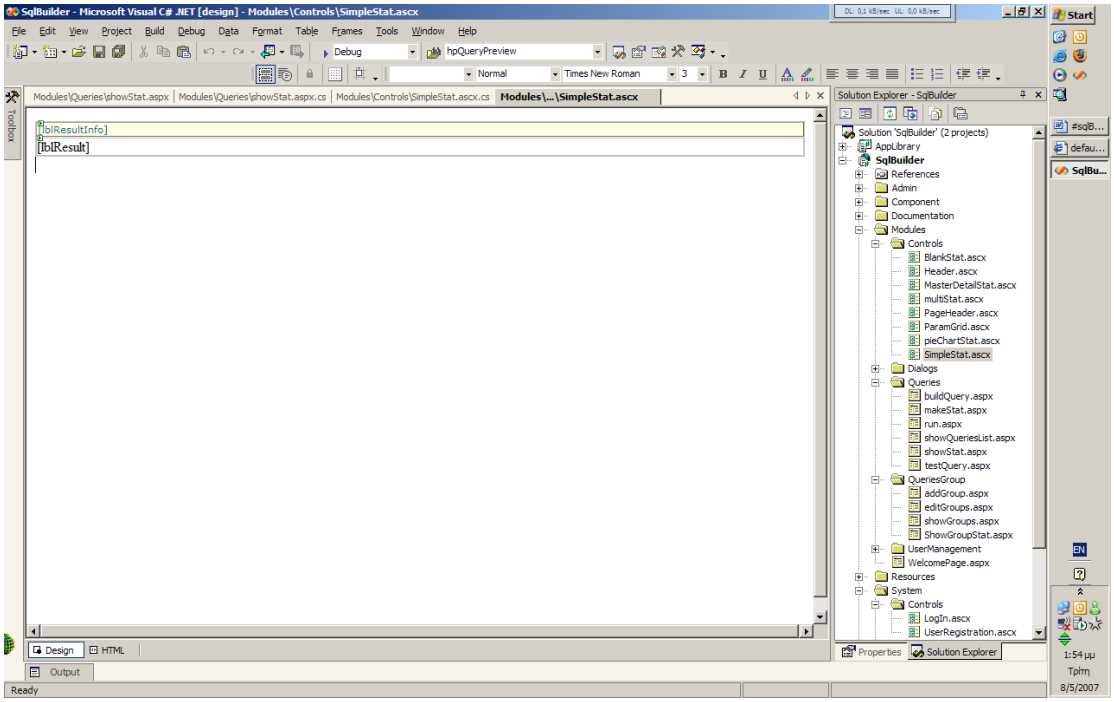
                        paramValuesTbl.Rows.Add(paramRow);
                        }
                                //paramRow["paramType"] = ;
                }
                BlankStat bs =
(BlankStat)this.LoadControl(String.Format("../Controls/{0}.ascx",statType));
                bs.GetQueryInfo( QueryID,paramValuesTbl );
                if( statType == "multiStat" )
                        DataGridResult = bs.DataGridResult();
                result.Controls.Add(bs);
        }
        catch(Exception ex){
                //clear previous error messages
                errorReport.Text = string.Empty;
                //continue
                errorReport.Text += " Μήνυμα Συστήματος : " + ex.Message ;
                errorReport.Text += "<br> Stack Trace : " + ex.StackTrace ;
                errorReport.Visible = true;
        }
}
```

4.5 Εκτέλεση Ερωτήματος και User Controls

Όπως αναφέραμε σε προηγούμενη ενότητα κατά την εκτέλεση ενός ερωτήματος βάσει του τύπου του αποτελέσματος που έχουμε ορίσει οτι θέλουμε καλείται και το αντίστοιχο User Control για την εκτέλεση. Ας δούμε πια είναι αυτα τα user controls που χρησιμοποιούμε στην εκτέλεση.

4.5.1 Απλό – SimpleStat User Control

Όπως μπορούμε να δούμε και απο την εικόνα 23 αυτο το UserControl αποτελείται μόνο από δύο Labels, στα οποία αποτυπώνονται πληροφορίες σχετικά με το χρόνο εκτέλεσης του ερωτήματος και το αποτέλεσμα του.



Εικόνα 25. Απλό - SimpleStat UserControl

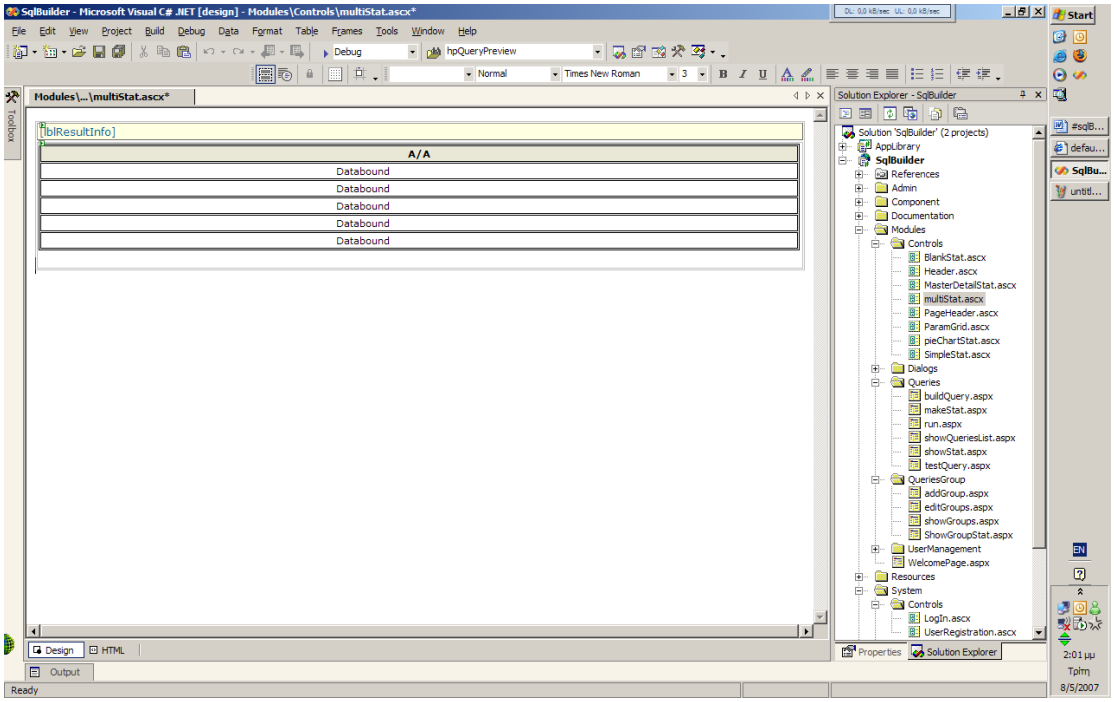
Κατά την εκτέλεση αυτού του τύπου ερωτήματος αποθηκεύουμε σε μια μεταβλητή τον χρόνο στον οποίο άρχισε η εκτέλεση του. Στην συνέχεια εκτελούμε το ερώτημα με την χρήση κάποιων μεθόδων που έχουμε δημιουργήσει στην βιβλιοθήκη μας και μετά την εκτέλεση του ερωτήματος αποθηκεύουμε ξανά τον νέο χρόνο λήξης εκτέλεσης του ερωτήματος και υπολογίζουμε την διαφορά τους.

```
public override void GetQueryInfo( int queryID, DataTable paramTable ){
        //get initial time
        int startTime = DateTime.Now.Millisecond;
        QueryDataSet queryDataSet = (QueryDataSet)Session["queryDataSet"] ;
        //exec query
        lblResult.Text = DataManager.ExecuteSimpleStatQuery(queryID, queryDataSet,
paramTable);
        //get final time
        int endTime = DateTime.Now.Millisecond;
        //show infos
        lblResultInfo.Text = string.Format("Αποτελέσμα - Χρόνος Εκτέλεσης : {0}
δευτ/πτα.",((endTime-startTime)/60.0f) );

}
```

4.5.2 Πολλαπλό – MultiStat User Control

Το Control αυτό όπως μπορείτε να δείτε απο την εικόνα 24 αποτελείται απο ένα Label στο οποίο αποτυπώνουμε τις πληροφορίες εκτέλεσης του ερωτήματος και από ένα DataGrid στο οποίο αποτυπώνουμε το αποτέλεσμα της εκτέλεσης του ερωτήματος. Στην λίστα αυτή, οι στήλες της δημιουργούνται δυναμικά μέσα από τον κώδικα, όπως έχουμε αναφέρει οτι γίνεται.



Εικόνα 26. Πολλαπλό – MultiStat User Control

Η εκτέλεση του πολλαπλού τύπου είναι επίσης σχετικά απλή. Όπως φαίνεται και απο τον κώδικα που έχουμε παραθέσει τα βήματα είναι τα εξής:

1. Εκτελούμε το ερώτημα
2. Έλεγχος πεδίων που πρέπει να εμφανιστούν
3. Προετοιμασία της λίστας
4. Εμφάνιση αποτελέσματος

```
public override void GetQueryInfo( int queryID, DataTable paramTable ){
  //We want also to count the time of execution
  DateTime startTime = DateTime.Now;
  //Get important info such as connectionString
  queryDataSet = (QueryDataSet)Session["queryDataSet"] ;
  //This is a method that executesquery based on database type (Access,MSSQL,etc)
  ResultDataTable = DataManager.ExecuteMultiStatQuery(queryID, queryDataSet,
paramTable);
  #region "show specific fields"
  if( ResultDataTable!=null ){
    if( queryDataSet.Tables["QueryField"].Rows.Count > 0 ){
      string qfName = string.Empty;
      DataRow[] rows =
queryDataSet.Tables["QueryField"].Select("QF_QueryID="+queryID);
      foreach(DataRow r in rows ){
        //after query builder i insert fields with tablename, seperated by ".". So
clear them and procced
          if( r["QF Name"].ToString().IndexOf(".")>0 )
            qfName =
r["QF_Name"].ToString().Substring(r["QF_Name"].ToString().IndexOf(".")+1);
          else
            qfName = r["QF Name"].ToString();

          if( ResultDataTable.Columns.Contains( qfName ) ){
            ResultDataTable.Columns[qfName].ColumnName = qfName;
            ResultDataTable.Columns[qfName].Caption = r["QF Title"].ToString();
          }
          else
            ResultDataTable.Columns.Remove(qfName);
        }
      }
    }
  #endregion
  //Now that we have the columns that we will show, let's create our
```

```
  //datagrid with the appropriate type of columns
  DataGridManager.prepareGrid(dgResult, queryDataSet, ResultDataTable,queryID );
  dgResult.DataBind();
  //get end time
  DateTime endTime = DateTime.Now;
  int RowCount = (ResultDataTable!=null)?ResultDataTable.Rows.Count:0;
  TimeSpan timeCount = endTime-startTime;
  lblResultInfo.Text = string.Format("Αποτελέσματα [{0}] - Χρόνος Εκτέλεσης : [{1}]
δευτερόλεπτα.",RowCount , (timeCount.Milliseconds/100.0) );

}
```

Το μόνο βήμα που αξίζει να αναφερθούμε είναι η προετοιμασία της λίστας. Αυτό γίνεται σύμφωνα με τον κώδικα που παραθέτουμε. Όπως έχουμε αναφέρει κατά την λειτουργική περιγραφή της εφαρμογής στο κεφάλαιο

3.4.2.3 Φόρμα Δημιουργίας και Επεξεργασίας Ερωτήματος, μπορούμε να θέσουμε κάποιες ιδιότητες στα πεδία. Τώρα για να μπορέσουμε να δούμε τις ιδιότητες που έχουμε θέσει έχουμε γράψει τον αντίστοιχο κώδικα.

```
public static void prepareGrid(DataGrid dg, QueryDataSet srcDs, DataTable
gridsDataSource, int queryId){
        //loop into table to get queryfield prperties, if it has
        bool isDateTime=false;
        bool hasBeenCreated = false;// this boolean variable is used because we want to
know if this column has been created, so it won't pass through another process
        DataGridColumn dgc=null;

        DataRow[] rows = srcDs.Tables["QueryField"].Select("QF QueryID="+queryId);
        foreach(DataRow r in rows){
                hasBeenCreated = false;
                //check if field is datetime field
                string qfName = string.Empty;

        if(!gridsDataSource.Columns.Contains(r["QF Title"].ToString())){
                if( r["QF_Name"].ToString().IndexOf(".")>0 )
                        qfName =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                else
                        qfName = r["QF Name"].ToString();
                if( gridsDataSource.Columns.Contains(qfName) ){
                        if( gridsDataSource.Columns[qfName].DataType ==
typeof(DateTime)){
                                isDateTime = true;
                        }
                }
        }

        //now go on. Based on properties create suitables gridcolumns
        int switchCase = Utils.GetSafeIntValue(r["QF PossibleCombinations"]);
        switch(switchCase){
                case 1:
                #region "case 1. Hyper Link"
                        dgc = new HyperLinkColumn();
                        HyperLinkColumn hpCol2 = dgc as HyperLinkColumn;
                        if( r["QF Name"].ToString().IndexOf(".")>0 )
                                hpCol2.DataTextField =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                        else
                                hpCol2.DataTextField = r["QF_Name"].ToString();
                        hpCol2.HeaderText = r["QF Title"].ToString();
                        hpCol2.NavigateUrl = r["QF Url"].ToString();
                        hpCol2.Target = r["QF UrlTarget"].ToString();
                        if(isDateTime){
                                hpCol2.DataTextFormatString = "{0:dd/MM/yyyy}";
                        }
                #endregion
                        break;
                case 2:
                #region "case2. SubQuery link"
                #region filldataset
                QueryDataSet qDs = new QueryDataSet();
                //queryparamtype

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ QUERYPARAMTYPE SELECT SP", CommandType.StoredProcedure, qDs , "QueryParamType" ) ;
                //StatType

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ STATTYPE SELECT SP", CommandType.StoredProcedure, qDs , "StatType" ) ;
                //get userid
                int userId =
int.Parse(DataManager.ExecuteScalar(ConfigurationManager.DatabaseConnectionString ,
"SELECT Q CreatedByUserID FROM QQ Query WHERE QueryID="+(int)r["QF DetailQueryID"]));
                //QueryGroup
                SqlParameter[] param = new SqlParameter[1] ;
                param[0] = new SqlParameter("@QG CreatedByUserID", userId ) ;

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ QUERYGROUP SELECTBYKEY SP", CommandType.StoredProcedure, param , qDs ,
"QueryGroup" ) ;
                //Query
```

```
                param = new SqlParameter[1] ;
                param[0] = new SqlParameter("@QueryID", (int)r["QF DetailQueryID"]) ;

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ QUERY SELECTBYKEY SP", CommandType.StoredProcedure, param , qDs , "Query" ) ;
                //QueryParams
                param = new SqlParameter[1] ;
                param[0] = new SqlParameter("@QP QueryID", (int)r["QF DetailQueryID"])
;

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ_QUERYPARAM_SELECTBYKEY_SP", CommandType.StoredProcedure, param , qDs ,
"QueryParam" ) ;
                #region "Fill Query Tables"
                param = new SqlParameter[1] ;
                param[0] = new SqlParameter("@QT QueryID", (int)r["QF DetailQueryID"])
;

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ QUERYTABLE SELECTBYKEY SP", CommandType.StoredProcedure, param , qDs ,
"QueryTable" ) ;
                #endregion
                //QueryFields
                param = new SqlParameter[1] ;
                param[0] = new SqlParameter("@QF QueryID", (int)r["QF DetailQueryID"])
;

        DataManager.FillDataSet(ConfigurationManager.DatabaseConnectionString ,
"QQ_QUERYFIELD_SELECTBYKEY_SP", CommandType.StoredProcedure, param , qDs ,
"QueryField" ) ;
                #endregion
                dgc = new MasterDetailColumn();
                MasterDetailColumn mcol = dgc as MasterDetailColumn;
                if( r["QF Name"].ToString().IndexOf(".")>0 )
                        mcol.DataField =
r["QF_Name"].ToString().Substring(r["QF_Name"].ToString().IndexOf(".")+1);
                else
                        mcol.DataField = r["QF Name"].ToString();

                mcol.ChildField = r["QF_LookUpConnectionField"].ToString();
                mcol.HeaderText = r["QF_Title"].ToString();
                mcol.QQ QUERY ID = (int)r["QF DetailQueryID"];
                mcol.qDs = qDs;
                //in order to avoid crazy numbers about response, we need to adjust
query to bring data based on
                //current row datafield's value
                if (isDateTime) {
                        mcol.DataFormatString = "{0:dd/MM/yyyy}";
                }
                #endregion
                break;
        case 3:
        #region "case 3. Look Up Table"
                dgc = new LookupColumn();
                LookupColumn bCol = dgc as LookupColumn;
                if( r["QF Name"].ToString().IndexOf(".")>0 )
                        bCol.DataField =
r["QF_Name"].ToString().Substring(r["QF_Name"].ToString().IndexOf(".")+1);
                else
                        bCol.DataField = r["QF Name"].ToString();

                bCol.HeaderText = r["QF Title"].ToString();

                string sqlCmd = string.Format("SELECT * FROM {0}",
r["QF LookUpTableName"].ToString());
                int qID = (int)r["QF QueryID"];
                int qgID =
(int)srcDs.Tables["Query"].Select("QueryID="+qID)[0]["Q QueryGroupID"];
                string clientConnectionString =
srcDs.Tables["QueryGroup"].Select("QueryGroupID="+qgID)[0]["QG_ConnectionString"].ToSt
ring();

                bCol.LookupTable =
DataManager.FillDataGrid(clientConnectionString,sqlCmd, CommandType.Text);
                                                bCol.LookupConnectionField =
r["QF_LookUpConnectionField"].ToString();
                bCol.LookupResultField = r["QF_LookUpResultField"].ToString();
```

```
                if(isDateTime){
                        bCol.DataFormatString = "{0:dd/MM/yyyy}";
                }
        #endregion
                break;
        case 4:
        #region "case 4. Look Up SQL"
                dgc = new LookupColumn();
                LookupColumn bCol4 = dgc as LookupColumn;
                if( r["QF_Name"].ToString().IndexOf(".")>0 )
                        bCol4.DataField =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                else
                        bCol4.DataField = r["QF Name"].ToString();

                bCol4.HeaderText = r["QF_Title"].ToString();

                string lookupSQL = r["QF LookUpSQL"].ToString();
                int lookupSqlQID = (int)r["QF QueryID"];
                int lookupSqlQgID =
(int)srcDs.Tables["Query"].Select("QueryID="+lookupSqlQID)[0]["Q QueryGroupID"];
                string lookupSQLclientConnectionString =
srcDs.Tables["QueryGroup"].Select("QueryGroupID="+lookupSqlQgID)[0]["QG_ConnectionStri
ng"].ToString();
                //Now it's time to get the fields from query. The logic is:
                // 1. Seperate fields from the rest "select statement", based on words
SELECT and FROM
                // 2. Split fields based on ","
                string editSql4 = lookupSQL.ToUpper().Substring(
lookupSQL.IndexOf("SELECT")+"SELECT".Length,lookupSQL.IndexOf("FROM")-"SELECT".Length
);
                string[] middleSql4 = editSql4.Split(new char[]{','});

                bCol4.LookupTable =
DataManager.FillDataGrid(lookupSQLclientConnectionString,lookupSQL, CommandType.Text);
                bCol4.LookupConnectionField = middleSql4[0].Trim();
                bCol4.LookupResultField = middleSql4[1].Trim();

                if(isDateTime){
                        bCol4.DataFormatString = "{0:dd/MM/yyyy}";
                }
        #endregion
                break;
        case 5:
        #region "case 5. Hyper Link + Look Up Table"
                dgc = new HyperLinkLookupColumn();
                HyperLinkLookupColumn hplCol7 = dgc as HyperLinkLookupColumn;
                if( r["QF Name"].ToString().IndexOf(".")>0 )
                        hplCol7.DataTextField =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                else
                        hplCol7.DataTextField = r["QF_Name"].ToString();

                hplCol7.HeaderText = r["QF Title"].ToString();

                string hplCol7SqlCmd = string.Format("SELECT * FROM {0}",
r["QF_LookUpTableName"].ToString());
                int hplCol7qID = (int)r["QF_QueryID"];
                int hplCol7qgID =
(int)srcDs.Tables["Query"].Select("QueryID="+hplCol7qID)[0]["Q QueryGroupID"];
                string hplCol7CCS =
srcDs.Tables["QueryGroup"].Select("QueryGroupID="+hplCol7qgID)[0]["QG_ConnectionString
"].ToString();

                hplCol7.LookupTable =
DataManager.FillDataGrid(hplCol7CCS,hplCol7SqlCmd, CommandType.Text);
                hplCol7.LookupConnectionField =
r["QF_LookUpConnectionField"].ToString();
                hplCol7.LookupResultField = r["QF_LookUpResultField"].ToString();

                hplCol7.MyNavigateUrl = r["QF Url"].ToString();
                hplCol7.MyTarget = r["QF UrlTarget"].ToString();
                if(isDateTime){
                        hplCol7.DataTextFormatString = "{0:dd/MM/yyyy}";
                }
        #endregion
```

```
                break;
        case 6:
        #region "case 6. Hyper Link + Look Up SQL"
                dgc = new HyperLinkLookupColumn();
                HyperLinkLookupColumn hplCol8 = dgc as HyperLinkLookupColumn;
                if( r["QF_Name"].ToString().IndexOf(".")>0 )
                        hplCol8.DataTextField =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                else
                        hplCol8.DataTextField = r["QF Name"].ToString();

                hplCol8.HeaderText = r["QF_Title"].ToString();

                string hplCol8SqlCmd = r["QF LookUpSQL"].ToString();
                int hplCol8qID = (int)r["QF QueryID"];
                int hplCol8qgID =
(int)srcDs.Tables["Query"].Select("QueryID="+hplCol8qID)[0]["Q_QueryGroupID"];
                string hplCol8CCS =
srcDs.Tables["QueryGroup"].Select("QueryGroupID="+hplCol8qgID)[0]["QG ConnectionString
"].ToString();
                //Now it's time to get the fields from query. The logic is:
                // 1. Seperate fields from the rest "select statement", based on words
SELECT and FROM
                // 2. Split fields based on ","
                string editSql6 = hplCol8SqlCmd.ToUpper().Substring(
hplCol8SqlCmd.IndexOf("SELECT")+"SELECT".Length,hplCol8SqlCmd.IndexOf("FROM")-
"SELECT".Length );
                string[] middleSql6 = editSql6.Split(new char[]{','});

                hplCol8.LookupTable =
DataManager.FillDataGrid(hplCol8CCS,hplCol8SqlCmd, CommandType.Text);
                hplCol8.LookupConnectionField = middleSql6[0].Trim();
                hplCol8.LookupResultField = middleSql6[1].Trim();

                hplCol8.MyNavigateUrl = r["QF_Url"].ToString();
                hplCol8.MyTarget = r["QF_UrlTarget"].ToString();
                if(isDateTime){
                        hplCol8.DataTextFormatString = "{0:dd/MM/yyyy}";
                }
        #endregion
                break;
        case 7:
        #region "case 7. SubQuery link + Look Up Table"
                dgc = new HyperLinkLookupColumn();
                HyperLinkLookupColumn hplCol5 = dgc as HyperLinkLookupColumn;
                if( r["QF_Name"].ToString().IndexOf(".")>0 )
                        hplCol5.DataTextField =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                else
                        hplCol5.DataTextField = r["QF Name"].ToString();

                hplCol5.HeaderText = r["QF_Title"].ToString();

                string hplCol5SqlCmd = string.Format("SELECT * FROM {0}",
r["QF LookUpTableName"].ToString());
                int hplCol5qID = (int)r["QF QueryID"];
                int hplCol5qgID =
(int)srcDs.Tables["Query"].Select("QueryID="+hplCol5qID)[0]["Q_QueryGroupID"];
                string hplCol5CCS =
srcDs.Tables["QueryGroup"].Select("QueryGroupID="+hplCol5qgID)[0]["QG ConnectionString
"].ToString();

                hplCol5.LookupTable =
DataManager.FillDataGrid(hplCol5CCS,hplCol5SqlCmd, CommandType.Text);
                hplCol5.LookupConnectionField =
r["QF LookUpConnectionField"].ToString();
                hplCol5.LookupResultField = r["QF LookUpResultField"].ToString();

                hplCol5.MyNavigateUrl =
String.Format("../../Modules/Queries/showStat.aspx?QueryID={0}",r["QF_DetailQueryID"])
;
                hplCol5.MyTarget = r["QF UrlTarget"].ToString();
                if(isDateTime){
                        hplCol5.DataTextFormatString = "{0:dd/MM/yyyy}";
                }
        #endregion
                break;
```

```
        case 8:
        #region "case 8. SubQuery link + Look Up SQL"
                dgc = new HyperLinkLookupColumn();
                HyperLinkLookupColumn hplCol6 = dgc as HyperLinkLookupColumn;
                if( r["QF Name"].ToString().IndexOf(".")>0 )
                        hplCol6.DataTextField =
r["QF_Name"].ToString().Substring(r["QF_Name"].ToString().IndexOf(".")+1);
                else
                        hplCol6.DataTextField = r["QF Name"].ToString();

                hplCol6.HeaderText = r["QF_Title"].ToString();

                string hplCol6SqlCmd = r["QF LookUpSQL"].ToString();
                int hplCol6qID = (int)r["QF QueryID"];
                int hplCol6qgID =
(int)srcDs.Tables["Query"].Select("QueryID="+hplCol6qID)[0]["Q QueryGroupID"];
                string hplCol6CCS =
srcDs.Tables["QueryGroup"].Select("QueryGroupID="+hplCol6qgID)[0]["QG_ConnectionString
"].ToString();
                //Now it's time to get the fields from query. The logic is:
                // 1. Seperate fields from the rest "select statement", based on words
SELECT and FROM
                // 2. Split fields based on ","
                string editSql8 = hplCol6SqlCmd.ToUpper().Substring(
hplCol6SqlCmd.IndexOf("SELECT")+"SELECT".Length,hplCol6SqlCmd.IndexOf("FROM")-
"SELECT".Length );
                string[] middleSql8 = editSql8.Split(new char[]{','});

                hplCol6.LookupTable =
DataManager.FillDataGrid(hplCol6CCS,hplCol6SqlCmd, CommandType.Text);
                hplCol6.LookupConnectionField = middleSql8[0].Trim();
                hplCol6.LookupResultField = middleSql8[1].Trim();

                hplCol6.MyNavigateUrl =
String.Format("../../Modules/Queries/showStat.aspx?QueryID={0}",r["QF_DetailQueryID"])
;
                hplCol6.MyTarget = r["QF UrlTarget"].ToString();
                if(isDateTime){
                        hplCol6.DataTextFormatString = "{0:dd/MM/yyyy}";
                }
        #endregion
                break;
        default:
        #region "All Others"
                dgc = new BoundColumn();
                BoundColumn lcol = dgc as BoundColumn;
                if( r["QF_Name"].ToString().IndexOf(".")>0 )
                        lcol.DataField =
r["QF Name"].ToString().Substring(r["QF Name"].ToString().IndexOf(".")+1);
                else
                        lcol.DataField = r["QF Name"].ToString();

                lcol.HeaderText = r["QF_Title"].ToString();
                if (isDateTime) {
                        lcol.DataFormatString = "{0:dd/MM/yyyy}";
                }

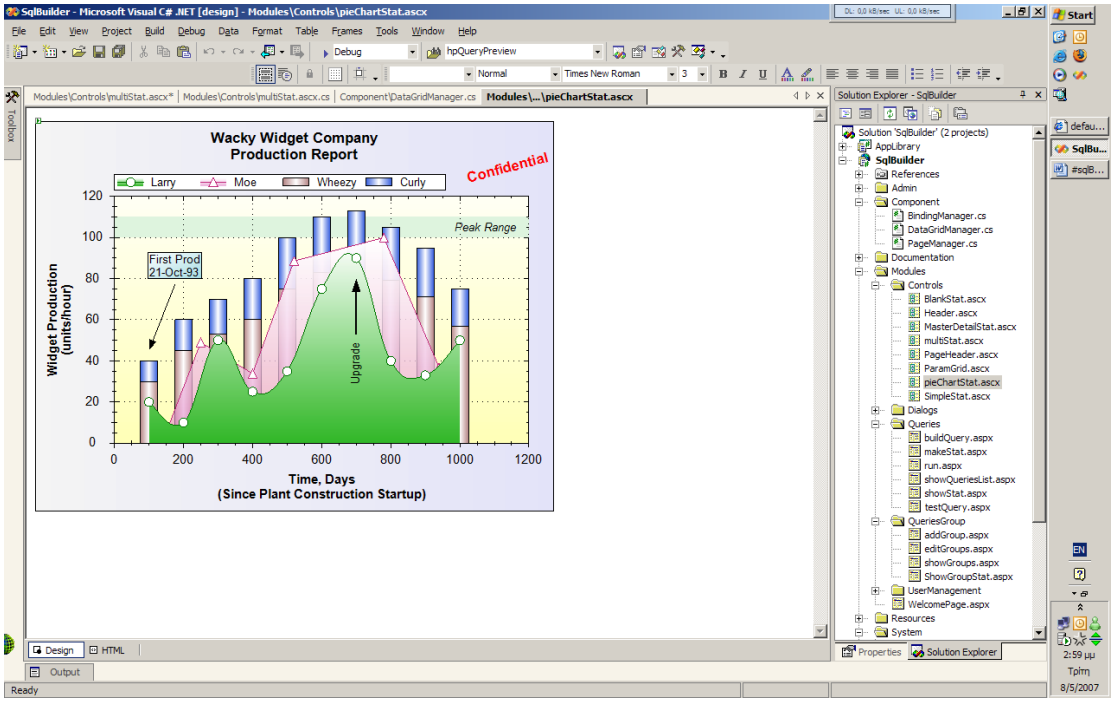
        #endregion
                break;
        }

        //now add to grid the column
        if(dgc!=null){
                dg.Columns.Add(dgc);
        }

        }
        dg.DataSource = gridsDataSource;
}
}
```

4.5.3 Γράφημα – PieChartStat User Control

Το User Control Γράφημα αποτελείται απο μία ανοικτή βιλιοθήκη που χρησιμοποιούμε με την ονομασία ZedGraph, όπως φαίνεται στην εικόνα 25 που ακολουθεί.



Εικόνα 27. Γράφημα – PieChartStat User Control

Η δημιουργία των γραφημάτων γίνεται προγραμματιστικά βάσει του είδους του γραφήματος που έχουμε επιλέξει. Σε αυτό το σημείο να θυμήσουμε οτι υπάρχουν τρία είδη γραφημάτων. Το γράφημα μπάρας, το Ιστόγραμμα και το γράφημα Πίτας. Ας δούμε τώρα πως δημιουργείται ένα γράφημα.

```
private void OnRenderGraph(System.Drawing.Graphics g, ZedGraph.MasterPane mPane){
        //get Data from session
        queryDataSet = (QueryDataSet)Session["queryDataSet"] ;
        //execute query
        ResultDataTable = DataManager.ExecuteMultiStatQuery( qID, queryDataSet,
paramDataTable );
        // Now begin with graph creation
        // get a reference to the GraphPane
        GraphPane myPane = mPane[0];
        // set some information, such as Graph Title, Horizontal Title, Vertical Title
        myPane.Title.Text =
queryDataSet.Query.Select("QueryID="+qID)[0]["Q Title"].ToString();
        //By default we set at XAxis the first element and at YAxis the second
        if( queryDataSet.QueryField.Select("QF_QueryID="+qID).Length >= 2 ) {
                myPane.XAxis.Title.Text =
queryDataSet.QueryField.Select("QF QueryID="+qID)[0]["QF Title"].ToString();
                myPane.YAxis.Title.Text =
queryDataSet.QueryField.Select("QF QueryID="+qID)[1]["QF Title"].ToString();
        }else{
                myPane.XAxis.Title.Text = "";
                myPane.YAxis.Title.Text = "";
        }

        //Now based on chartType that user have selected, create the appropriate chart
        int chartType =
Utils.GetSafeIntValue(queryDataSet.Query.Select("QueryID="+qID)[0]["Q_StatTypeID"]);
        switch(chartType){
                case 3:
                #region "CurveChart"
                //Create a PointPareList to add values
                        PointPairList ppl = new PointPairList();
                        foreach(DataRow dr in ResultDataTable.Rows) {
                                ppl.Add( Convert.ToDouble(dr[0]),
```

```
Convert.ToDouble(dr[1]),string.Format("({0},{1})",dr[0],dr[1]));
                        }
                        //Generate a red curve with diamond symbols
                        LineItem lnItem = myPane.AddCurve( string.Empty, ppl, Color.Red,
SymbolType.Diamond);
                #endregion
                        break;
                case 4:
                #region "BarChart"
                //Set values
                        string[] labels2 = new string[ResultDataTable.Rows.Count];
                        double[] x = new double[ResultDataTable.Rows.Count];
                        int i = 0 ;
                        foreach(DataRow dr in ResultDataTable.Rows) {
                                labels2[i] = dr[1].ToString();
                                x[i] = Convert.ToDouble(dr[0]);
                                i++;
                        }
                        //attach values to baritem
                        BarItem brItem = myPane.AddBar(string.Empty, x, x, Color.Red);
                        brItem.Bar.Fill = new Fill(Color.Blue, Color.White, Color.Blue);
                        brItem.ValueAxis(myPane);
                #endregion
                        break;
                case 5:
                #region "PieChart"
                //Set values
                        string[] pChartLabel = new string[ResultDataTable.Rows.Count];
                        double[] pChartValues = new double[ResultDataTable.Rows.Count];
                        int count = 0 ;
                        foreach(DataRow dr in ResultDataTable.Rows) {
                                pChartLabel[count] = dr[1].ToString();
                                pChartValues[count] = Convert.ToDouble(dr[0]);
                                count++;
                        }
                        //attach vales to pieChart
                        PieItem [] pItem = new PieItem[pChartValues.Length] ;
                        //Set default color red
                        Color curColor = Color.RosyBrown;
                        //add pie items
                        for( int iloop=0; iloop<pChartLabel.Length; iloop++ ){
                                pItem[iloop] = myPane.AddPieSlice( pChartValues[iloop],
curColor, 0.03, pChartLabel[iloop]+" ("+pChartValues[iloop].ToString()+")");
                        //set next color
                                curColor = ColorSymbolRotator.StaticNextColor;
                        }
                #endregion
                        break;
                }

                myPane.Legend.Position = LegendPos.Right;
                myPane.AxisChange(g);

}
```

5. Βελτιστοποίηση της Εφαρμογής

Αναμφισβήτητα μία εφαρμογή μπορεί να εξελιχθεί και να αναπτυχθεί περιταίρω βάσει πάντα των αναγκών και των απαιτήσεων που υπάρχουν, τόσο από τους χρήστες όσο και από την εξέλιξη της τεχνολογίας. Ορισμένες προτάσεις που μπορούν να βελτιστοποιήσουν την εφαρμογή είναι οι εξής:

- Υποστήριξη νέων τύπων βάσεων δεδομένων
- Υποστήριξη Web Services
- Διεργασία Αντιγράφων Ασφαλείας

5.1 Υποστήριξη νέων τύπων Βάσεων Δεδομένων

Η εφαρμογή αυτή αναπτύχθηκε ώστε να υποστηρίζει βάσεις δεδομένων τύπου Microsoft Access και Microsoft SQL Server. Οι βάσεις αυτές υποστηρίχθηκαν από την αρχή γιατί όπως έχουμε πει στην εισαγωγή του κειμένου αυτού, η εφαρμογή αυτή αναπτύχθηκε για να καλύψει τις ανάγκες μιας υπάρχουσας εφαρμογής η οποία χρησιμοποιεί για βάση δεδομένων τον Microsoft SQL Server. Οι τύποι βάσεων δεδομένων που θα μπορούσαν να υποστηριχθούν είναι η MySQL και η Oracle.

Για την υποστήριξη των νέων βάσεων δεδομένων αυτού που πρέπει να γίνει είναι ενσωματωθεί ο οδηγός προγράμματος(driver) της νέας βάσης δεδομένων και να υλοποιηθούν οι εντολές εκτέλεσης των ερωτημάτων.

5.2 Υποστήριξη Web Services

Για να μπορέσουμε να εκτελέσουμε ένα ερώτημα σε μια απομακρυσμένη βάση δεδομένων πρέπει αυτή η βάση να είναι εμφανής στο διαδίκτυο ώστε να μπορούμε να επικοινωνήσουμε μαζί της. Αυτό όπως καταλαβαίνετε περιέχει κινδύνους που αφορούν την ασφάλεια των δεδομένων της βάσης δεδομένων, καθώς είναι εκτεθειμένη. Για να αποφύγουμε τέτοιου είδους κινδύνους θα μπορούσαμε να χρησιμοποιήσουμε τα Web Services. Το Web Service ουσιαστικά θα είναι ένα επίπεδο μεταξύ της εφαρμογής και της βάσης δεδομένων το οποίο θα είναι υπευθυνο για την μεταφορά των δεδομένων. Με αυτό τον τρόπο η βάση δεδομένων δεν θα είναι εκτεθειμένη στο διαδίκτυο και θα παρέχεται μεγαλύτερη ασφάλεια.

5.3 Διεργασία Αντιγράφων Ασφαλείας

Ένα άλλο σημαντικό σημείο που θα βελτιώσει την εφαρμογή είναι η δημιουργία διεργασίας για την λήψη αντιγράφων ασφαλείας. Αυτό μπορεί να γίνει είτε δημιουργώντας μία προγραμματισμένη διεργασία στον Microsoft SQL Server, η οποία θα εκτελείται μια δεδομένη στιγμή, είτε μπορεί από το περιβάλλον εργασίας της εφαρμογής να δίδεται η εντολή λήψης αντιγράφων ασφαλείας.

6. Βιβλιογραφικές Αναφορές

ASP.NET - http://aspnet.4guysfromrolla.com

CodeProject - http://www.codeproject.com

Microsoft Press - "BUILDING WEB SOLUTIONS WITH ASP.NET AND ADO.NET"

Microsoft Library – http://msdn2.microsoft.com/en-us/library/default.aspx

O'REILLY - ADO.NET IN A NUTSHELL

Web.Config - http://www.sitepoint.com/article/web-config-file-demystified

Wikipedia – The Free Encyclopedia - http://en.wikipedia.org/

Πηγές από μηχανή αναζήτησης Google - www.google.com