ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. ΔΙΚΤΥΟΚΕΝΤΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Διπλωματική Εργασία

Σχεδιασμός και ανάπτυξη ετερογενούς κατανεμημένου συστήματος, προσανατολισμένου σε υπηρεσίες (SOA), για την ολοκλήρωση επιχειρησιακών διαδικασιών στο πλαίσιο επιθεωρήσεων υγιεινής και ασφάλειας τροφίμων.

Μεταπτυχιακός Φοιτητής: Παναγιώτης Σάτος, ΜΕ08102
Επιβλέπουσα Καθηγήτρια: Δρ. Βέρα-Αλεξάνδρα Σταυρουλάκη, Λέκτορας

Πειραιάς 2010

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, αισθάνομαι την ανάγκη να ευχαριστήσω τους ανθρώπους που με στήριξαν καθ' όλη τη διάρκεια της δωδεκάμηνης συγγραφής της. Ιδιαίτερες ευχαριστίες στην επιβλέπουσα καθηγήτρια κα. Βέρα Σταυρουλάκη για τη συνεχή βοήθεια και εμπιστοσύνη της στις δυνατότητές μου. Η συνεργασία μας υπήρξε άκρως εποικοδομητική. Θερμές ευχαριστίες στο διευθύνοντα σύμβουλο της εταιρείας Biocert κ. Δημήτρη Παναγή για την παροχή όλου του υλικού και του εξοπλισμού που χρειάστηκε το σύστημα για να υλοποιηθεί. Χωρίς τη βοήθειά του, τόσο κατά τη φάση της ανάλυσης όσο και κατά τη λειτουργία, δεν θα ήταν εφικτή η ολοκλήρωση του συστήματος και η εξαγωγή ασφαλών συμπερασμάτων. Πάνω από όλους όμως θέλω να ευχαριστήσω τη σύζυγό μου και συνάδελφο κα. Αθηνά Σίμου. Χωρίς την στήριξή της δεν θα ήταν δυνατή η παρακολούθηση του μεταπτυχιακού προγράμματος. Την ευχαριστώ και για την υπομονή της τα ατελείωτα βράδια που ξενυχτούσα μπροστά στην οθόνη. Τέλος, όχι όμως και τελευταία, ευχαριστώ τις οικογένειές μας, τους γονείς και τα αδέλφια μας για τη δική τους υποστήριξη.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία περιγράφει τη σχεδίαση και την ανάπτυξη ενός πληροφοριακού συστήματος, προσανατολισμένου σε υπηρεσίες, που ολοκληρώνει τις επιχειρησιακές διαδικασίες στο πλαίσιο υγιεινής και ασφάλειας τροφίμων. Μέσω του συστήματος μελετήθηκε η υπηρεσιοστρεφής αρχιτεκτονική ως μέσο ολοκλήρωσης ετερογενών κατανεμημένων συστημάτων. Συγκρίθηκε η διαχείριση δεδομένων μεταξύ των παραδοσιακών σχεσιακών βάσεων δεδομένων και των XML αρχείων. Αναλύονται οι λόγοι που κάνουν το σύστημα καινοτόμο, όχι μόνο σε ελληνικό αλλά και παγκόσμιο επίπεδο. Αναλύεται η αρχιτεκτονική του και περιγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν, οι οποίες στο σύνολό τους αποτελούν τεχνολογίες αιχμής. Γίνεται αναλυτική επεξήγηση του τρόπου ανάπτυξής του τόσο σε λογικό όσο και σε φυσικό επίπεδο. Τέλος, παρουσιάζονται οι λειτουργίες που παρέχει και γίνεται αναφορά στα συμπεράσματα απόδοσης, όπως αυτά προέκυψαν από την εφαρμογή του σε πραγματικές συνθήκες.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

1 ΕΙΣΑΓΩΓΗ

1.1 Ορισμός προβλήματος

Η επιθεώρηση υγιεινής και ασφάλειας τροφίμων (Hazard Analysis Critical Control Point, HACCP) δημιουργήθηκε για να καλύψει την ανάγκη ενιαίας αντιμετώπισης της υγιεινής και ασφάλειας τροφίμων. Βάσει της κείμενης ευρωπαϊκής νομοθεσίας (178/2004) κάθε επιχείρηση που ασχολείται με τον κλάδο των τροφίμων θα πρέπει να εφαρμόζει και να επιδεικνύει την αποτελεσματικότητα ενός τέτοιου συστήματος. Οι επιθεωρήσεις αποτελούνται από πολλές και διαφορετικές μεταξύ τους διαδικασίες, οι οποίες δύσκολα μπορούν να ολοκληρωθούν κάτω από ένα ενιαίο πλαίσιο.

Ενδεικτικά, οι αρχές του HACCP είναι:

- Εντοπίζονται οι τυχόν κίνδυνοι για την ασφάλεια των τροφίμων, οι οποίοι πρέπει να προληφθούν, να εξαλειφθούν ή να μειωθούν σε αποδεκτά επίπεδα, με σκοπό την παραγωγή ασφαλών τροφίμων.
- Εντοπίζονται τα κρίσιμα σημεία ελέγχου στο στάδιο ή στα στάδια, στα οποία ο έλεγχος είναι ουσιαστικής σημασίας για την πρόληψη ή την εξάλειψη ενός κινδύνου για την ασφάλεια των τροφίμων ή για την μείωσή του, ώστε να καταστεί δυνατή η επίτευξη του στόχου παραγωγής ασφαλών τροφίμων.
- Καθορίζονται κρίσιμα όρια στα κρίσιμα σημεία ελέγχου, με τα οποία χωρίζεται το αποδεκτό από το μη αποδεκτό, όσον αφορά την πρόληψη, την εξάλειψη ή τη μείωση των κινδύνων που έχουν εντοπιστεί.
- Καθορίζονται και εφαρμόζονται αποτελεσματικές διαδικασίες παρακολούθησης στα κρίσιμα σημεία ελέγχου.
- Καθορίζονται τα διορθωτικά μέτρα, όταν η παρακολούθηση υποδεικνύει ότι ένα κρίσιμο σημείο ελέγχου βρίσκεται εκτός ελέγχου.
- Καθορίζονται οι διαδικασίες επαλήθευσης, για την αποτελεσματική λειτουργία των μέτρων.
- Τηρούνται αρχεία ώστε να εξασφαλίζεται η ουσιαστική εφαρμογή των μέτρων και να καθίστανται δυνατοί οι επίσημοι έλεγχοι. [28]

Η ανάπτυξη ενός τέτοιου συστήματος αποτελείται από τα παρακάτω στάδια:

- Επιλογή της ομάδας haccp
- Περιγραφή του προϊόντος (τροφίμου)
- Προσδιορισμός της σχεδιαζόμενης χρήσης του προϊόντος
- Κατασκευή του διαγράμματος ροής της παραγωγικής διαδικασίας
- Επαλήθευση του διαγράμματος ροής
- Καταγραφή των κινδύνων σε όλα τα στάδια της παραγωγής και των αντίστοιχων προληπτικών μέτρων
- Καθορισμός των κρίσιμων σημείων παραγωγής (ccps)
- Καθορισμός των κρίσιμων ορίων για τα ccps
- Εγκατάσταση συστήματος παρακολούθησης των ccps και των κρίσιμων ορίων τους
- Καθορισμός των διορθωτικών ενεργειών για τις αποκλίσεις από τα κρίσιμα όρια
- Εγκατάσταση συστήματος αρχειοθέτησης και καταγραφής του σχεδίου haccp
- Προσδιορισμός των διαδικασιών επαλήθευσης του συστήματος haccp

Η εφαρμογή ενός συστήματος haccp σχετίζεται με τις εξής έννοιες:

1. Ποιότητα
2. Υγιεινή
3. Ορθή Βιομηχανική Πρακτική

Ποιότητα

Ποιότητα γενικά είναι η ικανότητα ενός προϊόντος να ανταποκρίνεται στο σκοπό για τον οποίο προορίζεται. Ειδικότερα για τα τρόφιμα είναι ο βαθμός προσαρμογής στις απαιτήσεις του καταναλωτή που σχετίζονται με τα χαρακτηριστικά του όπως:

- Οργανοληπτικά χαρακτηριστικά (χρώμα, μέγεθος, σχήμα, υφή)
- Θρεπτική αξία
- Συμφωνία με την νομοθεσία
- Συσκευασία
- Ασφάλεια
- Διαθεσιμότητα

Υγιεινή

Εννοούμε την διατήρηση καλών συνθηκών υγιεινής στην βιομηχανική μονάδα.

- Υγιεινή του περιβάλλοντος εργασίας
- Υγιεινή πρώτων υλών και συστατικών
- Συνθήκες υγιεινής κατά την παραγωγική διαδικασία, την αποθήκευση, και την μεταφορά των προϊόντων.

Ορθή Βιομηχανική Πρακτική

Εννοούμε τους κανόνες-απαιτήσεις υγιεινής για την βιομηχανία τροφίμων που αφορούν:

- Το προσωπικό της βιομηχανίας
- Την τοποθεσία και τον σχεδιασμό της βιομηχανικής εγκατάστασης
- Τις συσκευές και τα μηχανήματα παραγωγής
- Γενική υγιεινή, καθαρισμός και απολυμάνσεις
- Την επιλογή των πρώτων υλών
- Τις διεργασίες παραγωγής
- Τα υλικά συσκευασίας και την προσθήκη ετικετών
- Συστήματα ελέγχου ποιότητας

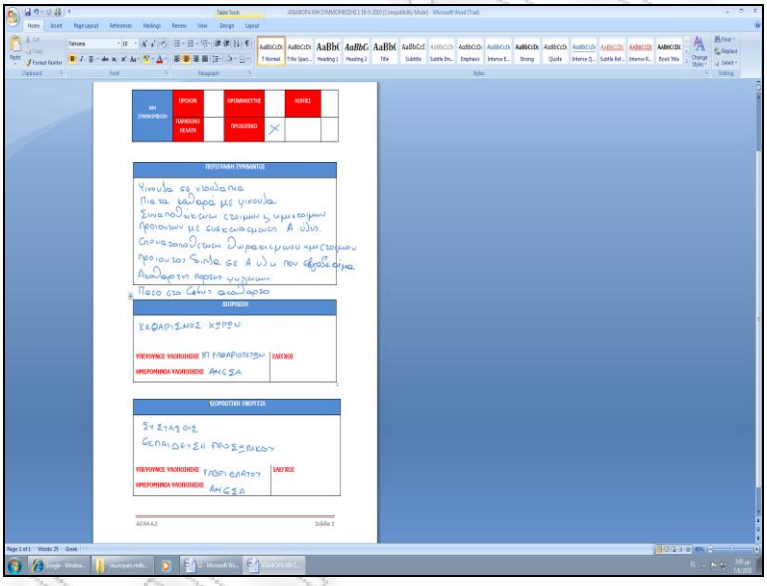
Με την ολοκλήρωση των παραπάνω βημάτων ξεκινά η εφαρμογή του συστήματος δηλαδή η εφαρμογή της κείμενης νομοθεσίας και η προσαρμογή των απαιτήσεων αυτής πάνω στην λειτουργία της εταιρείας, η οποία περιλαμβάνει την τεκμηρίωση της εφαρμογής. Δηλαδή, την καταγραφή όλων των ενεργειών που προβλέπονται σε ένα σύστημα ασφάλειας τροφίμων και βέβαια την επαλήθευση της αποτελεσματικότητας ενός τέτοιου συστήματος. Η επαλήθευση αναφέρεται στην δυνατότητα της εταιρείας να αποδεικνύει εμπράκτως την δυνατότητα να παράγει ασφαλές προϊόν. Για την επαλήθευση της αποτελεσματικότητας απαιτείται πέραν των άλλων η διενέργεια εσωτερικών επιθεωρήσεων με σκοπό την τεκμηρίωση της εφαρμογής αυτού. Οι επιθεωρήσεις αυτές (audits) διακρίνονται σε :

1ου μέρους: πραγματοποιείται από την ίδια την επιχείρηση και η οποία προγραμματίζεται από την αρχή του έτους και σκοπός είναι να καλύπτει σε διάστημα ενός έτους τον πλήρη έλεγχο του συνόλου της δραστηριότητας της εταιρείας.

2ου μέρους: πραγματοποιείται από εξωτερικό συνεργάτη μετά από ανάθεση έργου σε αυτόν από την εταιρεία και μπορεί να καλύπτει κομμάτι/ια ή και το σύνολο της δραστηριότητας του πελάτη (αντικείμενο παρούσας εργασίας)

3ου μέρους: πραγματοποιείται είτε από φορείς πιστοποίησης με σκοπό την λήψη αντίστοιχου πιστοποιητικού - βεβαίωσης είτε από αντίστοιχο αρμόδιο κρατικό φορέα (ΕΦΕΤ ή κτηνιατρική υπηρεσία).

Σκοπός των επιθεωρήσεων αυτών είναι να εκτιμηθεί ο βαθμός εφαρμογής του συστήματος, η αποτελεσματικότητα αυτού, καθώς και η προσαρμογή του στην κείμενη νομοθεσία. Τα αποτελέσματα των επιθεωρήσεων αυτών καταγράφονται σε ειδικά έντυπα (αναφορές μη συμμόρφωσης), όπως το παρακάτω, στα οποία και εμφανίζονται (Εικόνα 1.1).



Εικόνα 1.1 Έντυπο ερωτηματολόγιο επιθεώρησης.

Μη συμμόρφωση χαρακτηρίζεται κάθε πιθανή παρέκκλιση από τα επιθυμητά επίπεδα (standards) της εταιρείας και της νομοθεσίας και μπορεί να αναφέρεται στο σύνολο της δραστηριότητας της εταιρείας

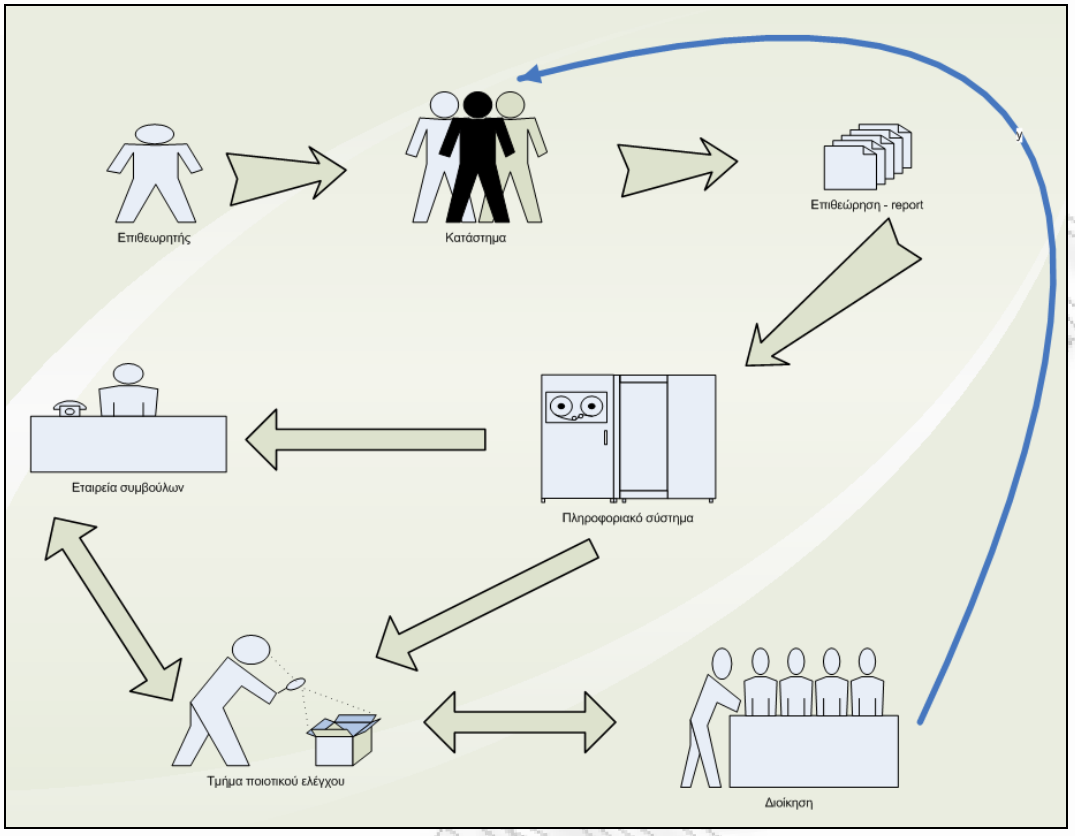
- Το γεγονός
- Τα αίτια
- Η διορθωτική ενέργεια (άμεση ενέργεια με σκοπό την άμεση επιδιόρθωση του προβλήματος)
- Η διόρθωση (ενέργεια/ες που πρέπει να γίνει ώστε να μην ξαναεμφανιστεί το πρόβλημα)

Μετά την τεκμηρίωση των Μη Συμμορφώσεων ακολουθεί η διαδικασία κλεισίματος αυτών και μετά την πάροδο καθορισμένου χρονικού διαστήματος ακολουθεί η στατιστική ανάλυση αυτών και η εξαγωγή συμπερασμάτων για την σωστή ή όχι

λειτουργία του συστήματος. Μια εσωτερική επιθεώρηση οποιουδήποτε τύπου πραγματοποιείται με τη βοήθεια ερωτηματολογίων (τυποποιημένων ανάλογα με τον φορέα επιθεώρησης) στα οποία και καταγράφονται όλα τα προαναφερόμενα συμβάντα. Τα ερωτηματολόγια αυτά διαφέρουν από φορέα σε φορέα. Θα πρέπει όμως να περιλαμβάνουν τις ελάχιστες απαιτήσεις της νομοθεσίας. Όλα τα ερωτηματολόγια που χρησιμοποιούνται σήμερα στην αγορά είναι τυποποιημένα και σε έντυπη μορφή. Αυτό σημαίνει ότι ο επιθεωρητής θα πρέπει να σημειώνει σε αυτά όλα τα συμβάντα κατά την διάρκεια της επιθεώρησης και στο τέλος αυτής να ενημερώνει τον εκπρόσωπο της διοίκησης για αυτά. Ο εκπρόσωπος της διοίκησης στην συνέχεια θα πρέπει να ενημερώσει την διοίκηση και να ληφθούν τα απαραίτητα διορθωτικά μέτρα. Η χρήση των εντύπων έχει ως αποτέλεσμα την χρονική καθυστέρηση στο στάδιο ενημέρωσης της διοίκησης ιδίως σε περιπτώσεις εταιρειών όπου η έδρα της εταιρείας είναι μεγάλη απόσταση από την έδρα της παραγωγικής μονάδας ή ακόμα περισσότερο όταν μια εταιρεία διαθέτει πάρα πολλά υποκαταστήματα (π.χ. Super Market). Κάθε τέτοια καθυστέρηση έχει ως συνέπεια την αύξηση του κινδύνου προστίμου από αρμόδιο κρατικό φορέα που υπολογίζεται σε κάποιες χιλιάδες ευρώ στην καλύτερη των περιπτώσεων. Σκοπός λοιπόν είναι η δημιουργία ενός συστήματος ειδοποιήσεων (real time) μέσα από το οποίο η εταιρεία-πελάτης θα μπορεί σε πραγματικό χρόνο να δει τα αποτελέσματα των επιθεωρήσεων και να κινήσει τις όποιες διαδικασίες ελέγχου των μη συμμορφώσεων.
Από τα παραπάνω μπορούμε εύκολα να συμπεράνουμε ότι υπάρχουν πολλοί εμπλεκόμενοι σε μία επιθεώρηση HACCP. Το μεγαλύτερο πρόβλημα βέβαια εστιάζεται στο γεγονός ότι οι επιμέρους εμπλεκόμενοι δεν αποτελούν μέλη τμήματος οργανισμού ή εταιρείας. Στις περισσότερες περιπτώσεις προέρχονται από διαφορετικούς οργανισμούς ή εταιρείες (η πλειοψηφία των επιθεωρητών είναι εξωτερικοί συνεργάτες και όχι υπάλληλοι της εταιρείας που αναλαμβάνει την επιθεώρηση). Αυτό συνεπάγεται μεγάλους χρόνους για την ολοκλήρωση μιας επιθεώρησης και υψηλό βαθμό γραφειοκρατικών διαδικασιών, με ό,τι συνεπάγεται αυτό.

1.2 Σκοπός εργασίας

Σκοπός της παρούσας εργασίας ήταν η μελέτη της ολοκλήρωσης συστημάτων προσανατολισμένα σε υπηρεσίες σε ετερογενή κατανεμημένα περιβάλλοντα. Η σχεδίαση και η ανάπτυξη ενός ολοκληρωμένου συστήματος με αυτές τις προδιαγραφές, έγινε με γνώμονα το πρόβλημα της ενότητας 1.1 ώστε να ολοκληρώσει τις επιχειρησιακές διαδικασίες που απαιτούνται στο πλαίσιο των επιθεωρήσεων υγιεινής και ασφάλειας τροφίμων (επιθεώρηση HACCP). Επίσης στόχευε στην οργάνωση των εμπλεκόμενων μελών, καταφέρνοντας την παράλληλη επεξεργασία των πληροφοριών στο κάθε στάδιο της, κατά τα άλλα συριακής, επιθεώρησης, διαδικασίας επιθεώρησης. Το σύστημα αποτέλεσε την «καρδιά» της όλης διαδικασίας (Εικόνα 1.2) σε αρκετά υψηλό επίπεδο αφαίρεσης.



Εικόνα 1.2 Στάδια επιθεώρησης και εμπλεκόμενα μέλη.

Για την επίτευξη του αρχικού στόχου, το σύστημα αναπτύχθηκε σε Υπηρεσιοστρεφή Αρχιτεκτονική (Service-Oriented Architecture, SOA) ολοκληρώνοντας τις διαδικασίες μέσω Web Services.

1.3 Δομή εργασίας

Η παρούσα εργασία αποτελείται από εννιά (9) κεφάλαια.

- Το 1ο κεφάλαιο αποτελεί την εισαγωγή, στην οποία αρχικά ορίζεται το πρόβλημα με το οποίο ασχολείται η εργασία αυτή. Αναλύεται ο σκοπός της, η δομή της και τέλος αναφέρονται παρόμοια συστήματα της ελληνικής αλλά και της παγκόσμιας αγοράς.
- Στο 2ο κεφάλαιο αναλύονται οι απαιτήσεις ενός τέτοιου συστήματος καθώς και οι περιορισμοί για την υλοποίηση του.
- Στο 3ο κεφάλαιο αναφέρονται οι τεχνολογίες και οι πλατφόρμες, οι οποίες χρησιμοποιήθηκαν κατά τη διάρκεια ανάπτυξης του συστήματος.
- Στο 4ο κεφάλαιο αναλύεται η λογική σχεδίαση του συστήματος.
- Στο 5ο η φυσική του σχεδίαση.
- Στο 6ο αναλύονται τα επιμέρους project που αποτελούν το σύστημα, με έμφαση στα σημαντικά σημεία του κώδικα.
- Στο 7ο περιγράφεται αναλυτικά η λειτουργία του συστήματος με προβολή των επιμέρους οθονών του.

- Στο 8^{ο} κεφάλαιο αναγράφεται η ανατροφοδότηση (feedback) του συστήματος, όπως προέκυψε από την εφαρμογή του για τέσσερις (4) εβδομάδες, σε πραγματικές συνθήκες στην εταιρεία Biocert στη Λαμία.
- Τέλος, στο 9^{ο} κεφάλαιο αναγράφονται τα συμπεράσματα που προέκυψαν, καθώς και προτάσεις για μελλοντικές επεκτάσεις και προσθήκες του συστήματος.

1.4 Βιβλιογραφική επισκόπηση

Τυποποιημένα συστήματα επιθεωρήσεων δεν υπάρχουν στην ελληνική αγορά αλλά και στις αγορές του εξωτερικού. Αυτό συμβαίνει γιατί κάθε εμπλεκόμενος σε επιθεώρηση επιχείρησης τροφίμων σχεδιάζει τα δικά του ερωτηματολόγια τα οποία και χρησιμοποιεί κατά την εργασία του. Ολοκληρωμένο σύστημα αυτόματης ειδοποίησης πελάτη δεν υπάρχει πουθενά στην Ελλάδα. Όλη η ροή εργασίας (workflow) μιας επιθεώρησης γίνεται με ερωτηματολόγια και αναφορές επιθεώρησης σε έντυπη μορφή. Η αποστολή τους γίνεται μέσω e-mail και σε μερικές περιπτώσεις ταχυδρομικά!
Ένα τέτοιο παρόμοιο σύστημα παρουσιάστηκε για πρώτη φορά στους Πανασιατικούς αγώνες στην DOHA στο QATAR όπου ο φορέας πιστοποίησης και επιθεωρήσεων TUV REINALD σχεδίασε για τους αγώνες αυτούς ένα σύστημα με το οποίο οι επιθεωρητές πραγματοποιούσαν την επιθεώρηση τους με την βοήθεια ερωτηματολογίου σε smart phones και στην συνέχεια πραγματοποιούσαν αποστολή των αποτελεσμάτων στα κεντρικά του φορέα ο οποίος στην συνέχεια θα ειδοποιούσε τον πελάτη γραπτώς για τυχόν αποκλίσεις – μη συμμορφώσεις. Οι επιθεωρητές μετά το τέλος της επιθεώρησης συνέτασσαν ειδικό έντυπο (report) το οποίο, αφού το υπέγραφαν, το έστελναν στα κεντρικά της εταιρείας.

Με βάση αυτή την ιδέα, στα πλαίσια της παρούσας εργασίας, αναπτύχθηκε ένα παρόμοιο σύστημα. Συγκεκριμένα η ανάγκη δημιουργίας του ήταν η ακόλουθη:
Όλες οι εταιρείες τροφίμων υποχρεούνται από τον νόμο να τηρούν σύστημα υγιεινής και ασφάλειας τροφίμων. Όλες αυτές οι επιχειρήσεις πραγματοποιούν εσωτερικές επιθεωρήσεις σε ποσοστό 99% με την βοήθεια εξωτερικών συμβούλων. Οι εταιρείες συμβούλων από την μεριά τους χρησιμοποιούν πλήθος επιθεωρητών με σκοπό την κάλυψη του όγκου των επιθεωρήσεων. Η ανάγκη λοιπόν για άμεση ενημέρωση τόσο του πελάτη όσο και της εταιρείας συμβούλων για τα αποτελέσματα των επιθεωρήσεων ήταν επιτακτική και χρειαζόταν να γίνεται σε πραγματικό χρόνο δηλαδή την ίδια ημερομηνία της επιθεώρησης.

2 ΑΠΑΙΤΗΣΕΙΣ

Σε μία επιθεώρηση εμπλέκονται οι παρακάτω ρόλοι:

1. Εταιρεία συμβούλων (Consulting Services, CS)
2. Επιθεωρητής (Lead Auditor, LA)
3. Υποκατάστημα ή Προμηθευτής υπό επιθεώρηση
4. Πελάτης
    i. Τμήμα Ποιοτικού Ελέγχου (Quality Control, QC)
    ii. Διοίκηση

Οι εταιρείες τροφίμων που εφαρμόζουν σύστημα υγιεινής και ασφάλειας τροφίμων (HACCP) αναθέτουν σε εξωτερικό συνεργάτη (εταιρεία συμβούλων) την επιθεώρηση των συστημάτων τους, στα καταστήματα και στους προμηθευτές τους (Second Part Audit - Επιθεώρηση Δευτέρου Βαθμού).

Οι επιχειρησιακές διαδικασίες που ολοκληρώνονται από το σύστημα είναι οι ακόλουθες:

1. Προετοιμασία ερωτηματολογίου για την επιθεώρηση.
2. Ενημέρωση επιθεωρητή και αποστολή του ερωτηματολογίου.
3. Διεξαγωγή επιθεώρησης από τον επιθεωρητή.
4. Αποστολή αναφοράς (αποτελέσματα επιθεώρησης) της επιθεώρησης στην εταιρεία συμβούλων.
5. Εξαγωγή αναφορών (αποτελέσματα για τον πελάτη) από την εταιρεία συμβούλων.
6. Ενημέρωση του πελάτη για τα αποτελέσματα της επιθεώρησης.

Από τα παραπάνω συμπεραίνεται ότι το σύστημα πρέπει να αποτελείται από τρία (3) επίπεδα. Κι αυτό γιατί σε υψηλό επίπεδο αφαίρεσης θα λέγαμε ότι οι βασικές οντότητες είναι η εταιρεία συμβούλων, οι επιθεωρητές και οι πελάτες. Και οι τρεις είναι γεωγραφικά απομακρυσμένες και εμπλέκονται σε διαφορετικά στάδια της επιθεώρησης. Αναπτύσσοντας λοιπόν το σύστημα σε τρία επίπεδα, είναι εφικτό το κάθε ένα να λειτουργεί παράλληλα. Από τις παραπάνω επιχειρησιακές διαδικασίες, η $1^{η}$, η $2^{η}$ και η $5^{η}$ είναι διαδικασίες της εταιρείας συμβούλων, η $3^{η}$ και η $4^{η}$ των επιθεωρητών και η $6^{η}$ των πελατών. Τα επίπεδα στα οποία χωρίστηκε το σύστημα είναι τα ακόλουθα:

- Στο πρώτο, υπάρχουν τα δεδομένα του συστήματος. Τα δεδομένα αποθηκεύονται σε βάση δεδομένων και XML αρχεία, ενώ η διαχείρισή τους γίνεται από την εταιρεία συμβούλων.
- Στο δεύτερο, υπάρχουν οι εφαρμογές των επιθεωρητών (Clients).
- Στο τρίτο, υπάρχει η διεπαφή χρήστη (web εφαρμογή) για την προβολή των αποτελεσμάτων στους πελάτες.

Στο πρώτο, εμπλέκονται ο ρόλος του επιθεωρητή και το υπό επιθεώρηση υποκατάστημα ή προμηθευτής. Στο δεύτερο, η εταιρεία συμβούλων και στο τρίτο, το Τμήμα Ποιοτικού Ελέγχου (QC) του πελάτη.
Για το πρώτο επίπεδο, η χρήση XML αρχείων ως μέσο αποθήκευσης κρίνεται απαραίτητη, λόγω του γεγονότος ότι οι επιθεωρητές κατά τη διάρκεια μιας επιθεώρησης είναι πολύ πιθανό να μην έχουν διαδικτυακή πρόσβαση. Αυτό θα είχε ως αποτέλεσμα την υποχρεωτική αποθήκευση των ερωτηματολογίων και των

αναφορών, είτε σε τοπική βάση δεδομένων είτε σε τοπικά αρχεία. Η πρώτη περίπτωση απορρίφθηκε για τους προφανείς λόγους. Αυτοί είναι το αυξημένο κόστος και η ενημέρωση - συντήρηση πολλών βάσεων δεδομένων. Για παράδειγμα, στην περίπτωση αλλαγής της δομής ενός πίνακα, θα έπρεπε να ενημερωθούν όλες οι βάσεις δεδομένων για όλους τους επιθεωρητές. Για τους παραπάνω λόγους επιλέχθηκε η δεύτερη περίπτωση, για την οποία χρησιμοποιήθηκαν XML αρχεία, μιας και μέσω αυτών είναι εφικτός ο ορισμός της δομής των δεδομένων και ακόμη η αποτύπωση του schema, όπως στις βάσεις δεδομένων.
Όσον αφορά τη διαδικασία της επιθεώρησης, αρχικά η εταιρεία συμβούλων θα δημιουργεί τα ερωτηματολόγια που θα χρησιμοποιούνται στις επιθεωρήσεις των υποκαταστημάτων και των προμηθευτών. Τα ερωτηματολόγια θα ομαδοποιούνται σε κατηγορίες, ανάλογα με το είδος της επιθεώρησης. Το κάθε ερωτηματολόγιο θα χωρίζεται σε τομείς και ο κάθε τομέας θα περιλαμβάνει μία ή περισσότερες ερωτήσεις. Υπάρχουν περιπτώσεις υπέρ-τομέων, δηλαδή τομείς που περιέχουν τομείς και όχι ερωτήσεις. Παράδειγμα ενός ερωτηματολογίου απεικονίζεται στη συνέχεια.

ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ			
ΕΓΚΑΤΑΣΤΑΣΕΙΣ			
Εξωτερικό περιβάλλον			
Σαφής οριοθέτηση εγκατάστασης	nai		
Όχι γειτνίαση με πηγές μόλυνσης	nai		
Ικανοποιητική διαμόρφωση, καθαριότητα, υγιεινή περιβάλλοντος χώρου	oxi		
Σχεδιασμός – Διαχωρισμός – Επάρκεια χώρων			
Ο σχεδιασμός των χώρων και η ροή της παραγωγής:	apotelesma	paratiriseis	
Αποτρέπει κίνδυνο διασταυρούμενης επιμόλυνσης	n/a		
Εξασφαλίζει ανεξάρτητη είσοδο- έξοδο προσωπικού, Α' υλών, βοηθητικών υλών, υλικών συσκευασίας & τελικών προϊόντων	n/a		
Εξασφαλίζει μέγεθος επαρκές για τον συνήθη όγκο παραγωγής	nai		
Υπάρχοντες διακριτοί χώροι:			
Παραλαβής των Α' υλών, βοηθητικών υλών και υλικών συσκευασίας.	n/a		
Αποθήκευσης Α' υλών (υπό ψύξη, κατάψυξη) σε κατάλληλες συνθήκες.	oxi		
Αποθήκευσης βοηθητικών υλών	nai		
Αποθήκευσης υλικών συσκευασίας	nai		
Τεμαχισμού κρέατος	nai		
Παραγωγής κιμά- παρασκευασμάτων	oxi		
Α΄και Β΄ συσκευασίας	oxi		
Αποθήκευσης τελικών προϊόντων.	n/a		
Αποστολής έτοιμων προιόντων	nai		
Καθαρισμού -απολύμανσης του εξοπλισμού	n/a		
Εργαστηριακή υποστήριξη εντός ή εκτός της επιχείρησης	nai		
Αποδυτήρια -Χώροι υγιεινής προσωπικού	nai		
Καντίνα/Εστιατόριο προσωπικού	nai		
Γενικά χαρακτηριστικά παραγωγικών & βοηθητικών χώρων			
Δάπεδα. Από κατάλληλα υλικά κατασκευής,σε καλή κατάσταση συντήρησης και με δυνατότητα καθαρισμού, αποστράγγισης.	nai		
Τοίχοι -Διαχωριστικά. Από υλικά κατάλληλα: χρήση στεγανών, μη απορροφητικών και μη τοξικών υλικών που διευκολύνουν τον επαρκή καθαρισμό/απολύμανση.	nai		
Οροφές- Ψευδοροφές. Σχεδιασμός, κατασκευή, κατάσταση συντήρησης που να αποτρέπουν συσσώρευση ρύπων και τη συμπύκνωση υδρατμών.	oxi		
Παράθυρα / ανοίγματα: Σε σωστή θεση, εφοδιασμένα με πλέγματα προστασίας	oxi		
Ψυγεία: Κατάλληλα υλικά επένδυσης, δάπεδο με δυνατότητα καθαρισμού - αποστράγγισης	n/a		
Πόρτες. Από λείο μη απορροφητικό υλικό που διευκολύνει τον επαρκή καθαρισμό..	nai		
Φωτισμός:Επαρκής φυσικός ή τεχνητός φωτισμός, φωτιστικά μέσα προστατευόμενα.	n/a		

Εικόνα 2.1 Υπόδειγμα έντυπου ερωτηματολογίου επιθεώρησης κρέατος.

Μια επιθεώρηση περιλαμβάνει ενέργειες σε τρία επίπεδα, όπως αναφέρθηκε και παραπάνω. Πριν την επιθεώρηση απαιτείται η προετοιμασία του ερωτηματολογίου, ανάλογα με το είδος της επιθεώρησης (επιθεώρηση κρέατος, ψαριού κλπ.) και η ανάθεσή του σε επιθεωρητή. Ο επιθεωρητής αφού κανονίσει την ημερομηνία και την ώρα που θα γίνει η επιθεώρηση στο υποκατάστημα ή στον προμηθευτή του πελάτη, προβαίνει στη διαδικασία της επιθεώρησης. Για κάθε ερώτηση του ερωτηματολογίου, επιλέγει μία εκ των απαντήσεων: ΝΑΙ, ΟΧΙ και Δ/Α (Δεν Απαιτείται). Αφού απαντηθούν όλες οι ερωτήσεις, για τον κάθε τομέα ορίζεται η σημαία. Αν ένας τομέας είναι προβληματικός χαρακτηρίζεται με κόκκινη σημαία, αν είναι ελλιπής με μπλε κι αν πληροί τις προϋποθέσεις με πράσινη. Με το πέρας της επιθεώρησης, στέλνει τα αποτελέσματα στην εταιρεία συμβούλων. Η εταιρεία συμβούλων, ενημερώνει το Τμήμα Ποιοτικού Ελέγχου (QC) του πελάτη για τα αποτελέσματα της επιθεώρησης. Εφ' όσον ζητηθεί, η εταιρεία είναι υπεύθυνη να τροφοδοτήσει το QC με αναφορές (Reports) των επιθεωρήσεων. Ο πελάτης (Διοίκηση) ενημερώνεται από το QC και εφ΄ όσον ζητηθεί και από την εταιρεία συμβούλων.

Ως παραδοχές – περιορισμούς του συστήματος μπορούμε να ορίσουμε τις ακόλουθες:

- Η εταιρεία συμβούλων διαθέτει πολλούς επιθεωρητές (Auditors).
- Κάθε επιθεωρητής (Auditors) μπορεί να επιθεωρεί ένα ή περισσότερα υποκαταστήματα (Branches) ή προμηθευτές (Suppliers).
- Σε κάθε επιθεώρηση (Audits) μπορούν να παραστούν ένας ή περισσότεροι επιθεωρητές (Auditors).
- Ένας επιθεωρητής (Auditors) δεν μπορεί την ίδια στιγμή να πραγματοποιεί περισσότερες από μία επιθεωρήσεις (Audits).
- Ο αριθμός των επιθεωρήσεων (Audits) ημερησίως, δεν μπορεί να υπερβαίνει τις τρεις (3) για κάθε επιθεωρητή (Auditors).
- Ο επιθεωρητής (Auditors) δεν θα έχει πρόσβαση στο σύστημα, πλην της εφαρμογής που θα χρησιμοποιεί για την επιθεώρηση (Audits).
- Κάθε πελάτης (Customers) έχει ένα ή περισσότερα υποκαταστήματα (Branches).
- Κάθε πελάτης (Customers), υποκατάστημα (Branches) ή προμηθευτής (Suppliers) έχει μία ή περισσότερες επαφές (Contacts).
- Ένα υποκατάστημα (Branches) μπορεί να εμπλέκεται σε καμία ή πολλές επιθεωρήσεις (Audits).
- Ένα υποκατάστημα (Branches) προμηθεύεται από έναν ή περισσότερους προμηθευτές (Suppliers).
- Ένας προμηθευτής (Suppliers) μπορεί να εμπλέκεται σε καμία ή πολλές επιθεωρήσεις (Audits).
- Ένας προμηθευτής (Suppliers) προμηθεύει ένα ή περισσότερα υποκαταστήματα (Branches).
- Κανένας ρόλος του συστήματος δεν θα μπορεί να τροποποιεί τα δεδομένα, μετά το πέρας της επιθεώρησης (Audits).
- Μία επιθεώρηση έχει ως αποτέλεσμα πράσινη σημαία, όταν όλοι οι τομείς της επιθεώρησης έχουν πράσινη σημαία.
- Μία επιθεώρηση έχει ως αποτέλεσμα μπλε σημαία, όταν τουλάχιστον ένας τομέας έχει μπλε σημαία.
- Μία επιθεώρηση έχει ως αποτέλεσμα κόκκινη σημαία (προβληματική), όταν τουλάχιστον ένας τομέας έχει κόκκινη σημαία.

Όπως αναλύθηκε παραπάνω, για να αποφευχθεί η χρήση βάσης δεδομένων σε κάθε client, τα ερωτηματολόγια καθώς και τα αποτελέσματα της επιθεώρησης θα αποθηκεύονται σε XML αρχεία, τα οποία θα λαμβάνονται και θα αποστέλλονται στο server μέσω web services. Για κάθε επιθεώρηση ο επιθεωρητής θα μπορεί, μέσω της εφαρμογής, να κάνει λήψη του αντίστοιχου ερωτηματολογίου. Για την ευκολία της επιθεώρησης, θα χρησιμοποιείται Tablet PC. Επομένως η εφαρμογή του επιθεωρητή θα πληροί της προδιαγραφές μιας touch screen εφαρμογής. Όλα τα ερωτηματολόγια και τα αποτελέσματα των επιθεωρήσεων θα αποθηκεύονται σε βάση δεδομένων, στο server της εταιρείας συμβούλων. Τα ερωτηματολόγια και οι αναφορές που θα υπάρχουν στους clients (επιθεωρητές), καθώς και τα αντίστοιχα που θα είναι αποθηκευμένα στη βάση δεδομένων, θα είναι itps αρχεία. Δηλαδή τα XML αρχεία θα είναι κρυπτογραφημένα με δικό μας αλγόριθμο κρυπτογράφησης, παρέχοντας ασφάλεια των περιεχομένων τους τόσο στην μεταφορά, όσο και στην αποθήκευσή τους τοπικά στον κάθε client. Με αυτό τον τρόπο διασφαλίζεται η περίπτωση υποκλοπής της δομής των ερωτηματολογίων (τομείς, ερωτήσεις), που έχει συντάξει η εταιρεία συμβούλων. Για παράδειγμα, τα ερωτηματολόγια που θα αποστέλλονται στους επιθεωρητές (εξωτερικούς συνεργάτες) θα αποθηκεύονται τοπικά σε κρυφό (hidden) φάκελο, ώστε να μην είναι προσπελάσιμα. Στην περίπτωση που τα εντοπίσει ο επιθεωρητής, δεν θα είναι δυνατή η αντιγραφή τους ή χρησιμοποίησή τους από τον επιθεωρητή για επιθεωρήσεις εκτός της εταιρείας συμβούλων. Αντίστοιχα διασφαλίζεται και η ασφάλεια των αποτελεσμάτων της επιθεώρησης, σε περίπτωση υποκλοπής τους κατά τη μεταφορά. Πρόσβαση στα δεδομένα της βάσης θα έχει μόνον ο ορισμένος, από την εταιρεία συμβούλων, υπεύθυνος υλοποίησης της συγκεκριμένης υπηρεσίας, μέσω μιας DB Application (Back-end). Μέσω αυτής της εφαρμογής θα μπορούν να εξαχθούν αναφορές (SAP Crystal Reports) και να αποσταλούν στο QC του πελάτη. Οι αναφορές εξάγονται αυτόματα, αφού ο διαχειριστής παραλάβει τα αποτελέσματα για μια επιθεώρηση. Επίσης, μέρος των αποτελεσμάτων της επιθεώρησης θα προβάλλεται στον πελάτη ακριβώς μετά την αποστολή της αναφοράς από τον επιθεωρητή. Με χρήση των Google Maps θα υπάρχει η δυνατότητα της μαζικής προβολής όλων των υποκαταστημάτων - προμηθευτών που επιθεωρήθηκαν. Στα αποτελέσματα της επιθεώρησης, ο υπεύθυνος του QC, έχει άμεση πρόσβαση, ανεξαρτήτως γεωγραφικής θέσης, μιας και τα αποτελέσματα προβάλλονται σε διαδικτυακό τόπο. Στα αποτελέσματα της επιθεώρησης έχει πρόσβαση μετά από διαδικασία αυθεντικοποίησης, για λόγους ασφαλείας του απορρήτου.

3 ΤΕΧΝΟΛΟΓΙΕΣ ΥΛΟΠΟΙΗΣΗΣ

Όπως αναλύθηκε και στα προηγούμενα κεφάλαια, το σύστημα αποτελεί ένα καινοτόμο σύστημα και το μοναδικό Ολοκληρωμένο Πληροφοριακό Σύστημα επιθεωρήσεων HACCP στην Ελλάδα. Με την ίδια φιλοσοφία επιλέχθηκαν και οι τεχνολογίες – πλατφόρμες με τις οποίες αναπτύχθηκε το σύστημα. Στο σύνολό τους αποτελούν λύσεις τελευταίας τεχνολογίας, με το συνδυασμό τους να παρουσιάζει υψηλό επιστημονικό και τεχνολογικό ενδιαφέρον.
Όπως αναφέρθηκε, η αρχιτεκτονική στην οποία αναπτύχθηκε το σύστημα είναι Υπηρεσιοστρεφής (SOA). Το σύστημα διαθέτει δύο Web Servers. Ο πρώτος (Apache) τρέχει σε Linux περιβάλλον σε λειτουργικό openSUSE 11.1 και φιλοξενεί την ιστοσελίδα, μέσω της οποίας γίνεται η εγκατάσταση και οι ενημερώσεις των εφαρμογών της εταιρείας συμβούλων και των επιθεωρητών. Ο δεύτερος (IIS) τρέχει σε περιβάλλον Windows σε λειτουργικό Windows Server 2008 R2 και φιλοξενεί την ιστοσελίδα, από την οποία ενημερώνονται οι πελάτες για τα αποτελέσματα των επιθεωρήσεων. Τα Java Web Services εκτελούνται στον Oracle WebLogic Application Server 11g R1 σε Linux περιβάλλον, όπως και η βάση δεδομένων Oracle Database 10g Express Edition. Το λειτουργικό σύστημα των υπολογιστών των επιθεωρητών είναι Windows 7, λειτουργικό που χρησιμοποιεί η πλειοψηφία των χρηστών, με αποτέλεσμα να μην χρειάζεται κάποιο εξειδικευμένο λειτουργικό για την υποστήριξη του συστήματος από τους επιθεωρητές. Χρησιμοποιήθηκαν ως Integrated Development Environments (IDE) το Oracle JDeveloper 11g Update 2 για την ανάπτυξη των Java Web Services και το Microsoft Visual Studio 2008 SP1 για την ανάπτυξη των εφαρμογών της εταιρείας συμβούλων και των επιθεωρητών, όπως επίσης και της Web εφαρμογής που προβάλει στους πελάτες τα αποτελέσματα των επιθεωρήσεων. Η τελευταία αναπτύχθηκε σε Ajax Framework με τεχνολογία ASP.NET Ajax. Τα αποτελέσματα προβάλλονται σε χάρτη μέσω του Google Maps API. Όλες οι Windows εφαρμογές χρησιμοποίησαν το .NET Framework 3.5 SP1 ενώ το γραφικό υποσύστημα για την δημιουργία διεπαφών χρήστη ήταν το Windows Presentation Foundation (WPF) που αποτελεί το πιο σύγχρονο γραφικό υποσύστημα δημιουργίας Desktop εφαρμογών της Microsoft. Όπως αναφέρθηκε και παραπάνω η εγκατάσταση και οι ενημερώσεις των εφαρμογών γίνονται διαδικτυακά μέσω ιστοσελίδας. Για την επίτευξη αυτής της λειτουργικότητας το Deployment των εφαρμογών έγινε μέσω της τεχνολογίας ClickOnce. Επίσης μέσω των Crystal Reports επιτεύχθηκε η αυτόματη εξαγωγή των αναφορών από την εταιρεία συμβούλων για τους πελάτες σε αρχεία της μορφής doc. Για όλα τα παραπάνω η ανάπτυξη έγινε μέσω των γλωσσών προγραμματισμού Java, C# καθώς και των XML, XAML, JavaScript, SQL και ASP.NET.

Παρακάτω παρατίθενται συγκεντρωτικά οι τεχνολογίες και οι πλατφόρμες που χρησιμοποιήθηκαν:

Architecture	Service-Oriented Architecture (SOA)
Web Server	1. Apache HTTP Server 2.2.15 2. Microsoft Internet Information Services 7.5
Application Server	Oracle WebLogic Server 11g R1
RDBMS	Oracle Database 10g Express Edition
Server OS	1. Linux openSUSE 11.1 2. Windows Server 2008 R2
Client OS	Microsoft Windows 7
IDE	1. Oracle JDeveloper 11g Update2 2. Microsoft Visual Studio 2008 SP1
Internet API	Java Web Services
Web Application Framework (Ajax Framework)	ASP.NET Ajax
Web Mapping	Google Maps API
Desktop Application Framework	.NET Framework 3.5 SP1
Graphical Subsystem	Windows Presentation Foundation (WPF)
Technology for Deploying WPF-based Software	ClickOnce
Programming Languages	XML, Java, C#, XAML, JavaScript, SQL, ASP.NET
Reporting Software	SAP Crystal Reports

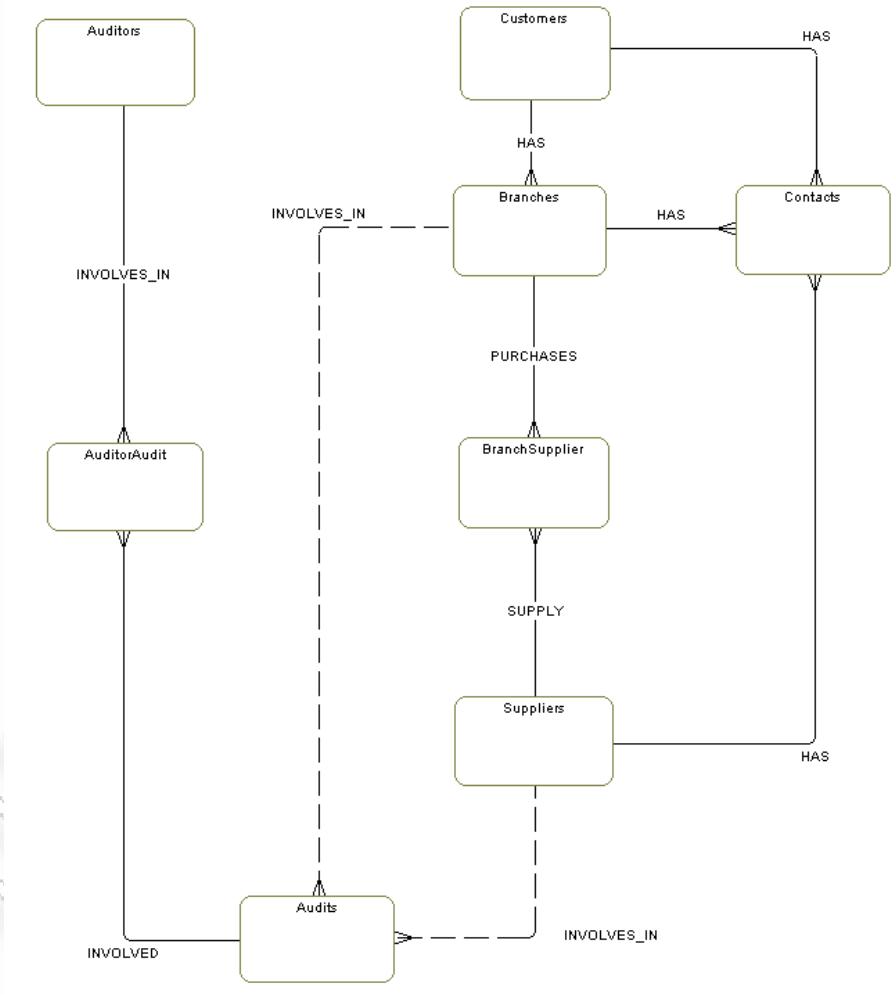
Να σημειωθεί ότι το σύστημα διαθέτει Servers που λειτουργούν και σε περιβάλλον Windows και σε Linux (ετερογενές σύστημα), εφ' όσον σε πραγματικές συνθήκες η πλειοψηφία της αγοράς υποστηρίζει κα τις δύο πλατφόρμες. Επομένως, μελετήθηκε η συμπεριφορά ενός τέτοιου συστήματος, όπως επίσης και η ολοκλήρωση των επιμέρους διαδικασιών του σε ετερογενές περιβάλλον.

4 ΛΟΓΙΚΗ ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΟΣ

Για το σχεδιασμό της βάσης, σε υψηλό επίπεδο αφαίρεσης, είναι απαραίτητος ο καθορισμός των οντοτήτων του συστήματος. Το σύστημα της εταιρείας συμβούλων αποτελείται από τις εξής οντότητες:

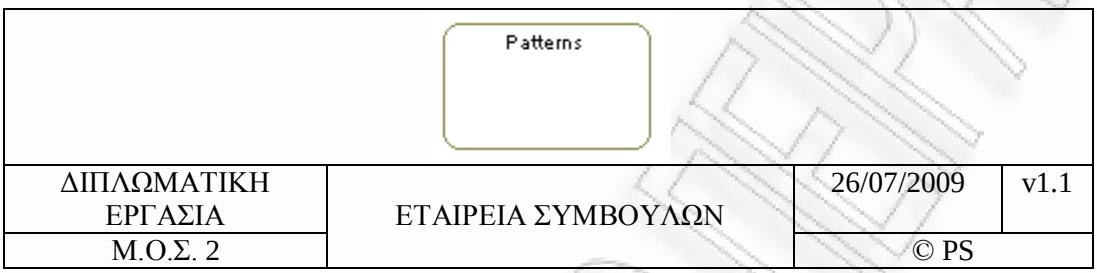
- Επιθεωρητές (Auditors)
- Επιθεωρήσεις (Audits)
- Πελάτες (Customers)
- Υποκαταστήματα (Branches)
- Προμηθευτές (Suppliers)
- Επαφές (Contacts)

Παρακάτω (Εικόνα 4.1) απεικονίζεται το Εννοιολογικό Μοντέλο (Μοντέλο Οντοτήτων – Συσχετίσεων) του συστήματος της εταιρείας συμβούλων, πάνω στο οποίο θα βασιστεί η δημιουργία της βάσης δεδομένων.



Εικόνα 4.1 Μοντέλο Οντοτήτων – Συσχετίσεων του συστήματος.

Παρατηρούμε ότι εκτός από τις οντότητες για κάθε ρόλο του συστήματος, υπάρχουν και δύο επιπλέον. Οι AuditorAudit και BranchSupplier. Αυτές έχουν δημιουργηθεί, διότι οι σχέσεις Επιθεωρητή – Επιθεώρησης και Υποκατάστημα – Προμηθευτής είναι πολλά προς πολλά, κάτι το οποίο στο σχεδιασμό βάσεων δεδομένων προτιμάται να αποφεύγεται.
Επίσης υπάρχει ανεξάρτητη οντότητα, η οποία δεν συνδέεται με τις προηγούμενες του συστήματος. Η ύπαρξή της κρίνεται αναγκαία για την συγκράτηση των προτύπων των ερωτηματολογίων από τα οποία ο διαχειριστής θα παράγει τα ερωτηματολόγια για κάθε επιθεώρηση.



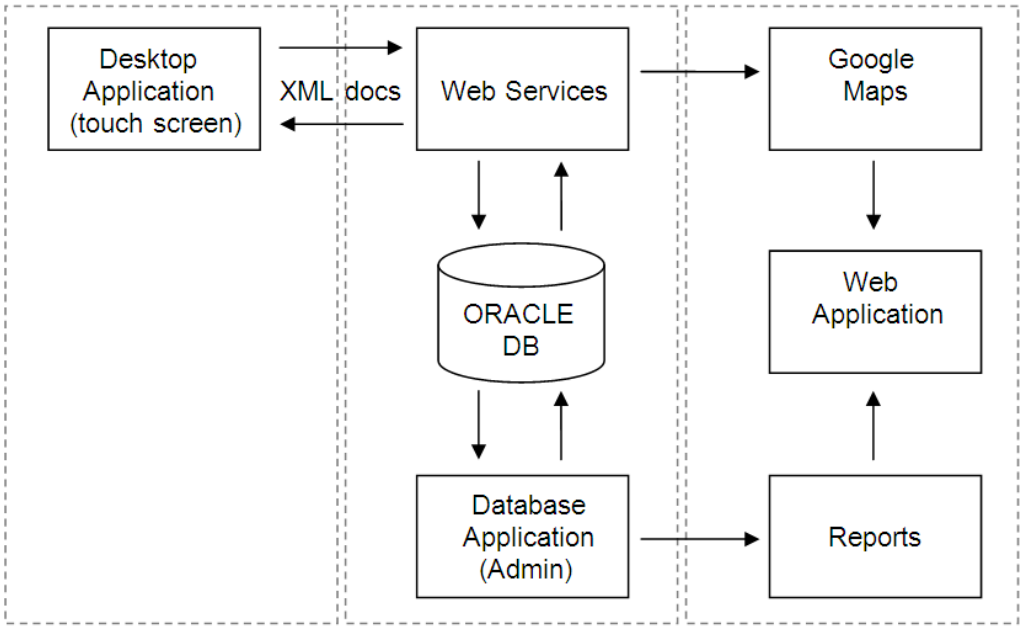
Εικόνα 4.2 Οντότητα προτύπων.

5 ΦΥΣΙΚΗ ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΟΣ

Όπως προέκυψε από τις απαιτήσεις του συστήματος (Κεφάλαιο 2), το σύστημα θα αποτελείται από τρία επιμέρους αυτόνομα συστήματα (Εικόνα 5.1). Σε πιο χαμηλό επίπεδο αφαίρεσης το σύστημα αποτελείται από τα παρακάτω υποσυστήματα:

- Σύστημα διαχείρισης των δεδομένων (C# WPF & Crystal Reports) και εξαγωγής αναφορών από την εταιρεία συμβούλων (Αποστολή και Λήψη ερωτηματολογίων – αναφορών μέσω Java Web Services)
- Σύστημα για την καταγραφή των δεδομένων της επιθεώρησης από τους επιθεωρητές (C# WPF Application)
- Σύστημα προβολής των αποτελεσμάτων στο QC (Ajax website)

Παρακάτω απεικονίζονται τα τρία επιμέρους συστήματα:



Εικόνα 5.1 Τα τρία (3) επίπεδα του ολοκληρωμένου συστήματος.

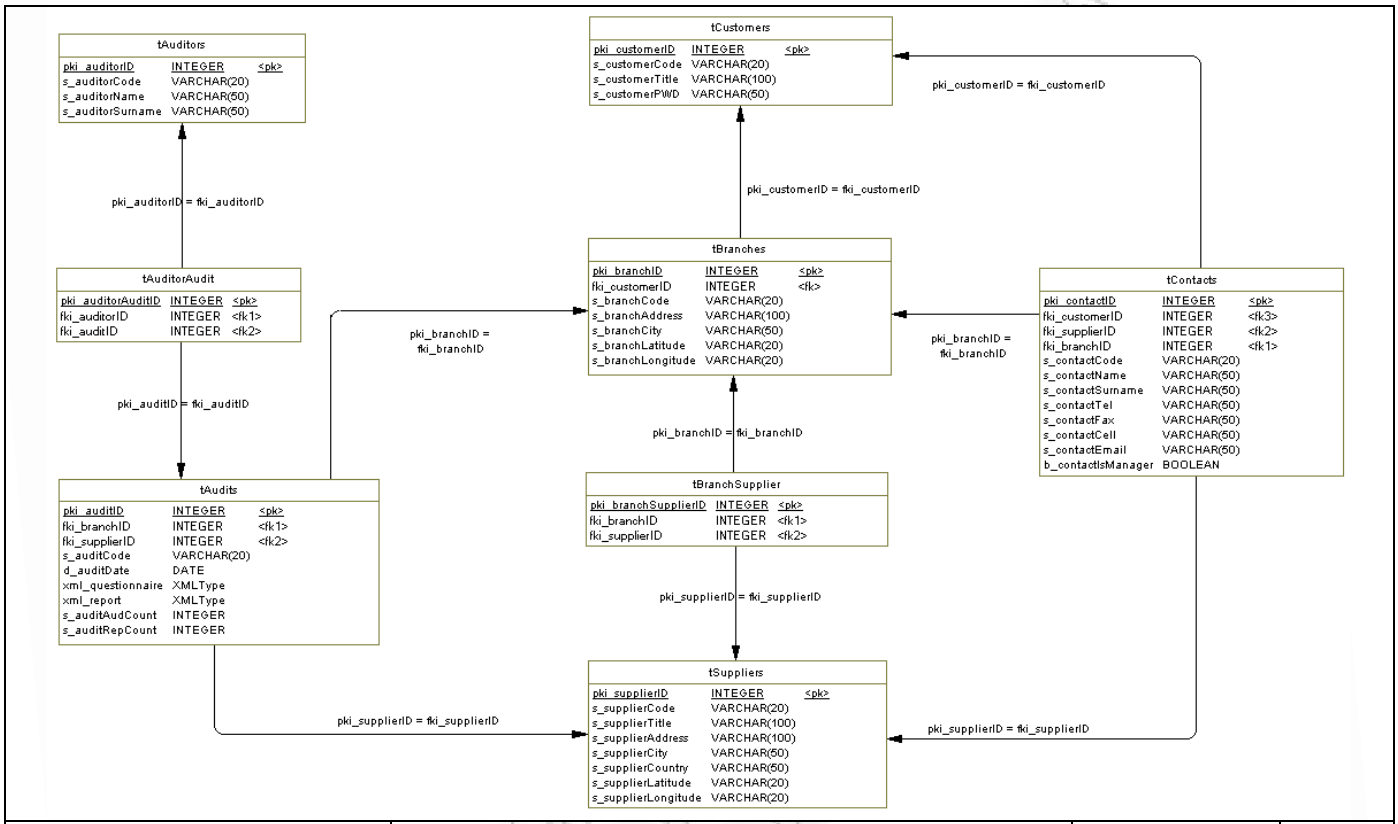
Το πρώτο υποσύστημα (μεσαίο ορθογώνιο σχήμα, εικόνα 5.1), αποτελεί το κύριο υποσύστημα του συστήματος. Σε αυτό υπάρχει η βάση δεδομένων που συγκρατεί όλα τα στοιχεία των πελατών καθώς και τα ερωτηματολόγια – αναφορές των επιθεωρήσεων. Η βάση δεδομένων τρέχει σε περιβάλλον Linux ενώ η εφαρμογή διαχείρισης της βάσης (Back-end) σε περιβάλλον Windows. Επίσης σε αυτό το επίπεδο (υποσύστημα) φιλοξενούνται τα Java Web Services, σε περιβάλλον Linux, που είναι υπεύθυνα για την αποστολή των ερωτηματολογίων στους επιθεωρητές, την λήψη των αναφορών τους και την προώθηση των αποτελεσμάτων στη Web εφαρμογή των πελατών.
Το δεύτερο υποσύστημα (αριστερό ορθογώνιο σχήμα, εικόνα 5.1), είναι η εφαρμογή που χρησιμοποιούν οι επιθεωρητές αρχικά για την λήψη των ερωτηματολογίων και στη συνέχεια για την αποστολή των αναφορών με το πέρας της επιθεώρησης.
Το τρίτο και τελευταίο υποσύστημα αριστερό ορθογώνιο σχήμα, εικόνα 5.1), είναι η Web εφαρμογή μέσω της οποίας το QC των πελατών μπορεί να ενημερώνεται για τα αποτελέσματα των επιθεωρήσεων στα υποκαταστήματα και στους προμηθευτές τους. Τα αποτελέσματα προβάλλονται σε μορφή καρφίτσας σε χάρτη (Google Maps API).

Με βάση το Μοντέλο Οντοτήτων – Συσχετίσεων (Μ.Ο.Σ.), της προηγούμενης ενότητας, σχεδιάστηκε η βάση δεδομένων του συστήματος. Για κάθε οντότητα του Μ.Ο.Σ. σχεδιάστηκε ένας πίνακας. Επομένως σχεδιάστηκαν οι ακόλουθοι οχτώ (8) πίνακες, οι οποίοι θα συγκρατούν τα εξής δεδομένα:

1. tAuditors
   Στοιχεία κάθε επιθεωρητή.
2. tAudits
   Κωδικός του υποκαταστήματος ή του προμηθευτή που θα επιθεωρηθεί, την ημερομηνία επιθεώρησης καθώς και τα ερωτηματολόγια – αναφορές των επιθεωρήσεων σε μορφή itps (XML encrypted). Επίσης θα υπάρχει πεδίο που θα συγκρατεί των συνολικό αριθμό επιθεωρητών που εμπλέκονται στην επιθεώρηση και πεδίο για το πλήθος των αναφορών που έχουν αποσταλεί από τους επιθεωρητές. Με τα δύο τελευταία πεδία δίνεται η δυνατότητα ελέγχου από το σύστημα του αριθμού των αναφορών που εκκρεμούν, σε περίπτωση που εμπλέκονται περισσότεροι του ενός επιθεωρητές στην επιθεώρηση.
3. tCustomers
   Κωδικός και επωνυμία του πελάτη.
4. tBranches
   Στοιχεία του υποκαταστήματος.
5. tSuppliers
   Στοιχεία του προμηθευτή.
6. tContacts
   Στοιχεία των επαφών του πελάτη ή των υποκαταστημάτων – προμηθευτών.
7. tAuditorAudit
   Συνδυασμός των κωδικών επιθεωρητή – επιθεώρησης.
8. tBranchSupplier
   Συνδυασμός των κωδικών υποκατάστημα – προμηθευτής.

Το πρωτεύον κλειδί κάθε πίνακα είναι ακέραιος αριθμός, ο οποίος παίρνει αυτόματη αρίθμηση. Στους πίνακες για τους οποίους οι εγγραφές απαιτούν κωδικό για τη διαχείριση τους από το σύστημα (π.χ. κωδικός επιθεωρητή), δεν θα εμφανίζεται το πρωτεύον κλειδί (αποτελεί πληροφορία της βάσης και όχι του χρήστη) αλλά θα υπάρχει πεδίο (s_<ΟΝΟΜΑ ΠΙΝΑΚΑ>Code) τύπου αλφαριθμητικό (String), όπου θα παίρνει αυτόματα τιμή της μορφής Z–XXX–YY (Z = διακριτικό κωδικού, XXX = αύξων αριθμός, YY = τα 2 τελευταία ψηφία του τρέχοντος έτους).

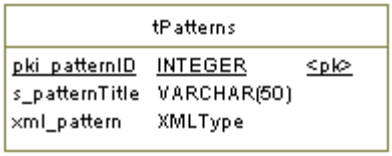
Στην εικόνα 5.2 απεικονίζεται το Σχεσιακό Σχήμα (Σχεσιακό Μοντέλο) του συστήματος. Προβάλλονται οι πίνακες της βάσης, τα πρωτεύοντα και τα ξένα κλειδιά, τα υπόλοιπα πεδία και οι σχέσεις των πινάκων.



Εικόνα 5.2 Σχεσιακό Μοντέλο του συστήματος.

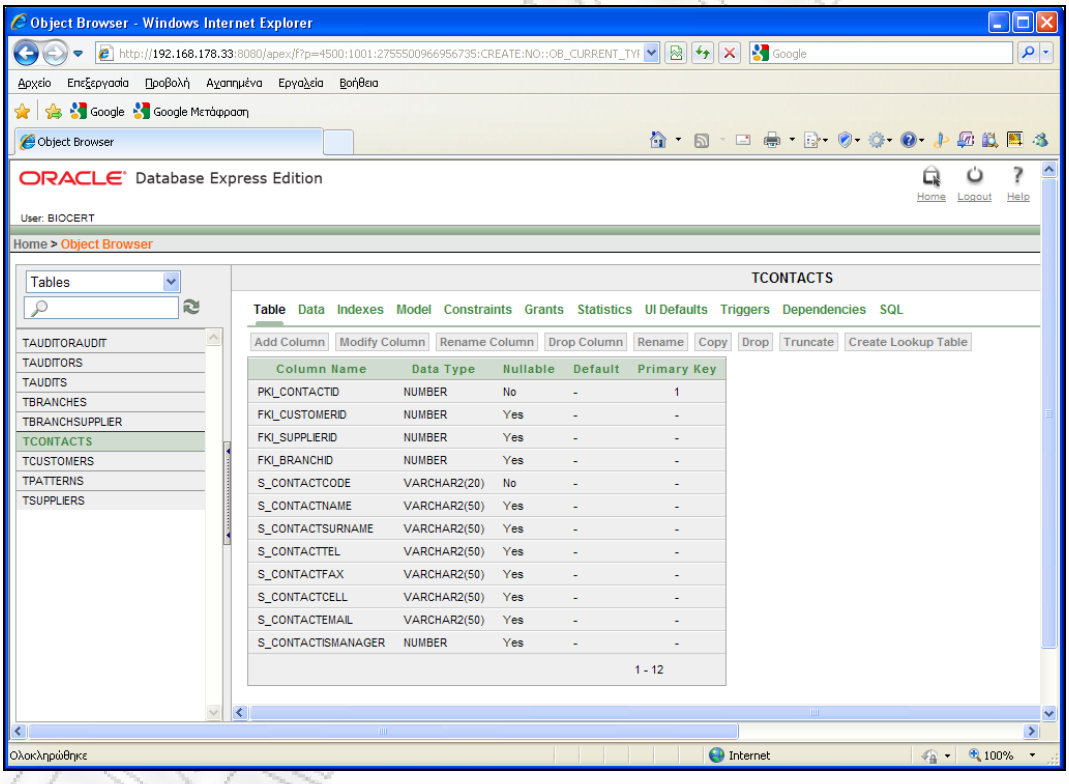
Το διακριτικό κάθε πίνακα είναι το όνομα της οντότητας ακολουθούμενο από το γράμμα ‘t’ (table). t<ΟΝΟΜΑ ΟΝΤΟΤΗΤΑΣ> (π.χ. tAuditors)
Τα ονόματα των πεδίων αποτελούνται από δύο (2) μέρη. Το πρώτο, πριν από την κάτω παύλα (underscore), υποδηλώνει τον τύπο και το δεύτερο υποδηλώνει τον πίνακα στον οποίο ανήκει και το χαρακτηριστικό του συγκρατεί.
Για τον κάθε τύπο αντιστοιχεί ένα γράμμα:
i → Integer (ακέραιος)
s → String (αλφαριθμητικό)
d → Date (ημερομηνία)
xml → XML
pk → Primary Key (πρωτεύον κλειδί)
fk → Foreign Key (ξένο κλειδί)

Επίσης για την οντότητα των προτύπων (Patterns) θα δημιουργηθεί ο αντίστοιχος πίνακας (tPatterns).



Εικόνα 5.3 Πίνακας Προτύπων.

Στο Server του συστήματος εγκαταστάθηκε το λειτουργικό σύστημα (Linux openSUSE 11.1) και στη συνέχεια η Oracle Database (10g Express Edition) [6]. Μέσω της διεπαφής διαχείρισης της Oracle (web-based) δημιουργήθηκε η βάση δεδομένων με τους πίνακες και τις συσχετίσεις τους, βάσει του Σχεσιακού Σχήματος. Με την ολοκλήρωση της διαδικασίας θα έχουμε τη δομή της Εικόνας 5.4



Εικόνα 5.4 Οι πίνακες της βάσης δεδομένων (Oracle RDBMS).

6 ΥΛΟΠΟΙΗΣΗ

Αρχικά ορίστηκε ο κύκλος ζωής του συστήματος. Αυτός αποτελείται από τα παρακάτω στάδια:

1. Συλλογή απαιτήσεων
2. Λογική σχεδίαση
3. Φυσική σχεδίαση
4. Ανάπτυξη συστήματος
5. Λειτουργία συστήματος
6. Ανατροφοδότηση συστήματος

Για το 6^{ο} στάδιο του κύκλου ζωής, η ανατροφοδότηση (feedback) προέκυψε μετά από λειτουργία του συστήματος (5^{ο} στάδιο) για τέσσερις εβδομάδες στην εταιρεία Biocert, σε πραγματικές συνθήκες. Να σημειωθεί ότι η παραπάνω εταιρεία παρείχε όλο το υλικό (hardware) που απαιτήθηκε για την υλοποίηση και λειτουργία του συστήματος (Servers, Tablet PC κ.λπ.).

Το σύστημα αποτελείται από πέντε (5) αυτόνομα projects.

1. Delta Library
2. Delta Server
3. Delta Client
4. Delta Services
5. Delta Web

Η συγγραφή του κώδικα σε όλα τα projects είναι ενιαία. Τα ονόματα των μεταβλητών έχουν ως δομή τη μορφή x_yyyZzzzz. Όπου x είναι το γράμμα a (argument) αν η μεταβλητή είναι παράμετρος μεθόδου, l (local) αν είναι τοπική μεταβλητή και g αν είναι global μεταβλητή. Μετά το underscore τα τρία γράμματα (yyy) δηλώνουν τον τύπο της:
str → string (αλφαριθμητικό)
int → integer (ακέραιος)
dbl → double (πραγματικός)
bln → boolean (λογική)
xml → xml
arr → array (πίνακας)
lbl → label (ετικέτα)
Τέλος ακολουθεί το όνομα της μεταβλητής (Zzz…) το οποίο και την χαρακτηρίζει.

Το project Delta Library είναι η βιβλιοθήκη του συστήματος, μέσω της οποίας κρυπτογραφούνται τα ερωτηματολόγια και αποκρυπτογραφούνται οι αναφορές. Αυτό είναι απαραίτητο διότι τα ερωτηματολόγια περιέχουν ευαίσθητες πληροφορίες της εταιρείας συμβούλων ενώ οι αναφορές ευαίσθητες πληροφορίες των πελατών. Έτσι υπάρχει ασφάλεια τόσο στην αποστολή – λήψη των XML αρχείων όσο και στην αποθήκευσή τους στη βάση δεδομένων και τοπικά τους υπολογιστές των επιθεωρητών. Μετά την κρυπτογράφηση, τα XML αρχεία μετατρέπονται σε αρχεία δικής μας επέκτασης (.itps). Επίσης, μέσω της βιβλιοθήκης παράγονται οι τυχαίοι κωδικοί πρόσβασης των πελατών, όταν από την εφαρμογή του διαχειριστή δημιουργήσουμε έναν νέο πελάτη.

Το project Delta Server αποτελεί την εφαρμογή του διαχειριστή. Μέσω της εφαρμογής, ο διαχειριστής διαχειρίζεται τους επιθεωρητές της εταιρείας, τους πελάτες και τα καταστήματα τους, τους προμηθευτές και τις επαφές τους. Έχει τη δυνατότητα να δημιουργήσει τα πρότυπα των ερωτηματολογίων και μέσω αυτών να δημιουργήσει επιθεωρήσεις τις οποίες μπορεί να αναθέσει σε έναν ή περισσότερους επιθεωρητές. Τέλος, μπορεί από τις αναφορές των επιθεωρητών να εξάγει τις τελικές αναφορές που θα αποστέλλονται στους πελάτες.

Το project Delta Client είναι η εφαρμογή που διαχειρίζονται οι επιθεωρητές. Μπορούν να προβούν στη λήψη των διαθέσιμων ερωτηματολογίων που τους ανατέθηκαν και στη συνέχεια να προχωρήσουν στην επιθεώρηση. Τέλος αποστέλλουν τις αναφορές της επιθεώρησης στην εταιρεία συμβούλων.

Το Delta Services αποτελεί τη ραχοκοκαλιά (backbone) του συστήματος. Περιλαμβάνει τα Web Services μέσω των οποίων ολοκληρώνονται οι επιμέρους διαδικασίες. Συγκεκριμένα, μέσω αυτών αποστέλλονται τα ερωτηματολόγια στους επιθεωρητές και οι αναφορές στην εταιρεία συμβούλων. Επίσης, συλλέγει όλες τις απαραίτητες πληροφορίες μέσω των αναφορών, ώστε να προβάλλει στους πελάτες τα αποτελέσματα των επιθεωρήσεων στους χάρτες (Google Maps).

Τέλος, το Delta Web είναι μια web εφαρμογή μέσω της οποίας οι πελάτες, εισάγοντας τον κωδικό τους, μπορούν να δουν σε Google Maps τα αποτελέσματα των επιθεωρήσεων των καταστημάτων και των προμηθευτών τους.

6.1 Delta Library

Είναι ένα μη εκτελέσιμο project, δηλαδή δεν παράγεται κάποιο εκτελέσιμο αρχείο (exe) αλλά ένα dll (dynamic-link library) αρχείο το οποίο εισάγεται και χρησιμοποιείται στα project Delta Server, Delta Client και Delta Web. Είναι γραμμένο σε C# σε .NET Framework 3.5 SP1 και έχει αναπτυχθεί με χρήση του Visual Studio 2008 SP1 IDE. Αποτελείται από τρεις (3) κλάσεις.

Η κλάση `public class clEncryption` αποτελείται από μία μέθοδο την `public string Encrypt(string a_strXML, string a_strPath)`. Είναι τύπου string και επομένως επιστρέφει αλφαριθμητικό. Επιστρέφει δηλαδή το itps αρχείο. Ως παραμέτρους δέχεται το αρχικό XML αρχείο καθώς και την τοποθεσία που θα δημιουργηθούν τα προσωρινά αρχεία, ώστε να γίνει η κρυπτογράφηση. Η διαδρομή επιστρέφεται δυναμικά από το project μέσω της εντολής:

```
Environment.GetFolderPath(Environment.SpecialFolder.
CommonApplicationData)
```

Είναι ένας κρυφός φάκελος όπου στα Windows XP, για παράδειγμα, είναι η διαδρομή «C:\Documents and Settings\All Users\Application Data»
Με την ολοκλήρωση της διαδικασίας η εφαρμογή διαγράφει το προσωρινά αρχεία (xml & itps) από τον φάκελο.

```
if (File.Exists(l_strFile))
{
      File.Delete(l_strFile);
}
```

Αντίστοιχη λειτουργικότητα έχει και η κλάση `public class clDecryption` μέσω της μεθόδου `public string Decrypt(string a_strEnc, string a_strPath`. Αντίστοιχα δέχεται ως όρισμα το itps αρχείο και επιστρέφει το αρχικό xml αρχείο.
Η τρίτη κλάση είναι η `public class clSecurity : IDisposable`. Είναι η κλάση που αποτελεί την ασφάλεια του συστήματος. Περιέχει τον κωδικό του διαχειριστή και τον κωδικό των επιθεωρητών. Ο δεύτερος δεν είναι στη βάση δεδομένων διότι η εφαρμογή των επιθεωρητών Delta Client θα μπορεί να λειτουργήσει και offline, δηλαδή χωρίς την ύπαρξη Internet, στην περίπτωση για παράδειγμα όπου η επιθεώρηση γίνεται σε χώρο όπου δεν υπάρχει πρόσβαση στο Διαδίκτυο. Επομένως, είναι απαραίτητη η υποστήριξη της σύνδεσης στην εφαρμογή, χωρίς έλεγχο του κωδικού στη βάση δεδομένων αλλά από το τοπικό Delta Library dll. Τόσο ο κωδικός του διαχειριστή, ο οποίος είναι ένας γιατί ένας θα είναι και ο διαχειριστής του συστήματος, όσο και ο κωδικός των επιθεωρητών, ο οποίος είναι κοινός, δεν είναι σε «καθαρή» μορφή. Όπως μπορείτε να δείτε στη συνέχεια υπάρχουν οι συνόψεις των κωδικών, όπως προέκυψαν μετά την εφαρμογή του κρυπτογραφικού αλγορίθμου σύνοψης SHA-1. Αυτή είναι μια πολύ καλή δικλείδα ασφαλείας μιας και από μια σύνοψη δεν μπορεί να επανέλθει το αρχικό κείμενο.

```
private const string g_adminPWD    =
"20b97eb8e9d316d3d3c0d4530381fda3c053d9a8";
private const string g_auditorPWD =
"f1be2835f2fecb9408fd6287f5b72e22d0581674";
```

Άρα, αν υποθέσουμε ότι κάποιος αποκτά πρόσβαση στο dll και μπορεί να δει εντός των κλάσεων, δεν θα μπορέσει να συνδεθεί ούτε στην εφαρμογή του διαχειριστή αλλά ούτε και στην αντίστοιχη του επιθεωρητή. Κι αυτό γιατί η κάθε εφαρμογή δέχεται ως είσοδο την «καθαρή» μορφή του κωδικού και συγκρίνει αν είναι ίδιες οι συνόψεις τους. Στην περίπτωση του διαχειριστή θα πρέπει να εισαχθεί ο κωδικός !db@bc0# ενώ στην περίπτωση του επιθεωρητή ο κωδικός 82467305.
Μέσω της μεθόδου `private static string GenerateCode(string a_strCode, Encoding a_enc)` επιστρέφεται η σύνοψη του κωδικού που έχει εισαχθεί από το χρήστη ώστε να συγκριθεί με τις συνόψεις που συγκρατεί το dll.

```
byte[] buffer = a_enc.GetBytes(a_strCode);
SHA1CryptoServiceProvider cryptoTransformSHA1 = new
SHA1CryptoServiceProvider();
string l_hash = BitConverter.ToString(cryptoTransformSHA1.
ComputeHash(buffer)).Replace("-", "");
return l_hash.ToLower();
```

Τέλος, στην κλάση υπάρχει και η μέθοδος `public string RandomPassword()` η οποία επιστρέφει έναν τυχαίο κωδικό οχτώ ψηφίων, ο οποίος είναι και ο κωδικός που δημιουργείται όταν εισάγουμε έναν καινούριο πελάτη μέσω του Delta Server, όπως θα δούμε στη συνέχεια. Τα ψηφία από τα οποία θα αποτελείται ο κωδικός διαβάζονται από τον ακόλουθο πίνακα χαρακτήρων: `char[] l_characters = "!@#$%&*0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ".ToCharArray();` και επιλέγονται τυχαία μέσω του αντικειμένου `Random RndGenerator = new Random();` το οποίο επιστρέφει τη θέση του επόμενου χαρακτήρα του κωδικού `int x = RndGenerator.Next(1,l_characters.Length);`

Επίσης, υπάρχει έλεγχος ώστε ο κάθε χαρακτήρας του κωδικού να εμφανίζεται μόνο μία φορά. Αυτό επιτυγχάνεται με το παρακάτω block κώδικα:

```
string l_pwd = string.Empty;
```

```
for (int i = 0; i < 8; i++)
{
      int x = RndGenerator.Next(1, l_characters.Length);
      if (!l_pwd.Contains(l_characters.GetValue(x).ToString()))
      {
            l_pwd += l_characters.GetValue(x);
      }
      else
      {
            i--;
      }
}
return l_pwd;
```

6.2 Delta Server

Το project περιλαμβάνει την εφαρμογή που χρησιμοποιεί ο διαχειριστής της εταιρείας συμβούλων, μέσω της οποίας διαχειρίζεται τους επιθεωρητές, τους πελάτες με τα υποκαταστήματά και τους προμηθευτές τους με τα πρότυπα ερωτηματολόγια. Επίσης δημιουργεί επιθεωρήσεις αναθέτοντάς τες σε επιθεωρητές. Τέλος, μέσω των αναφορών που λαμβάνει από τους επιθεωρητές εξάγει τις τελικές αναφορές που αποστέλλονται στους πελάτες. Είναι γραμμένο σε C# σε .NET Framework 3.5 SP1 και έχει αναπτυχθεί με χρήση του Visual Studio 2008 SP1 IDE. Στα βασικά μέρη της δομή του περιλαμβάνονται classes, windows, controls και crystal report.

Οι τρεις κλάσεις του project είναι βοηθητικές. Η πρώτη είναι η `class Utilities : IDisposable`. Περιέχει μία μέθοδο `public BitmapImage CreateImage(string a_strPath)` η οποία δέχεται σαν όρισμα μία σχετική διαδρομή (relative path) και επιστρέφει την εικόνα της διαδρομής σε μορφή BitmapImage. Η παρούσα μέθοδος βρίσκει εφαρμογή στη προσθήκη εικόνων στα κουμπιά.
Η δεύτερη είναι η `public class CloseableTabItem : TabItem`. Είναι μια κλάση που χρησιμοποιείται για τη δημιουργία ενός control σε μορφή καρτέλας (Tab). Με αυτόν τον τρόπο το control της εφαρμογής προβάλλεται στο κεντρικό παράθυρο σε μορφή καρτέλας. Η μορφή της καρτέλας ορίζεται εντός του αρχείου Themes/generic.xaml που είναι τύπου ResourceDictionary.
Τρίτη κλάση είναι η `class clCodes`, η οποία μέσω των οχτώ μεθόδων της, που θα αναλυθούν στη συνέχεια, δημιουργεί τους κωδικούς για κάθε νέα εγγραφή (επιθεωρητή, πελάτη, προμηθευτή, υποκαταστήματος, επαφής, επιθεώρησης). Όπως αναφέρθηκε και στη φυσική σχεδίαση του συστήματος, οι κωδικοί είναι της μορφής Z–XXX–YY (Z = διακριτικό κωδικού, XXX = αύξων αριθμός, YY = τα 2 τελευταία ψηφία του τρέχοντος έτους). Εξαίρεση αποτελεί ο κωδικός της επιθεώρησης όπου περιέχει αρχικά τρία αντί για ένα ψηφία και είναι της μορφής AUD–XXX–YY (XXX = αύξων αριθμός, YY = τα 2 τελευταία ψηφία του τρέχοντος έτους)

Αναλυτικά οι μέθοδοι της κλάσης:

```
public String NewAuditorCode(String a_strLastAuditorCode)
public String NewCustomerCode(String a_strLastCustomerCode)
public String NewSupplierCode(String a_strLastSupplierCode)
public String NewBranchCode(String a_strLastBranchCode)
public String NewCustomerContactCode(String a_strLastContactCode)
public String NewSupplierContactCode(String a_strLastContactCode)
```

```
public String NewBranchContactCode(String a_strLastContactCode)
public String NewAuditCode(String a_strLastAuditCode)
```

Και οι οχτώ κλάσεις είναι τύπου αλφαριθμητικό και δέχονται ως όρισμα τον τελευταίο κωδικό. Η κλάση είναι υπεύθυνη να δημιουργήσει τον επόμενο και να τον επιστρέψει. Αρχικά χωρίζει από τον τελευταίο κωδικό τα τρία μέρη του.

```
String[] l_strLastAuditorCode=a_strLastAuditorCode.Split('-');
```

Το δεύτερο όρισμα είναι ο αύξων αριθμός, επομένως αυξάνει την τιμή του κατά ένα.

```
String l_strNewAuditorID =
      (int.Parse(l_strLastAuditorCode[1]) + 1).ToString();
```

Στην περίπτωση που ο αριθμός είναι μονοψήφιος ή διψήφιος προσθέτει στην αρχή μηδενικά, ώστε όλοι οι κωδικοί να έχουν την ίδια δομή.

```
String l_strCurrentYear = DateTime.Now.Year.ToString();
      if (l_strNewAuditorID.Length < 3)
      {
            if (l_strNewAuditorID.Length == 2)
            {
                  l_strNewAuditorID = "0" + l_strNewAuditorID;
            }
            else
            {
                  l_strNewAuditorID = "00" + l_strNewAuditorID;
            }
      }
```

Τέλος προσθέτει στην αρχή το διακριτικό του κωδικού (π.χ. Α για τους επιθεωρητές) και στο τέλος τα δύο τελευταία ψηφία του τρέχοντος έτους.

```
      String l_strNewAuditorCode = "A-" +
            l_strNewAuditorID.ToString() + "-" +
            l_strCurrentYear.Substring(2);
      return l_strNewAuditorCode;
```

Η εφαρμογή περιλαμβάνει δεκατέσσερα παράθυρα (windows), εννιά UserControls και ένα crystal report.

Το πρώτο παράθυρο εμφανίζεται κατά την εκκίνηση της εφαρμογής. Είναι το παράθυρο της σύνδεσης `public partial class winLogin : Window`. Η σύνδεση είναι υποχρεωτική διότι μέσω της εφαρμογής ο χρήστης έχει πρόσβαση σε ευαίσθητες πληροφορίες των πελατών. Αρχικά περιέχει μια static μεταβλητή η οποία συγκρατεί τη διαδρομή που χρησιμοποιεί η εφαρμογή για την αποθήκευση των προσωρινών αρχείων που απαιτούνται για τη κρυπτογράφηση και αποκρυπτογράφηση των ερωτηματολογίων - αναφορών.

```
public static string g_strPath = Environment.GetFolderPath(
Environment.SpecialFolder.CommonApplicationData)
+ "\\Delta Server";
```

Στο event `private void Window_Loaded(object sender, RoutedEventArgs e)` που καλείται μόλις ξεκινά το παράθυρο, ελέγχεται αν υπάρχει ο παραπάνω φάκελος αλλιώς τον δημιουργεί.

```
if (!Directory.Exists(g_strPath))
{
      Directory.CreateDirectory(g_strPath);
}
```

Το παράθυρο έχει δύο πεδία κειμένου. Ένα για το όνομα χρήστη και ένα για τον κωδικό. Το όνομα χρήστη έχει πάντα την τιμή 'Administrator'. Επίσης υπάρχουν δύο κουμπιά το 'OK' και το 'Cancel'. Στο πάτημα του κουμπιού 'OK' καθώς και στο

πάτημα του πλήκτρου ‘Enter’ από το πληκτρολόγιο `if (e.Key == Key.Enter)`, όταν ο κέρσορας βρίσκεται εντός των δύο πεδίων κειμένου, καλείται η μέθοδος `private void Authenticate()`. Εντός της μεθόδου αρχικά δημιουργείται ένα αντικείμενο τύπου clSecurity από το Delta Library dll. Καλείται η μέθοδος CheckAdminCode στην οποία εισάγεται ως όρισμα την τιμή του πεδίου ‘Κωδικός’. Αν επιτύχει η ταυτοποίηση του κωδικού, εμφανίζεται το κύριο παράθυρο της εφαρμογής, διαφορετικά προβάλλεται σχετικό ενημερωτικό μήνυμα.

```
using (clSecurity l_clSec = new clSecurity())
{
  if (l_clSec.CheckAdminCode(this.txtPassword.Password))
  {
    winMain l_winMain = new winMain();
    this.Visibility = Visibility.Hidden;
    l_winMain.Show();
    this.Close();
  }
  else
  {
    MessageBox.Show("Η σύνδεση απέτυχε!", "Delta Server",
    MessageBoxButton.OK, MessageBoxImage.Exclamation,
    MessageBoxResult.OK);
  }
}
```

Στο πάτημα του κουμπιού ‘Cancel’ κλείνει το παράθυρο.

```
this.Close();
```

Το βασικό παράθυρο της εφαρμογής είναι το `public partial class winMain : Window`. Στον κατασκευαστή (constructor) του παραθύρου εισάγεται στο παράθυρο το event της κλάσης CloseableTabItem μέσω της οποίας τα controls προβάλλονται στο TabControl του παραθύρου με δυνατότητα να κλείνουν από το κουμπί που φέρει το καθένα.

```
public winMain()
{
      InitializeComponent();
      this.AddHandler(CloseableTabItem.CloseTabEvent, new
      RoutedEventHandler(this.CloseTab));
}
```

Για το κλείσιμο της καρτέλας καλείται η παρακάτω μέθοδος

```
private void CloseTab(object source, RoutedEventArgs args)
{
      TabItem tabItem = args.Source as TabItem;
      if (tabItem != null)
      {
            TabControl tabControl = tabItem.Parent as
            TabControl;
            if (tabControl != null)
                  tabControl.Items.Remove(tabItem);
      }
}
```

Το παράθυρο διαχειρίζεται το event `private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)` το οποίο κατά το κλείσιμο του παραθύρου.

```
if (!this.CloseApplication())
{
      e.Cancel = true;
}
```

Η μέθοδος που καλείται προβάλει στο χρήστη ενημερωτικό μήνυμα για την επιβεβαίωση του κλεισίματος του παραθύρου.

```
private Boolean CloseApplication()
{
      if ((MessageBox.Show("Πρόκειται να κλείσετε την
      εφαρμογή." + Environment.NewLine + "Θέλετε να
      συνεχίσετε;", "Delta Server", MessageBoxButton.YesNo,
      MessageBoxImage.Question) == MessageBoxResult.Yes))
      {
            return true;
      }
      else
      {
            return false;
      }
}
```

Το βασικό παράθυρο περιέχει δώδεκα κουμπιά.

Τα πρώτα εννέα κουμπιά (Επιθεωρητές, Πελάτες, Καταστήματα, Προμηθευτές, Επαφές, Προμήθειες, Ερωτη/λόγια, Επιθεωρήσεις, Αναφορές) αποτελούν τα βασικά κουμπιά της εφαρμογής ενώ τα τρία τελευταία (Καρτέλες, Περί…, Έξοδος) τα βοηθητικά. Όλα τα βασικά κουμπιά, πλην του κουμπιού 'Αναφορές', έχουν την ίδια συμπεριφορά στο πάτημά τους. Αρχικά ελέγχεται αν η αντίστοιχη καρτέλα (UserControl) υπάρχει στο tabControl. Σε αυτή την περίπτωση την επιλέγει, αλλιώς τη δημιουργεί.

```
foreach (TabItem item in this.tabWindows.Items)
{
      if (item.Name == "tabAuditors")
      {
            item.Focus();
            return;
      }
}
ucAuditorsGrid grdAuditors = new ucAuditorsGrid();
```

Τα controls των προηγούμενων κουμπιών (ucAuditorsGrid, ucCustomersGrid, ucBranchesGrid, ucSuppliersGrid, ucContactsGrid, ucSuppliesGrid, ucReportsGrid, ucAuditsGrid) στο σύνολό τους περιέχουν ένα πλέγμα (Grid) για την προβολή των εγγραφών και για το οποίο διαχειρίζονται το event που καλείται στη δημιουργία του

```
private void Grid_Loaded(object sender, RoutedEventArgs e).
      using (g_taAuditors = new
            dsBiocertTableAdapters.TAUDITORSTableAdapter())
      {
            this.g_taAuditors.Fill(this.g_dsBiocert.TAUDITORS);
      }
```

Το κάθε ένα διαβάζει τις εγγραφές από πίνακα της βάσης δεδομένων, μέσω του αντίστοιχου TableAdapter. Το event που καλείται με το διπλό κλικ του ποντικιού εντός της περιοχής του Grid όπως και το event με το πάτημα του κουμπιού 'Επεξεργασία' καλούν τη μοναδική μέθοδο που έχουν τα controls. Για παράδειγμα η μέθοδος των επιθεωρητών είναι η `this.AuditorEdit();`. Η μέθοδος αυτή ανοίγει το αντίστοιχο παράθυρο μέσω του οποίου ο διαχειριστής επεξεργάζεται τις εγγραφές. Μπορεί να δημιουργήσει, να διαγράψει ή να ανανεώσει τα στοιχεία των εγγραφών.

```
winAuditors l_winAuditor = new winAuditors();
l_winAuditor.ShowDialog();
using (g_taAuditors = new
      dsBiocertTableAdapters.TAUDITORSTableAdapter())
{
```

```
            this.g_taAuditors.Fill(this.g_dsBiocert.TAUDITORS);
      }
```

Αφού το παράθυρο κλείσει τότε ξαναδιαβάζονται οι εγγραφές από τη βάση για να ενημερωθεί το Grid με τις τελευταίες αλλαγές του διαχειριστή. Εξαίρεση αποτελεί το control των επαφών (ucContactsGrid) το οποίο περιέχει μόνο κουμπί ανανέωσης, με το οποίο ξαναδιαβάζονται οι εγγραφές από τη βάση (επαφές πελατών, καταστημάτων και προμηθευτών). Η διαχείριση των επαφών γίνεται από τα controls των πελατών, καταστημάτων και προμηθευτών (ucCustomersGrid, ucBranchesGrid, ucSuppliersGrid) με το κάθε ένα να διαχειρίζεται τις δικές του εγγραφές. Αυτό επιτυγχάνεται με το πάτημα του δεύτερου κουμπιού (Επαφές) που περιέχουν τα controls.

```
      winContacts l_winContacts = new winContacts();
      l_winContacts.ContactOrigin =
            (int)winContacts.ContactType.Customer;
      l_winContacts.ShowDialog();
```

Το παράθυρο του ανοίγει και για τα τρία είδη των επαφών είναι το ίδιο `public partial class winContacts : Window`. Από το control, μέσω ενός enumerator, ορίζεται σε ένα property του παραθύρου ο τύπος των επαφών (πελάτη, καταστήματος ή προμηθευτή).

```
      public enum ContactType
      {
            Customer = 1, Supplier = 2, Branch = 3
      }
      public int ContactOrigin
      {
            get
            {
                  return g_intContactType;
            }
            set
            {
                  g_intContactType = value;
            }
      }
```

Τα παράθυρα που διαχειρίζονται τις εγγραφές (winAuditors, winCustomers, winBranches, winSuppliers, winContacts, winSupplies, winReports, winAudits) αρχικά διαβάζουν τις εγγραφές από τη βάση και τις εισάγουν σε ένα αντικείμενο τύπου CollectionView.

```
this.g_taAuditors.Fill(this.g_dsBiocert.TAUDITORS);
this.g_tamAuditors.TAUDITORSTableAdapter = g_taAuditors;
this.g_dsBiocert.TAUDITORS.DefaultView.Sort = "S_AUDITORCODE";
this.DataContext = this.g_dsBiocert.TAUDITORS;
this.g_viewCurrentView=(CollectionView)
   (CollectionViewSource.GetDefaultView(this.g_dsBiocert.TAUDITORS));
```

Το κάθε ένα παράθυρο περιέχει οχτώ κουμπιά. Τα πρώτα τέσσερα παρέχουν τη δυνατότητα περιήγησης ανάμεσα στις εγγραφές. Ουσιαστικά διαχειρίζονται το CollectionView object.

Για τη μετάβαση στην πρώτη εγγραφή καλείται η παρακάτω μέθοδος:

```
this.g_viewCurrentView.MoveCurrentToFirst();
```

Η μετάβαση στην προηγούμενη εγγραφή γίνεται με το παρακάτω block κώδικα:

```
if (this.g_viewCurrentView.CurrentPosition > 0)
{
      this.g_viewCurrentView.MoveCurrentToPrevious();
}
```

Μετάβαση στην επόμενη εγγραφή:

```
if (this.g_viewCurrentView.CurrentPosition <
    this.g_viewCurrentView.Count -1)
{
      this.g_viewCurrentView.MoveCurrentToNext();
}
```

Μετάβαση στην τελευταία εγγραφή:

```
this.g_viewCurrentView.MoveCurrentToLast();
```

Με τα τέσσερα επόμενα κουμπιά των παραθύρων ο διαχειριστής εισάγει νέα εγγραφή, διαγράφει εγγραφή, αναιρεί όλες τις αλλαγές ή τις αποθηκεύει.

Διαγραφή εγγραφής:

```
if ((this.g_viewCurrentView.CurrentPosition > -1))
{
      var row =
      ((System.Data.DataRowView)(this.g_viewCurrentView.CurrentItem))
      .Row;
      row.Delete();
}
```

Εισαγωγή εγγραφής (π.χ. νέος επιθεωρητής):

```
var row = this.g_dsBiocert.TAUDITORS.NewTAUDITORSRow();
if (this.g_dsBiocert.TAUDITORS.Count == 0)
{
  String l_strCurrentYear = DateTime.Now.Year.ToString();
  row.S_AUDITORCODE = "A-001-" + l_strCurrentYear.Substring(2);
}
else
{
  clCodes l_clAuditor = new clCodes();
  int l_intLastRecord = -1;
  int l_intLastCode = 0;
  for (int i = 0; i < this.g_dsBiocert.TAUDITORS.Count; i++)
  {
    if (this.g_dsBiocert.TAUDITORS[i].RowState !=
        DataRowState.Deleted)
    {
      String[] l_strAuditorCode =
      this.g_dsBiocert.TAUDITORS[i]["S_AUDITORCODE"].
           ToString().Split('-');
      if (l_intLastCode < int.Parse(l_strAuditorCode[1]))
      {
        l_intLastCode = int.Parse(l_strAuditorCode[1]);
        l_intLastRecord = i;
      }
    }
  }
  if (l_intLastRecord == -1)
  {
    String l_strCurrentYear = DateTime.Now.Year.ToString();
    row.S_AUDITORCODE = "A-001-" + l_strCurrentYear.Substring(2);
  }
  else
  {
    row.S_AUDITORCODE =
    l_clAuditor.NewAuditorCode(this.
    g_dsBiocert.TAUDITORS[l_intLastRecord]
    ["S_AUDITORCODE"].ToString());
  }
}
row.S_AUDITORSURNAME = "[Επώνυμο Επιθεωρητή]";
```

```
row.S_AUDITORNAME = "[Όνομα Επιθεωρητή]";
this.g_dsBiocert.TAUDITORS.AddTAUDITORSRow(row);
this.g_viewCurrentView.MoveCurrentToLast();
```

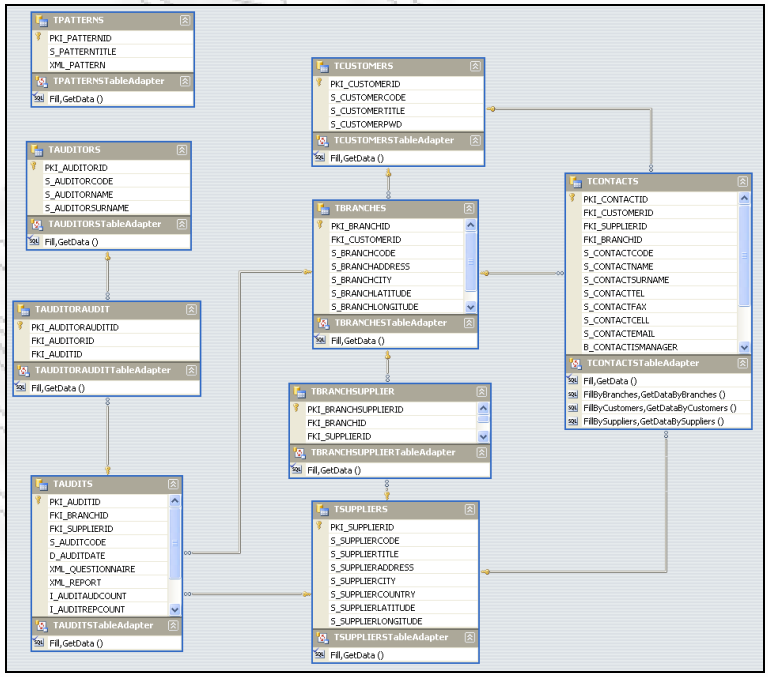
Αναίρεση αλλαγών:

```
if (this.g_dsBiocert.HasChanges())
{
  if ((MessageBox.Show("Πρόκειται να αναιρέσετε τις αλλαγές σας." +
       Environment.NewLine + "Θέλετε να συνεχίσετε;", "Delta Server",
       MessageBoxButton.YesNo, MessageBoxImage.Question) ==
       MessageBoxResult.Yes))
  {
    this.g_dsBiocert.RejectChanges();
    this.g_viewCurrentView.MoveCurrentToFirst();
    this.g_viewCurrentView.Refresh();
  }
}
```

Αποθήκευσης αλλαγών:

```
if (this.g_dsBiocert.HasChanges())
{
  if ((MessageBox.Show("Πρόκειται να αποθηκεύσετε τις αλλαγές σας." +
       Environment.NewLine + "Θέλετε να συνεχίσετε;", "Delta Server",
       MessageBoxButton.YesNo, MessageBoxImage.Question) ==
       MessageBoxResult.Yes))
  {
    this.g_tamAuditors.TAUDITORSTableAdapter.
    Update(this.g_dsBiocert.TAUDITORS);
  }
}
```

Όλες οι αλλαγές (εισαγωγή, διαγραφή, αναίρεση) γίνονται στο Dataset object και όχι στη βάση δεδομένων (Εικόνα 6.1). Με αυτόν τον τρόπο επιτυγχάνεται καλύτερη απόδοση (performance) της εφαρμογής, μιας και η διαχείριση των δεδομένων γίνεται στη μνήμη χωρίς την συνεχή επικοινωνία με τη βάση. Μόνο στην αποθήκευση η εφαρμογή επικοινωνεί με τη βάση δεδομένων αποθηκεύοντας ολόκληρο το Dataset.



Εικόνα 6.1 Το Dataset του Delta Server.

Στα παράθυρα επεξεργασίας των καταστημάτων και των προμηθευτών, υπάρχει ένα επιπλέον κουμπί (Delta Web). Στο πάτημά του ανοίγει, στον περιηγητή (browser), η σελίδα εύρεσης συντεταγμένων. Με αυτό τον τρόπο ο διαχειριστής μπορεί εύκολα και γρήγορα να καταχωρήσει για το κατάστημα ή τον προμηθευτή τις γεωγραφικές συντεταγμένες. Αυτές αποτελούν απαραίτητη πληροφορία για την προβολή των αποτελεσμάτων της επιθεώρησης στο χάρτη (Delta Web, Ενότητα 6.5).

```
System.Diagnostics.Process.Start
("http://psatos.satos.gr/coordinates.html");
```

Στο παράθυρο επεξεργασίας των ερωτηματολογίων καθώς και των επιθεωρήσεων υπάρχει το κουμπί 'Προβολή' που βρίσκεται δίπλα στην ετικέτα 'Ερωτήσεις'. Με το πάτημά του ανοίγει νέο παράθυρο (`public partial class winQuestions : Window`) όπου εμφανίζονται οι τομείς και οι ερωτήσεις του ερωτηματολογίου. Αυτό είναι απαραίτητο γιατί το πλήθος των τομέων και των ερωτήσεων δεν είναι στατικό αλλά δυναμικό. Αρχικά γίνεται ο έλεγχος αν η κλήση έχει γίνει από τα ερωτηματολόγια ή από τις επιθεωρήσεις. Στην πρώτη περίπτωση διαβάζει το XML (ερωτηματολόγιο) από τον πίνακα της βάση που συγκρατεί τα πρότυπα (tPatterns), ενώ στην δεύτερη περίπτωση διαβάζει από τον πίνακα των επιθεωρήσεων (tAudits).

```
l_xml.LoadXml(l_clDecryption.Decrypt(l_dvPattern[0]
      ["XML_PATTERN"].ToString(), winLogin.g_strPath));
l_xml.LoadXml(l_clDecryption.Decrypt(l_dvAudit[0]
      ["XML_QUESTIONNAIRE"].ToString(), winLogin.g_strPath));
```

Στο πάτημα του κουμπιού 'Τομέας' εισάγεται νέος τομέας στο ερωτηματολόγιο. Αυτό επιτυγχάνεται με την κλήση της μεθόδου `private void AddSection(String a_strText, bool a_blnEditMode)`. Η μέθοδος εισάγει σε ένα StackPanel ένα TextBox για την εισαγωγή της περιγραφής του τομέα και δύο κουμπιά. Το πρώτο για τη διαγραφή του τομέα και το δεύτερο για την εισαγωγή ερωτήσεων. Τα ονόματα των κουμπιών αποτελούνται από ένα σταθερό κείμενο, που διακρίνει την κατηγορία του κουμπιού (τομέα, ερώτησης), ακολουθούμενο από το τρέχοντα αριθμό των κουμπιών (αύξων αριθμός κουμπιών). Με αυτόν τον τρόπο επιτυγχάνεται η μοναδικότητα των ονομάτων των κουμπιών, για τον εντοπισμό της ταυτότητας του κουμπιού στο πάτημά του, αφού όλα καλούν την ίδια μέθοδο (`l_btnQuestion_Click` ή `l_btnDeleteSection_Click`).

```
StackPanel l_spSection = new StackPanel();
TextBox l_txtSection = new TextBox();
l_txtSection.Name = "txtSection_" + g_intSectionsCount;
g_btnQuestion = new Button();
g_btnQuestion.Click += new
System.Windows.RoutedEventHandler(l_btnQuestion_Click);
g_btnQuestion.Name = "btnQuestion_" + g_intSectionsCount;
l_btnDeleteSection.Name = "btnDelete_" + g_intSectionsCount;
l_btnDeleteSection.Click += new
System.Windows.RoutedEventHandler(l_btnDeleteSection_Click);
l_spSection.Children.Add(l_txtSection);
l_spSection.Children.Add(l_btnDeleteSection);
l_spSection.Children.Add(g_btnQuestion);
g_stackPanel.Children.Add(l_spSection);
```

Δίπλα σε κάθε τομέα υπάρχει το κουμπί 'Ερώτηση', το οποίο καλεί τη μέθοδο `private void AddQuestion(object sender, String a_strText, bool a_blnEditMode)` για την εισαγωγή ερώτησης στον τομέα. Η μέθοδος εισάγει σε ένα StackPanel ένα TextBox για την εισαγωγή της περιγραφής της ερώτησης και ένα κουμπί για τη διαγραφή της. Το όνομα του κουμπιού αποτελείται από ένα σταθερό κείμενο, που διακρίνει την κατηγορία του κουμπιού (ερώτησης), ακολουθούμενο από το τρέχοντα αριθμό των κουμπιών (αύξων αριθμός κουμπιών). Με αυτόν τον τρόπο

επιτυγχάνεται η μοναδικότητα του ονόματός του, όπως και στην περίπτωση των κουμπιών του τομέα (`l_btnDeleteQuestion_Click`). Από το όνομα του κουμπιού που κάλεσε τη μέθοδο (sender) βρίσκεται το όνομα του τομέα.

```
StackPanel l_spQuestion = new StackPanel();
l_spQuestion.Name = "l_spQuestion";
Button l_btnCurrent = (Button)sender;
TextBox l_txtQuestion = new TextBox();
l_txtQuestion.Name = "txtQuestion_" + l_btnCurrent.Name.
              Split('_').ElementAt(1) + "_" + g_intQuestionsCount;
Button l_btnDeleteQuestion = new Button();
l_btnDeleteQuestion.Name = "btnDelete_" +l_btnCurrent.Name.
              Split('_').ElementAt(1) + "_" + g_intQuestionsCount;
l_btnDeleteQuestion.Click += new
System.Windows.RoutedEventHandler(l_btnDeleteQuestion_Click);
l_spQuestion.Children.Add(l_txtQuestion);
l_spQuestion.Children.Add(l_btnDeleteQuestion);
g_uiElement = (UIElement)l_btnCurrent.Parent;
g_stackPanel.Children.Insert(g_stackPanel.Children.
                                IndexOf(g_uiElement) + 1, l_spQuestion);
```

Και για τη διαγραφή τομέα αλλά και για την διαγραφή της ερώτησης καλείται η ίδια μέθοδος (`private void RemoveSectionStackPanel(DependencyObject parent, int level, Button a_btnSender, bool a_blnIsQuestion)`). Από το τελευταίο όρισμά της (flag) η μέθοδος γνωρίζει αν πρόκειται να διαγράψει τομέα ή ερώτηση.

```
if (!a_blnIsQuestion)
{
  if (l_txtCurrent.Name == "txtSection_" +
      a_btnSender.Name.Split('_').ElementAt(1))
  {
    g_uiElement = (UIElement)l_txtCurrent.Parent;
```

Στη διαγραφή του τομέα, διαγράφονται και όλες οι ερωτήσεις του.

```
    //DELETE ALL QUESTIONS
    while (g_stackPanel.Children.Count > 0)
    {
      if (g_stackPanel.Children.IndexOf(g_uiElement) + 1 ==
          g_stackPanel.Children.Count)
      {
        g_stackPanel.Children.Remove(g_uiElement);
        return;
      }
      else
      {
        if (g_stackPanel.Children[g_stackPanel.Children.
            IndexOf(g_uiElement) + 1].GetType().Name == "StackPanel")
        {
          StackPanel l_sp = new StackPanel();
          l_sp = (StackPanel)g_stackPanel.Children[g_stackPanel.
                  Children.IndexOf(g_uiElement) + 1];
          if (l_sp.Name == "l_spQuestion")
          {
            g_stackPanel.Children.RemoveAt(g_stackPanel.
              Children.IndexOf(g_uiElement) + 1);
          }
          else
          {
            g_stackPanel.Children.Remove(g_uiElement);
            return;
          }
        }
```

```
        }
      }
    }
  }
  else
  {
    if (l_txtCurrent.Name == "txtQuestion_" +
        a_btnSender.Name.Split('_').ElementAt(1) + "_" +
        a_btnSender.Name.Split('_').ElementAt(2))
    {
      g_uiElement = (UIElement)l_txtCurrent.Parent;
      g_stackPanel.Children.Remove(g_uiElement);
    }
  }
```

Με το πάτημα του κουμπιού ‘Αποθήκευση’ δημιουργείται όλο το XML με βάση τα visual controls που απεικονίζουν τους τομείς και τις ερωτήσεις και αποθηκεύεται στη βάση. Σημαντικό είναι το γεγονός ότι το interface είναι σε XAML (Extensible Application Markup Language) η οποία είναι μία XML-based γλώσσα δημιουργίας User Interfaces. Έτσι στο σύστημα η XML αποτελεί μια κοινή γλώσσα επικοινωνίας μεταξύ γραφικού περιβάλλοντος και δομής των δεδομένων.

```
// CREATE XML QUESTIONNAIRE
XmlNode l_xmlDeclaration = g_xml.CreateXmlDeclaration("1.0", "UTF-8",
null);
g_xml.AppendChild(l_xmlDeclaration);
XmlNode l_xmlQUESTIONNAIRE = g_xml.CreateElement("QUESTIONNAIRE");
g_xml.AppendChild(l_xmlQUESTIONNAIRE);
// HEADER
XmlNode l_xmlHEADER = g_xml.CreateElement("HEADER");
l_xmlQUESTIONNAIRE.AppendChild(l_xmlHEADER);
XmlNode l_xmlCODE = g_xml.CreateElement("CODE");
l_xmlCODE.InnerText = g_strAuditCode;
l_xmlHEADER.AppendChild(l_xmlCODE);
XmlNode l_xmlTITLE = g_xml.CreateElement("TITLE");
l_xmlTITLE.InnerText = g_textBlock.Text;
l_xmlHEADER.AppendChild(l_xmlTITLE);
XmlNode l_xmlDATE = g_xml.CreateElement("DATE");
l_xmlHEADER.AppendChild(l_xmlDATE);
XmlNode l_xmlPLACE = g_xml.CreateElement("PLACE");
l_xmlPLACE.InnerText = g_strPlace;
l_xmlHEADER.AppendChild(l_xmlPLACE);
XmlNode l_xmlADDRESS = g_xml.CreateElement("ADDRESS");
l_xmlADDRESS.InnerText = g_strAddress;
l_xmlHEADER.AppendChild(l_xmlADDRESS);
XmlNode l_xmlLATITUDE = g_xml.CreateElement("LATITUDE");
l_xmlLATITUDE.InnerText = g_strLatitude;
l_xmlHEADER.AppendChild(l_xmlLATITUDE);
XmlNode l_xmlLONGITUDE = g_xml.CreateElement("LONGITUDE");
l_xmlLONGITUDE.InnerText = g_strLongitude;
l_xmlHEADER.AppendChild(l_xmlLONGITUDE);
// BODY
XmlNode l_xmlBODY = g_xml.CreateElement("BODY");
l_xmlQUESTIONNAIRE.AppendChild(l_xmlBODY);
// FIND ALL SECTIONS
for (int i = 0; i < VisualTreeHelper.GetChildrenCount(g_stackPanel);
i++)
{
  DependencyObject child = VisualTreeHelper.GetChild(g_stackPanel,
  i);
  StackPanel l_sp = new StackPanel();
```

```
  TextBox l_txtCurrent = new TextBox();
  string typeName = child.GetType().Name;
  if (typeName == "StackPanel")
  {
    l_sp = (StackPanel)child;
    for (int j=0; j<=l_sp.Children.Count - 1; j++)
    {
      typeName = l_sp.Children[j].GetType().Name;
      if (typeName == "TextBox")
      {
        l_txtCurrent = (TextBox)l_sp.Children[j];
        if (l_txtCurrent.Name.Split('_').ElementAt(0) ==
            "txtSection")
        {
          XmlNode l_xmlSECTION = g_xml.CreateElement("SECTION");
          l_xmlBODY.AppendChild(l_xmlSECTION);
          XmlNode l_xmlNAME = g_xml.CreateElement("NAME");
          l_xmlNAME.InnerText = l_txtCurrent.Text;
          l_xmlSECTION.AppendChild(l_xmlNAME);
          XmlNode l_xmlFLAG = g_xml.CreateElement("FLAG");
          l_xmlSECTION.AppendChild(l_xmlFLAG);
          XmlNode l_xmlNOTES = g_xml.CreateElement("NOTES");
          l_xmlSECTION.AppendChild(l_xmlNOTES);
          this.AddQuestionsToSection(l_txtCurrent.
          Name.Split('_').ElementAt(1), g_xml, l_xmlSECTION);
        }
      }
    }
  }
}
// SINGATURES
XmlNode l_xmlSINGATURES = g_xml.CreateElement("SINGATURES");
l_xmlQUESTIONNAIRE.AppendChild(l_xmlSINGATURES);
XmlNode l_xmlAUDITOR = g_xml.CreateElement("AUDITOR");
l_xmlSINGATURES.AppendChild(l_xmlAUDITOR);
XmlNode l_xmlCUSTOMER = g_xml.CreateElement("CUSTOMER");
l_xmlSINGATURES.AppendChild(l_xmlCUSTOMER);
// SAVE TO DB
if (g_blnIsPattern)
{
  for (int i = 0; i <= g_dsBiocert.TPATTERNS.Rows.Count - 1; i++)
  {
    String l_strTitle =
          (String)g_dsBiocert.TPATTERNS[i]["S_PATTERNTITLE"];
    if (l_strTitle == g_textBlock.Text)
    {
      clEncryption l_clEncryption = new clEncryption();
      g_dsBiocert.TPATTERNS.Rows[i]["XML_PATTERN"] =
      l_clEncryption.Encrypt(g_xml.InnerXml,winLogin.g_strPath); ;
      this.g_tamPatterns.TPATTERNSTableAdapter.
      Update(this.g_dsBiocert.TPATTERNS);
    }
  }
}
else
{
  using(DataView l_dvAudit = new DataView(this.g_dsBiocert.TAUDITS))
  {
    l_dvAudit.RowFilter = "S_AUDITCODE = '" + g_strAuditCode + "'";
    clEncryption l_clEncryption = new clEncryption();
```

```
        l_dvAudit[0]["XML_QUESTIONNAIRE"] =
        l_clEncryption.Encrypt(g_xml.InnerXml, winLogin.g_strPath);
        this.g_tamAudits.TAUDITSTableAdapter.Update(l_dvAudit[0].Row);
    }
}
```

Για τον κάθε τομέα που εισάγεται καλείται η μέθοδος `private void AddQuestionsToSection(String a_strSP, XmlDocument a_xml, XmlNode a_xmlSection)` η οποία εισάγει τις ερωτήσεις του τομέα. Η μέθοδος ελέγχει αν ο τομέας έχει ερωτήσεις. Στην περίπτωση των υπερτομέων δεν κάνει κάποια ενέργεια, αφού οι υπερτομείς δεν περιέχουν ερωτήσεις αλλά τομείς.

```
for (int j = 0; j <= l_sp.Children.Count - 1; j++)
{
  typeName = l_sp.Children[j].GetType().Name;
  if (typeName == "TextBox")
  {
    l_txtCurrent = (TextBox)l_sp.Children[j];
    if (l_txtCurrent.Name.Split('_').ElementAt(0) == "txtQuestion")
    {
      if (l_txtCurrent.Name.Split('_').ElementAt(1) == a_strSP)
      {
        XmlNode l_xmlQUESTION = a_xml.CreateElement("QUESTION");
        a_xmlSection.AppendChild(l_xmlQUESTION);
        XmlNode l_xmlCAPTION = a_xml.CreateElement("CAPTION");
        l_xmlCAPTION.InnerText = l_txtCurrent.Text;
        l_xmlQUESTION.AppendChild(l_xmlCAPTION);
        XmlNode l_xmlANSWER = a_xml.CreateElement("ANSWER");
        l_xmlQUESTION.AppendChild(l_xmlANSWER);
      }
    }
  }
}
```

Επίσης, στο παράθυρο επεξεργασίας των επιθεωρήσεων υπάρχει και το κουμπί ‘Προβολή’ που βρίσκεται δίπλα στην ετικέτα ‘Επιθεωρητές’. Με το πάτημά του ανοίγει το παράθυρο (`public partial class winSelectAuditors : Window`) που εμφανίζει όλους τους επιθεωρητές. Η ύπαρξη του παραθύρου είναι αναγκαία γιατί, όπως και στην περίπτωση των ερωτήσεων, ο αριθμός των επιθεωρητών που συμμετέχουν σε μία επιθεώρηση δεν είναι στατικός (σταθερός) αλλά δυναμικός. Έτσι με την προβολή όλων των επιθεωρητών ο διαχειριστής μπορεί να επιλέξει από έναν μέχρι και όλους. Στη δημιουργία του παραθύρου εκχωρούνται όλοι οι επιθεωρητές που έχουν επιλεγεί για την επιθεώρηση σε μία ArrayList.

```
for (int i = 0; i <= this.g_dsBiocert.TAUDITORAUDIT.Count - 1; i++)
{
  if (int.Parse(this.g_dsBiocert.TAUDITORAUDIT[i]
      ["FKI_AUDITID"].ToString()) == g_intAuditID)
  {
    g_intAuditorsID.Add(this.g_dsBiocert.
                        TAUDITORAUDIT[i]["FKI_AUDITORID"]);
  }
}
```

Κάθε CheckBox, που βρίσκεται μπροστά από το όνομα του κάθε επιθεωρητή, επιλέγεται (status = checked) στην περίπτωση που ο επιθεωρητής είχε επιλεγεί για την επιθεώρηση.

```
for (int i = 0; i <= this.g_dsBiocert.TAUDITORS.Count-1; i++)
{
  CheckBox l_chkAuditors = new CheckBox();
  l_chkAuditors.Content = " " +  this.g_dsBiocert.TAUDITORS[i]
```

```
  ["S_AUDITORSURNAME"].ToString() + " " +
  this.g_dsBiocert.TAUDITORS[i]["S_AUDITORNAME"].ToString() + "/" +
  this.g_dsBiocert.TAUDITORS[i]["S_AUDITORCODE"].ToString();
  // IF AUDITOR HAD SELECTED THEN CHECKED
  l_chkAuditors.IsChecked = this.CheckAuditors(l_chkAuditors);
  l_chkAuditors.Checked += new
                RoutedEventHandler(l_chkAuditors_Checked);
  l_chkAuditors.Unchecked += new
                RoutedEventHandler(l_chkAuditors_Unchecked);
  l_chkAuditors.Margin = new Thickness(10, 5, 0, 0);
  g_stackPanel.Children.Add(l_chkAuditors);
}
```

Ο έλεγχος έγινε μέσω της μεθόδου `private bool CheckAuditors(CheckBox a_chkBox)`. Η μέθοδος ελέγχει αν ο επιθεωρητής πρέπει να είναι επιλεγμένος και εισάγει την τιμή True ή False στο CheckBox.

```
using (DataView l_dvAuditors = new
       DataView(this.g_dsBiocert.TAUDITORS))
{
  l_dvAuditors.RowFilter = "S_AUDITORCODE = '" +
  a_chkBox.Content.ToString().Split('/').ElementAt(1) + "'";
  using (DataView l_dvAuditorAudit = new
         DataView(this.g_dsBiocert.TAUDITORAUDIT))
  {
    l_dvAuditorAudit.RowFilter = "FKI_AUDITORID = " +
    l_dvAuditors[0]["PKI_AUDITORID"] + " and FKI_AUDITID = " +
    g_intAuditID;
    if (l_dvAuditorAudit.Count > 0)
    {
      return true;
    }
    else
    {
      return false;
    }
  }
}
```

Στο πάτημα του κάθε CheckBox η τιμή του κλειδιού του επιθεωρητή εισάγεται ή αφαιρείται αντίστοιχα από την ArrayList που δημιουργήθηκε στην αρχή.

```
g_intAuditorsID.Add(l_dvAuditors[0]["PKI_AUDITORID"]);
g_intAuditorsID.Remove(l_dvAuditors[0]["PKI_AUDITORID"]);
```

Με το πάτημα του κουμπιού 'Αποθήκευση' αποθηκεύονται οι αλλαγές στη βάση.

```
// SAVE TO DB
for (int i = 0; i <= g_intAuditorsID.Count - 1; i++)
{
  var l_drAuditorAudit = this.g_dsBiocert.TAUDITORAUDIT.
                         NewTAUDITORAUDITRow();
  l_drAuditorAudit["FKI_AUDITORID"] = g_intAuditorsID[i];
  l_drAuditorAudit["FKI_AUDITID"] = g_intAuditID;
  this.g_dsBiocert.TAUDITORAUDIT.
  AddTAUDITORAUDITRow(l_drAuditorAudit);
}
this.g_tamAuditorAudit.TAUDITORAUDITTableAdapter.Update(this.g_dsBioc
ert.TAUDITORAUDIT);
```

Μετά την αποθήκευση των αλλαγών στη βάση, ενημερώνεται και το πεδίο του πίνακα που συγκρατεί το πλήθος των επιθεωρητών της επιθεώρησης. Το πεδίο αυτό χρησιμοποιείται για τη σύγκριση του αριθμού των αναφορών της επιθεώρησης με το

πλήθος των επιθεωρητών ώστε να γνωρίζει το σύστημα αν μία επιθεώρηση έχει ολοκληρωθεί, όπως περιγράφεται στη συνέχεια.

```
//Update Audit Auditors Count
using(dsBiocertTableAdapters.TAUDITSTableAdapter l_taAudits = new
dsBiocertTableAdapters.TAUDITSTableAdapter())
{
  l_taAudits.Fill(g_dsBiocert.TAUDITS);
  using (DataView l_dvAudits = new
         DataView(this.g_dsBiocert.TAUDITS))
  {
    l_dvAudits.RowFilter = "PKI_AUDITID = " + g_intAuditID;
    if (l_dvAudits.Count == 1)
    {
      l_dvAudits[0]["I_AUDITAUDCOUNT"] = g_intAuditorsID.Count;
      l_taAudits.Update(l_dvAudits[0].Row);
    }
  }
}
```

Το κουμπί ‘Αναφορές’ προβάλει στο διαχειριστή τις αναφορές που έχουν αποστείλει οι επιθεωρητές. Όπως έχει αναφερθεί, μία επιθεώρηση για να θεωρηθεί ολοκληρωμένη θα πρέπει ο αριθμός των αναφορών να ισούται με το πλήθος των επιθεωρητών. Αν ο αριθμός των αναφορών είναι μικρότερος σημαίνει ότι εκκρεμούν μία ή περισσότερες αναφορές.

```
int l_intFilesCount = 0;
ArrayList l_strFiles = new ArrayList();
using (System.Data.DataView l_dvAudit = new
System.Data.DataView(g_dsBiocert.TAUDITS))
{
  for (int i=0; i<=l_dvAudit.Count-1; i++)
  {
    if (l_dvAudit[i]["I_AUDITREPCOUNT"].ToString() != "")
    {
      if(int.Parse(l_dvAudit[i]["I_AUDITAUDCOUNT"].ToString()) > 0)
      {
        if (int.Parse(l_dvAudit[i]["I_AUDITAUDCOUNT"].ToString()) ==
            int.Parse(l_dvAudit[i]["I_AUDITREPCOUNT"].ToString()))
        {
          l_intFilesCount += 1;
```

Ο κωδικός της κάθε ολοκληρωμένης επιθεώρησης εισάγεται σε μία ArrayList.

```
          l_strFiles.Add(l_dvAudit[i]["S_AUDITCODE"].ToString());
        }
      }
    }
  }
}
```

Αν υπάρχουν ολοκληρωμένες επιθεωρήσεις ανοίγει το παράθυρο winSelectFiles, το οποίο δίνει τη δυνατότητα στο διαχειριστή, επιλέγοντας τον κωδικό της επιθεώρησης, να προβάλει την αναφορά της.

```
if (l_intFilesCount == 0)
{
  MessageBox.Show("Δεν υπάρχουν διαθέσιμες αναφορές!", "Delta
                  Server", MessageBoxButton.OK,
                  MessageBoxImage.Information, MessageBoxResult.OK);
}
else
{
```

```
winSelectFiles l_winSelectFiles = new winSelectFiles(this);
l_winSelectFiles.g_strFiles = l_strFiles;
l_winSelectFiles.ShowDialog();
```

Για τους κωδικούς επιθεωρήσεων που συγκρατεί η ArrayList, δημιουργείται ένα RadioButton για κάθε επιθεώρηση.

```
for (int i = 0; i <= g_strFiles.Count - 1; i++)
{
  RadioButton l_rbFile = new RadioButton();
  l_rbFile.FontSize = 14;
  l_rbFile.Content = " " + g_strFiles[i].ToString();
  l_rbFile.Tag = g_strFiles[i];
  l_rbFile.Checked += new RoutedEventHandler(l_rbFile_Checked);
  l_rbFile.Margin = new Thickness(10, 5, 0, 5);
  g_stackPanel.Children.Add(l_rbFile);
}
```

Αν ο διαχειριστής πατήσει το κουμπί 'OK' του παραθύρου, δημιουργείται ένα UserControl τύπου ucAudit, το οποίο περιέχει την αναφορά για την επιθεώρηση που έχει επιλεγεί (`g_winMain.g_strFileName = g_strFileName;`).

```
if (g_blnIsOK)
{
  foreach (TabItem item in this.tabWindows.Items)
  {
    if (item.Name == "tabAuditReport")
    {
      item.Focus();
      return;
    }
  }
  ucAudit l_ucAudit = new ucAudit();
  l_ucAudit.g_strFileName = g_strFileName;
  tiAudits.Content = l_ucAudit;
  this.tabWindows.Items.Add(tiAudits);
  this.tabWindows.SelectedItem = tiAudits;
}
}
```

Στη δημιουργία του control της αναφοράς (ucAudit), διαβάζεται το XML της αναφοράς και μετατρέπεται σε visual μορφή.

```
l_lblTitle.Content = "Αναφορά Επιθεώρησης";
l_txtSubtitle.Text = "Για την αναφορά προβάλλονται οι σημαίες των
τομέων και οι απαντήσεις των ερωτήσεών τους. Επίσης μπορείτε να δείτε
τις παρατηρήσεις για τον κάθε τομέα. Πατώντας το κουμπί 'Εξαγωγή
Αναφοράς' μπορείτε να εξάγετε την αναφορά σε μορφή Word (<ΚΩΔΙΚΟΣ
ΕΠΙΘΕΩΡΗΣΗΣ>.doc) στη διαδρομή 'C:\\Delta Reports'.";
//CREATE QUESTIONNAIRE
XmlDocument l_xml = new XmlDocument();
string l_str = "";
using (System.Data.DataView l_dvAudit = new
System.Data.DataView(g_dsBiocert.TAUDITS))
{
  l_dvAudit.RowFilter = "S_AUDITCODE = '" + g_strFileName + "'";
  l_str = l_dvAudit[0]["XML_REPORT"].ToString();
}
l_xml.LoadXml(l_clDecryption.Decrypt(l_str, winLogin.g_strPath));
// TITLE
l_txtTitle.Text = l_xml.GetElementsByTagName("TITLE")[0].InnerText;
// DATE
l_txtDate.Text = l_xml.GetElementsByTagName("DATE")[0].InnerText;
// PLACE
```

```
l_txtPlace.Text = l_xml.GetElementsByTagName("PLACE")[0].InnerText;
// ADDRESS
l_txtAddress.Text =
l_xml.GetElementsByTagName("ADDRESS")[0].InnerText;
XmlNodeList l_xmlNodeListSections =
l_xml.GetElementsByTagName("SECTION");
foreach (XmlNode l_xmlNodeSection in l_xmlNodeListSections)
{
  XmlElement l_xmlElementSection = (XmlElement)l_xmlNodeSection;
  XmlNodeList l_xmlNodeListQuestions =
  l_xmlElementSection.GetElementsByTagName("QUESTION");
```

Το Dataset dsReport (Εικόνα 6.2) περιέχει έναν ψεύτικο (dummy) πίνακα στον οποίο εισάγονται η περιγραφή κάθε τομέα και η σημαία του. Τις τιμές αυτές θα τις διαβάσει το αντικείμενο τύπου Crystal Report για να εξάγει την τελική αναφορά σε μορφή doc, όπως θα αναλυθεί στη συνέχεια.

```
  g_dsReport.tSections.AddtSectionsRow(l_xmlElementSection.
  GetElementsByTagName("NAME")[0].InnerText,
  l_xmlElementSection.GetElementsByTagName("FLAG")[0].InnerText,
  "Παρατηρήσεις Τομέα");
```

Αν ο τομέας έχει ερωτήσεις, τότε μετά την εισαγωγή του, εισάγονται οι ερωτήσεις του, οι απαντήσεις του και οι σημειώσεις του.

```
  if (l_xmlNodeListQuestions.Count > 0)
  {
    this.AddSection(l_xmlElementSection.
    GetElementsByTagName("NAME")[0].InnerText, true, true,
    l_xmlElementSection.GetElementsByTagName("FLAG")[0].InnerText);
    foreach (XmlNode l_xmlNodeQuestion in l_xmlNodeListQuestions)
    {
      XmlElement l_xmlElementQuestion =
      (XmlElement)l_xmlNodeQuestion;
      this.AddQuestion(g_btnQuestion,
      l_xmlElementQuestion.GetElementsByTagName
      ("CAPTION")[0].InnerText, true, ref l_spQuestions,
      l_xmlElementQuestion.GetElementsByTagName
      ("ANSWER")[0].InnerText);
    }
    g_uiElement = (UIElement)g_btnQuestion.Parent;
    l_spQuestionsNotes.Children.Add(l_spQuestions);
    g_stackPanel.Children.Insert(g_stackPanel.Children.
    IndexOf(g_uiElement) + 1, l_spQuestionsNotes);
    this.AddNotes(g_btnQuestion, null, false, ref l_spQuestionsNotes,
    l_xmlNodeListQuestions.Count, l_xmlElementSection.
    GetElementsByTagName("NOTES")[0].InnerText);
  }
```

Αν ο τομέας δεν έχει ερωτήσεις, είναι υπερτομέας και εισάγεται χωρίς σημειώσεις.

```
  else
  {
    this.AddSection(l_xmlElementSection.
    GetElementsByTagName("NAME")[0].InnerText, true,false,"");
  }
}
```

Τέλος εισάγονται οι υπογραφές του επιθεωρητή και του πελάτη.

```
this.AddSignatures(l_xml.GetElementsByTagName("AUDITOR")[0].InnerText
, l_xml.GetElementsByTagName("CUSTOMER")[0].InnerText);
```

Η μέθοδος που εισάγει τους τομείς (`private void AddSection(String a_strText, bool a_blnEditMode, bool a_blnHasQuestions, String`

a_strFlag)) διαβάζει την περιγραφή του τομέα και εισάγει την τιμή της σημαίας του, αν ο τομέας έχει ερωτήσεις (δεν είναι υπερτομέας).

```
StackPanel l_spSection = new StackPanel();
if (a_blnEditMode)
{
  l_txtSection.Text = a_strText;
}
else
{
  l_txtSection.Text = a_strText + g_intSectionsCount;
}
l_spSection.Children.Add(l_txtSection);
if (a_blnHasQuestions)
{
  l_lblSectionFlag.Content = "Σημαία:";
  l_cmbSection.Text = a_strFlag;
  l_spSection.Children.Add(l_lblSectionFlag);
  l_spSection.Children.Add(l_cmbSection);
}
```

Η μέθοδος που εισάγει τις ερωτήσεις και τις απαντήσεις του τομέα (`private void AddQuestion(object sender, String a_strText, bool a_blnEditMode, ref StackPanel a_spQuestions, string a_strAnswer)`) διαβάζει την ερώτηση και εισάγει την τιμή της απάντησης, δίπλα σε κάθε ερώτηση.

```
StackPanel l_spQuestion = new StackPanel();
if (a_blnEditMode)
{
  l_txtQuestion.Text = a_strText;
}
else
{
  l_txtQuestion.Text = a_strText + g_intQuestionsCount;
}
l_spQuestion.Children.Add(l_txtQuestion);
l_cmbQuestion.Text = a_strAnswer;
l_spQuestion.Children.Add(l_cmbQuestion);
a_spQuestions.Children.Add(l_spQuestion);
```

Η μέθοδος που εισάγει τις παρατηρήσεις του τομέα (`private void AddNotes(object sender, String a_strText, bool a_blnEditMode, ref StackPanel a_spQuestionsNotes, int a_intQuestionsCount, string a_strNotes)`) μετατρέπει την Base64 encoding τιμή του XML σε εικόνα και την εμφανίζει δίπλα στις ερωτήσεις του τομέα. Το ύψος τις εικόνας είναι δυναμικό και ισούται με το ύψος όλων των ερωτήσεων. Η μετατροπή γίνεται μέσω της μεθόδου `public Image Base64ToImage(Image a_image, string a_strBase64String)`.

```
Border l_border = new Border();
l_border.Height = a_intQuestionsCount * 25;
l_border.Child = this.Base64ToImage(l_image, a_strNotes);
a_spQuestionsNotes.Children.Add(l_border);
```

Η μέθοδος Base64ToImage δέχεται ως ορίσματα μία εικόνα και το encrypted κείμενο. Η επιστροφή της είναι η εικόνα που δέχτηκε ως όρισμα, στην οποία έχει εκχωρηθεί το decrypted κείμενο.

```
byte[] l_imageBytes = Convert.FromBase64String(a_strBase64String);
using (MemoryStream l_ms = new MemoryStream(l_imageBytes, 0,
l_imageBytes.Length))
{
  a_image.Source = BitmapFrame.Create(l_ms, BitmapCreateOptions.None,
  BitmapCacheOption.OnLoad);
}
```

```
return a_image;
```
Η μέθοδος που εισάγει τις υπογραφές του επιθεωρητή και του πελάτη της επιθεώρησης (`private void AddSignatures(string a_strAuditorSign, string a_strCustomerSign)`) δέχεται ως ορίσματα τα encrypted κείμενα από το XML που συγκρατούν τις υπογραφές τους. Μέσω της προηγούμενης μεθόδου (Base64ToImage) εισάγει στο τέλος της αναφοράς τις εικόνες των υπογραφών.

```
Border l_border = new Border();
Image l_image1 = new Image();
l_image1.Name = "signAuditor";
l_border.Child = this.Base64ToImage(l_image1, a_strAuditorSign);
Border l_border2 = new Border();
Image l_image2 = new Image();
l_image2.Name = "signCustomer";
l_border2.Child = this.Base64ToImage(l_image2, a_strCustomerSign);
StackPanel l_labels = new StackPanel();
l_labels.Orientation = Orientation.Horizontal;
l_labels.Children.Add(l_lblAuditorSign);
l_labels.Children.Add(l_lblCustomerSign);
StackPanel l_canvas = new StackPanel();
l_canvas.Orientation = Orientation.Horizontal;
l_canvas.Children.Add(l_border);
l_canvas.Children.Add(l_border2);
g_stackPanel.Children.Add(l_labels);
g_stackPanel.Children.Add(l_canvas);
```

Στο πάνω μέρος του control της αναφοράς υπάρχει το κουμπί ‘Εξαγωγή Αναφοράς’. Με το πάτημά του η εφαρμογή εξάγει σε συγκεκριμένη τοποθεσία στον υπολογιστή του διαχειριστή (C:\Delta Reports\) την τελική αναφορά της επιθεώρησης (μορφή doc), βάσει των αποτελεσμάτων της αναφοράς της επιθεώρησης από τον επιθεωρητή.

```
if ((MessageBox.Show("Πρόκειται να εξάγετε την αναφορά." +
Environment.NewLine + "Θέλετε να συνεχίσετε;", "Delta Server",
MessageBoxButton.YesNo, MessageBoxImage.Question) ==
MessageBoxResult.Yes))
{
  this.Cursor = Cursors.Wait;
  // EXPORT DOC
  string l_path = "C:\\Delta Reports";
  string l_pathFile = "";
```

Αν η αναφορά υπάρχει την ανοίγει. Μόνο στην περίπτωση που δεν υπάρχει τη δημιουργεί.

```
  if (!Directory.Exists(l_path))
  {
    DirectoryInfo l_di = Directory.CreateDirectory(l_path);
  }
  l_pathFile = l_path + "\\" + g_strFileName + ".doc";
  if (!File.Exists(l_pathFile))
  {
```

Δημιουργεί ένα αντικείμενο τύπου AuditReport (Crystal Report object) και εκχωρεί ως DataSource το Dataset dsReport (Εικόνα 6.2). Ο πίνακας tSections του Dataset περιέχει τόσες εγγραφές όσοι είναι και οι τομείς της αναφοράς. Οι εγγραφές εκχωρήθηκαν μετά την εισαγωγή κάθε τομέα, όπως αναλύθηκε πιο πάνω.

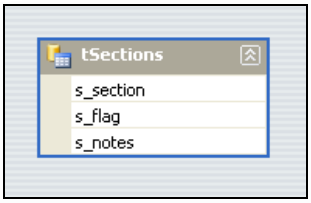
```
    AuditReport l_repAuditReport = new AuditReport();
    l_repAuditReport.SetDataSource(g_dsReport);
    ExportOptions CrExportOptions;
    DiskFileDestinationOptions CrDiskFileDestinationOptions = new
    DiskFileDestinationOptions();
```

```
PdfRtfWordFormatOptions CrFormatTypeOptions = new
PdfRtfWordFormatOptions();
CrDiskFileDestinationOptions.DiskFileName = l_pathFile;
CrExportOptions = l_repAuditReport.ExportOptions;
{
  CrExportOptions.ExportDestinationType =
  ExportDestinationType.DiskFile;
  CrExportOptions.ExportFormatType =
  ExportFormatType.WordForWindows;
  CrExportOptions.DestinationOptions =
  CrDiskFileDestinationOptions;
  CrExportOptions.FormatOptions = CrFormatTypeOptions;
}
```

Το εκτυπωτικό έχει πέντε παραμέτρους στις οποίες εκχωρούνται ο κωδικός της επιθεώρησης, ο τίτλος, η ημερομηνία, το κατάστημα και η διεύθυνση του καταστήματος.

```
// ADD CODE
CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinitions
crParameterFieldDefinitions1;
CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinition
crParameterFieldDefinition1;
ParameterValues crParameterValues1 = new ParameterValues();
ParameterDiscreteValue crParameterDiscreteValue1 = new
ParameterDiscreteValue();
crParameterDiscreteValue1.Value = g_strFileName;
crParameterFieldDefinitions1 =
l_repAuditReport.DataDefinition.ParameterFields;
crParameterFieldDefinition1 =
crParameterFieldDefinitions1["code"];
crParameterValues1 = crParameterFieldDefinition1.CurrentValues;
crParameterValues1.Clear();
crParameterValues1.Add(crParameterDiscreteValue1);
crParameterFieldDefinition1.ApplyCurrentValues(
crParameterValues1);
// ADD TITLE
CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinitions
crParameterFieldDefinitions2;
CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinition
crParameterFieldDefinition2;
ParameterValues crParameterValues2 = new ParameterValues();
ParameterDiscreteValue crParameterDiscreteValue2 = new
ParameterDiscreteValue();
crParameterDiscreteValue2.Value = g_strTitle;
crParameterFieldDefinitions2 =
l_repAuditReport.DataDefinition.ParameterFields;
crParameterFieldDefinition2 =
crParameterFieldDefinitions2["title"];
crParameterValues2 = crParameterFieldDefinition2.CurrentValues;
crParameterValues2.Clear();
crParameterValues2.Add(crParameterDiscreteValue2);
crParameterFieldDefinition2.ApplyCurrentValues(
crParameterValues2);
// ADD DATE
CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinitions
crParameterFieldDefinitions3;
CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinition
crParameterFieldDefinition3;
ParameterValues crParameterValues3 = new ParameterValues();
ParameterDiscreteValue crParameterDiscreteValue3 = new
```

```
        ParameterDiscreteValue();
        crParameterDiscreteValue3.Value = g_strDate;
        crParameterFieldDefinitions3 =
        l_repAuditReport.DataDefinition.ParameterFields;
        crParameterFieldDefinition3 =
        crParameterFieldDefinitions3["date"];
        crParameterValues3 = crParameterFieldDefinition3.CurrentValues;
        crParameterValues3.Clear();
        crParameterValues3.Add(crParameterDiscreteValue3);
        crParameterFieldDefinition3.ApplyCurrentValues(
        crParameterValues3);
        // ADD PLACE
        CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinitions
        crParameterFieldDefinitions4;
        CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinition
        crParameterFieldDefinition4;
        ParameterValues crParameterValues4 = new ParameterValues();
        ParameterDiscreteValue crParameterDiscreteValue4 = new
        ParameterDiscreteValue();
        crParameterDiscreteValue4.Value = g_strPlace;
        crParameterFieldDefinitions4 =
        l_repAuditReport.DataDefinition.ParameterFields;
        crParameterFieldDefinition4 =
        crParameterFieldDefinitions4["place"];
        crParameterValues4 = crParameterFieldDefinition4.CurrentValues;
        crParameterValues4.Clear();
        crParameterValues4.Add(crParameterDiscreteValue4);
        crParameterFieldDefinition4.ApplyCurrentValues(
        crParameterValues4);
        // ADD ADDRESS
        CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinitions
        crParameterFieldDefinitions5;
        CrystalDecisions.CrystalReports.Engine.ParameterFieldDefinition
        crParameterFieldDefinition5;
        ParameterValues crParameterValues5 = new ParameterValues();
        ParameterDiscreteValue crParameterDiscreteValue5 = new
        ParameterDiscreteValue();
        crParameterDiscreteValue5.Value = g_strAddress;
        crParameterFieldDefinitions5 =
        l_repAuditReport.DataDefinition.ParameterFields;
        crParameterFieldDefinition5 =
        crParameterFieldDefinitions5["address"];
        crParameterValues5 = crParameterFieldDefinition5.CurrentValues;
        crParameterValues5.Clear();
        crParameterValues5.Add(crParameterDiscreteValue5);
        crParameterFieldDefinition5.ApplyCurrentValues(
        crParameterValues5);
        // EXPORT
        l_repAuditReport.Export();
      }
      System.Diagnostics.Process.Start(l_pathFile);
      // END EXPORT
    }
```



Εικόνα 6.2 Το Dataset που χρησιμοποιεί
το εκτυπωτικό (Crystal Report).

Με το πάτημα του κουμπιού 'Έξοδος' κλείνει η εφαρμογή `this.Close();`
Με το κουμπί 'Περί…' προβάλλεται το παράθυρο `public partial class winAbout : Window` στο οποίο αναφέρεται η τρέχουσα έκδοση της εφαρμογής. Η έκδοση διαβάζεται από τη web τοποθεσία που έγινε deployment η (ClickOnce) εφαρμογή. Στην περίπτωση που δεν υπάρχει σύνδεση διαδικτύου, διαβάζεται από τις πληροφορίες που περιέχονται στο εκτελέσιμο αρχείο (exe) και οι οποίες ορίστηκαν κατά την ανάπτυξη (deployment) της εφαρμογής. Η δεύτερη περίπτωση βρίσκει εφαρμογή στο Delta Client (Ενότητα 6.3).

```
if(System.Deployment.Application.ApplicationDeployment.
   IsNetworkDeployed)
{
      System.Deployment.Application.ApplicationDeployment
      l_ad = System.Deployment.Application.
             ApplicationDeployment.CurrentDeployment;
      this.lblSubtitle.Text = this.lblSubtitle.Text +
                              l_ad.CurrentVersion.ToString();
}
else
{
      System.Reflection.Assembly thisAssembly =
             this.GetType().Assembly;
      this.lblSubtitle.Text = this.lblSubtitle.Text +
                              thisAssembly.GetName().Version;
}
```

Το κουμπί 'Καρτέλες' κλείνει τις ανοικτές καρτέλες που υπάρχουν στο παράθυρο, εφ' όσον υπάρχουν

```
if (this.tabWindows.Items.Count == 0)
{
      MessageBox.Show("Δεν υπάρχουν ανοικτές καρτέλες.", "Delta
      Client", MessageBoxButton.OK,
      MessageBoxImage.Information, MessageBoxResult.OK);
}
else
{
      if ((MessageBox.Show("Πρόκειται να κλείσετε τις ανοικτές
      καρτέλες." + Environment.NewLine + "Θέλετε να
      συνεχίσετε;", "Delta Client", MessageBoxButton.YesNo,
      MessageBoxImage.Question) == MessageBoxResult.Yes))
      {
            for (int i = this.tabWindows.Items.Count - 1;
            i >= 0; i--)
            {
                  this.tabWindows.Items.RemoveAt(i);
            }
      }
}
```

Τέλος, στο αρχείο app.config ορίζονται οι ρυθμίσεις λειτουργία της εφαρμογής. Με την εγκατάστασή της στον υπολογιστή του επιθεωρητή, μέσω του αρχείου μπορούμε να αλλάξουμε τη διεύθυνση, μέσω της οποίας επιτυγχάνεται η επικοινωνία με τη βάση δεδομένων.

```
<connectionStrings>
      <add
      name="DeltaServer.Properties.Settings.ConnectionString"
      connectionString="Data
      Source=79.166.212.169:1521/XE;Persist Security
      Info=True;User ID=biocert;Password=db!@#;Unicode=True"
      providerName="System.Data.OracleClient" />
</connectionStrings>
```

Αυτό είναι πολύ χρήσιμο στην περίπτωση που αλλάξει η διεύθυνση του Database Server. Σε διαφορετική περίπτωση θα έπρεπε να δημιουργηθεί καινούρια έκδοση της εφαρμογής με ενσωματωμένη την αλλαγή.

6.3 Delta Client

Το project περιλαμβάνει την εφαρμογή που χρησιμοποιούν οι επιθεωρητές, μέσω της οποίας λαμβάνουν τα ερωτηματολόγια και αποστέλλουν τις αναφορές μετά το πέρας της επιθεώρησης. Είναι γραμμένο σε C# σε .NET Framework 3.5 SP1 και έχει αναπτυχθεί με χρήση του Visual Studio 2008 SP1 IDE. Στα βασικά μέρη της δομής του περιλαμβάνονται classes, windows και controls.

Οι δύο κλάσεις του project είναι βοηθητικές. Η πρώτη είναι η `class Utilities : IDisposable` και η δεύτερη η `public class CloseableTabItem : TabItem`
Και οι δύο είναι οι ίδιες με αυτές που αναλύονται στην ενότητα 6.2 του Delta Server.

Η εφαρμογή περιλαμβάνει τέσσερα παράθυρα (windows). Τα winLogin, winMain, winAbout και winSelectFiles.

Το πρώτο παράθυρο εμφανίζεται κατά την εκκίνηση της εφαρμογής. Είναι το παράθυρο της σύνδεσης `public partial class winLogin : Window`. Η σύνδεση είναι υποχρεωτική ώστε να γνωρίζει το σύστημα ποια ερωτηματολόγια να στείλει, κατά την κλήση του αντίστοιχου Service από τον επιθεωρητή. Αρχικά περιέχει δύο static μεταβλητές οι οποίες συγκρατούν τη διαδρομή αποθήκευσης των ερωτηματολογίων και την αντίστοιχη των αναφορών.

```
public static string g_strPathQuestionnaires =
      Environment.GetFolderPath(Environment.SpecialFolder.Commo
      nApplicationData) + "\\Delta Client\\Questionnaires";
public static string g_strPathReports =
      Environment.GetFolderPath(Environment.SpecialFolder.Commo
      nApplicationData) + "\\Delta Client\\Reports";
```

Αρχικά, από το event `private void Window_Loaded(object sender, RoutedEventArgs e)`, ελέγχει αν οι παραπάνω φάκελοι υπάρχουν αλλιώς τους δημιουργεί

```
if (!Directory.Exists(g_strPathQuestionnaires))
{
      Directory.CreateDirectory(g_strPathQuestionnaires);
}
if (!Directory.Exists(g_strPathReports))
```

```
{
        Directory.CreateDirectory(g_strPathReports);
}
```

Το παράθυρο περιλαμβάνει δύο πεδία κειμένου, για την εισαγωγή του ονόματος του χρήστη και του κωδικού και δεκαπέντε κουμπιά. Επειδή η εφαρμογή «τρέχει» σε υπολογιστή με οθόνη αφής (touch screen), περιλαμβάνονται όλα τα κουμπιά που χρειάζονται για τη εισαγωγή του ονόματος χρήστη και του κωδικού, χωρίς τη χρήση πληκτρολογίου. Συγκεκριμένα είναι οι αριθμοί από μηδέν έως και εννιά (0…9), η παύλα (-) και το αγγλικό γράμμα A. Επίσης, υπάρχει το κουμπί C για το σβήσιμο, όπως το backspace στο πληκτρολόγιο.

Στο πάτημα των πεδίων κειμένου από τον χρήστη, εισάγεται σε μια global μεταβλητή το όνομα του πεδίου που έχει ενεργοποιηθεί.

```
private string g_strFocused = "";
g_strFocused = "username";
g_strFocused = "password";
```

Στο πάτημα του κάθε κουμπιού καλείται η μέθοδος `private void Calculator(object sender)`, η οποία αφού ελέγξει αν ο χρήστης έχει επιλέξει κάποιο από τα πεδία κειμένου, εισάγει την τιμή του κουμπιού στο επιλεγμένο πεδίο κειμένου. Διαφορετικά ενημερώνει το χρήστη ότι προαπαιτείται η επιλογή πεδίου κειμένου.

```
Button l_btn = (Button)sender;
if (g_strFocused == "")
{
      MessageBox.Show("Παρακαλώ, επιλέξτε το πεδίο στο" +
      Environment.NewLine + "οποίο θέλετε να εισάγετε τιμή.",
      "Delta Client", MessageBoxButton.OK,
      MessageBoxImage.Exclamation, MessageBoxResult.OK);
}
else if (g_strFocused == "username")
{
      this.txtUsername.Text += l_btn.Content;
}
else if (g_strFocused == "password")
{
      this.txtPassword.Password += l_btn.Content;
}
```

Εξαίρεση αποτελεί το κουμπί C, για το οποίο στο πάτημά του δεν καλείται η παραπάνω μέθοδος, αλλά για το ενεργοποιημένο πεδίου κειμένου αφαιρεί τον τελευταίο χαρακτήρα.

```
if (g_strFocused == "")
{
      MessageBox.Show("Παρακαλώ, επιλέξτε το πεδίο για το" +
      Environment.NewLine + "οποίο θέλετε να σβήσετε την τιμή
      του.", "Delta Client", MessageBoxButton.OK,
      MessageBoxImage.Exclamation, MessageBoxResult.OK);
}
else if (g_strFocused == "username")
{
      if (this.txtUsername.Text.Length != 0)
      {
            this.txtUsername.Text =
            this.txtUsername.Text.Substring(0,
            this.txtUsername.Text.Length - 1);
      }
}
else if (g_strFocused == "password")
```

```
        {
            if (this.txtPassword.Password.Length != 0)
            {
                this.txtPassword.Password =
                this.txtPassword.Password.Substring(0,
                this.txtPassword.Password.Length - 1);
            }
        }
```

Τέλος, τα κουμπιά 'OK' και 'Cancel' επιβεβαιώνουν ή ακυρώνουν αντίστοιχα τη διαδικασία σύνδεσης. Το πρώτο καλεί τη μέθοδο `this.Authenticate();` για τη σύνδεση στην εφαρμογή ενώ το δεύτερο κλείνει το παράθυρο `this.Close();`
Η μέθοδος Authenticate καλείται επίσης όταν ο χρήστης πατήσει το πλήκτρο 'Enter', στην περίπτωση που η εφαρμογή εκτελείται από συμβατικό υπολογιστή και όχι από Tablet PC.
Συγκεκριμένα η μέθοδος δημιουργεί ένα αντικείμενο τύπου clSecurity το οποίο και υπάρχει στο dll Delta Library και καλείται αφού έχει γίνει εισαγωγή της βιβλιοθήκης στην αρχή της κλάσης `using DeltaLibrary;` Από το αντικείμενο καλείται η μέθοδος CheckAuditorCode, η οποία δέχεται σαν όρισμα τον κωδικό που εισήγαγε ο χρήστης. Σε περίπτωση που επιτύχει η σύνδεση εμφανίζεται το κύριο παράθυρο της εφαρμογής ενώ σε διαφορετική περίπτωση ο χρήστης ενημερώνεται ότι η σύνδεση απέτυχε.

```
        using (clSecurity l_clSec = new clSecurity())
        {
            if (l_clSec.CheckAuditorCode(this.txtPassword.Password))
            {
                winMain l_winMain = new winMain();
                this.Visibility = Visibility.Hidden;
                l_winMain.g_strUser = this.txtUsername.Text;
                l_winMain.Show();
                this.Close();
            }
            else
            {
                MessageBox.Show("Η σύνδεση απέτυχε!", "Delta
                Client", MessageBoxButton.OK,
                MessageBoxImage.Exclamation, MessageBoxResult.OK);
            }
        }
```

Το βασικό παράθυρο της εφαρμογής είναι το `public partial class winMain : Window`. Στον κατασκευαστή (constructor) του παραθύρου εισάγεται στο παράθυρο το event της κλάσης CloseableTabItem μέσω της οποίας τα controls προβάλλονται στο TabControl του παραθύρου με δυνατότητα να κλείνουν από το κουμπί που φέρει το καθένα.

```
        public winMain()
        {
            InitializeComponent();
            this.AddHandler(CloseableTabItem.CloseTabEvent, new
            RoutedEventHandler(this.CloseTab));
        }
```

Για το κλείσιμο της καρτέλας καλείται η παρακάτω μέθοδος

```
        private void CloseTab(object source, RoutedEventArgs args)
        {
            TabItem tabItem = args.Source as TabItem;
            if (tabItem != null)
```

```
            {
                  TabControl tabControl = tabItem.Parent as
                  TabControl;
                  if (tabControl != null)
                        tabControl.Items.Remove(tabItem);
            }
      }
```

Το παράθυρο διαχειρίζεται δύο events το `private void Window_Loaded(object sender, RoutedEventArgs e)` και το `private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)` Το πρώτο καλείται κατά τη δημιουργία του παραθύρου και η βασική λειτουργία του είναι να γράφει στον τίτλο του παραθύρου το όνομα του επιθεωρητή, όπως προέκυψε από τη σύνδεση.

```
      this.Title += g_strUser;
```

Το δεύτερο καλείται κατά το κλείσιμο του παραθύρου.

```
      if (!this.CloseApplication())
      {
            e.Cancel = true;
      }
```

Η μέθοδος που καλείται προβάλει στο χρήστη ενημερωτικό μήνυμα για την επιβεβαίωση του κλεισίματος του παραθύρου.

```
      private Boolean CloseApplication()
      {
            if ((MessageBox.Show("Πρόκειται να κλείσετε την
            εφαρμογή." + Environment.NewLine + "Θέλετε να
            συνεχίσετε;", "Delta Client", MessageBoxButton.YesNo,
            MessageBoxImage.Question) == MessageBoxResult.Yes))
            {
                  return true;
            }
            else
            {
                  return false;
            }
      }
```

Το βασικό παράθυρο περιέχει έξι κουμπιά.

Με το κουμπί ‘Λήψη’, όπως γίνεται κατανοητό και από το όνομά του, γίνεται η λήψη των ερωτηματολογίων για τον συγκεκριμένο επιθεωρητή, εφ’ όσον υπάρχουν. Αρχικά καλείται η μέθοδος CheckForInternet η οποία ελέγχει αν το τερματικό που εκτελεί την εφαρμογή έχει πρόσβαση στο Διαδίκτυο, μιας και απαιτείται για τη λήψη των ερωτηματολογίων. Σε διαφορετική περίπτωση ενημερώνει το χρήστη με σχετικό μήνυμα.

```
      MessageBox.Show("Δεν βρέθηκε σύνδεση με το Διαδίκτυο!" +
      Environment.NewLine + "Ελέγξε τη σύνδεσή σας και
      ξαναπροσπαθήστε.", "Delta Client", MessageBoxButton.OK,
      MessageBoxImage.Exclamation, MessageBoxResult.OK);
```

Μετά τον επιτυχή έλεγχο σύνδεσης στο Διαδίκτυο, δημιουργείται αντικείμενο τύπου Web Service από το project Delta Services.

```
      using (DeltaClientService l_ws = new DeltaClientService())
      {
```

Η διεύθυνση του Web Service διαβάζεται από τις ρυθμίσεις τις εφαρμογής, οι οποίες αναλύονται στη συνέχεια.

```
            l_ws.Url =
            DeltaClient.Properties.Settings.Default.DeltaClient_Delta
            Services_DeltaClientService;
```

Καλείται η μέθοδος strCheckForAudits η οποία δέχεται ως παράμετρο το όνομα χρήστη του επιθεωρητή και επιστρέφει ένα αλφαριθμητικό το οποίο έχει την τιμή "error" αν προκύψει σφάλμα και την τιμή "zero" αν δεν υπάρχουν επιθεωρήσεις. Σε διαφορετική περίπτωση επιστρέφονται οι κωδικοί των επιθεωρήσεων χωρισμένοι με το χαρακτήρα ; (ελληνικό ερωτηματικό).

```
l_strAuditsCount =
l_ws.strCheckForAudits(this.g_strUser);
if (l_strAuditsCount.Equals("error"))
{
      this.Cursor = Cursors.Arrow;
      MessageBox.Show("Ο Application Server δεν είναι
      διαθέσιμος." + Environment.NewLine + "Παρακαλώ,
      δοκιμάστε αργότερα.", "Delta Client",
      MessageBoxButton.OK, MessageBoxImage.Exclamation,
      MessageBoxResult.OK);
}
else if (l_strAuditsCount.Equals("zero"))
{
      this.Cursor = Cursors.Arrow;
      MessageBox.Show("Δεν υπάρχουν νέα ερωτηματολόγια.",
      "Delta Client", MessageBoxButton.OK,
      MessageBoxImage.Information, MessageBoxResult.OK);
}
else
{
```

Αν υπάρχουν ερωτηματολόγια προς λήψη, προτού οι κωδικοί εκχωρηθούν στο αντικείμενο l_strWebFiles, τύπου ArrayList, αφαιρούνται αν υπάρχουν τιμές από προηγούμενη κλήση του Web Service.

```
      if (this.l_strWebFiles.Count > 0)
      {
            this.l_strWebFiles.RemoveRange(0,
            this.l_strWebFiles.Count);
      }
      string[] l_strAuditsCodes =
      l_strAuditsCount.Split(';');
      for (int l_intIndex = 0; l_intIndex <=
      l_strAuditsCount.Split(';').Length - 1;
      l_intIndex++)
      {
            this.l_strWebFiles.Add(
            l_strAuditsCodes.ElementAt(l_intIndex));
      }
```

Επόμενη ενέργεια είναι η κλήση της μεθόδου `this.CheckLocalFiles();`
Η μέθοδος αυτή ελέγχει αν κάποια από τα ερωτηματολόγια που βρέθηκαν προς λήψη, υπάρχουν τοπικά στον υπολογιστή του επιθεωρητή. Με αυτόν τον έλεγχο αποφεύγεται η λήψη ερωτηματολογίων τα οποία ο επιθεωρητής τα είχε κατεβάσει σε προηγούμενο έλεγχο. Αρχικά αφαιρούνται από τη μεταβλητή l_strLocalFiles, τύπου ArrayList, όλες οι τιμές που μπορεί να συγκρατεί από προηγούμενη κλήση της μεθόδου και εισάγει τους κωδικούς των επιθεωρήσεων ελέγχοντας τους φακέλους των ερωτηματολογίων και των αναφορών. Οι κωδικοί μπορούν εύκολα να ανακτηθούν αφού περιέχονται στα ονόματα των αρχείων.

```
if(this.l_strLocalFiles.Count > 0)
{
      this.l_strLocalFiles.RemoveRange(0,
      this.l_strLocalFiles.Count);
}
```

```
//CHECK FOR DOWNLOADED QUESTIONNAIRES
DirectoryInfo l_di = new
DirectoryInfo(winLogin.g_strPathQuestionnaires);
FileInfo[] l_fiFiles = l_di.GetFiles("*.itps");
foreach (FileInfo l_fi in l_fiFiles)
{
      l_strLocalFiles.Add(l_fi.Name.Substring(0, 10));
}
//CHECK FOR REPORTS
DirectoryInfo l_diRep = new
DirectoryInfo(winLogin.g_strPathReports);
FileInfo[] l_fiFilesRep = l_diRep.GetFiles("*.itps");
foreach (FileInfo l_fiRep in l_fiFilesRep)
{
      l_strLocalFiles.Add(l_fiRep.Name.Substring(0, 10));
}
```

Έχοντας δύο λίστες με τους κωδικούς των επιθεωρήσεων που επέστρεψε το Web Service και του κωδικούς για τους οποίους τα ερωτηματολόγια υπάρχουν τοπικά, γίνεται ο έλεγχος αν υπάρχει κάποια επιθεώρηση για την οποία το ερωτηματολόγιο υπάρχει στη λίστα των τοπικών ερωτηματολογίων. Αν υπάρχει εκχωρείται σε καινούρια λίστα.

```
List<int> l_intList = new List<int>();
for (int l_intIndex = 0; l_intIndex <=
this.l_strWebFiles.Count - 1; l_intIndex++)
{
      for (int l_intIndex2 = 0; l_intIndex2 <=
      this.l_strLocalFiles.Count - 1;
      l_intIndex2++)
      {
            if (this.l_strWebFiles[l_intIndex].
            ToString() ==
            this.l_strLocalFiles[l_intIndex2].ToStr
            ing())
            {
                  l_intList.Add(l_intIndex);
            }
      }
}
```

Οι κωδικοί που εκχωρήθηκαν στην καινούρια λίστα, αφαιρούνται από τη λίστα των υπό λήψη ερωτηματολογίων.

```
for (int i = l_intList.Count - 1; i >= 0; i--)
{
      this.l_strWebFiles.RemoveAt(i);
}
```

Επόμενο βήμα είναι να ελεγχθεί αν υπάρχει καινούριο ερωτηματολόγιο προς λήψη. Και στην περίπτωση που υπάρχει και στην περίπτωση που δεν υπάρχει ο επιθεωρητής ενημερώνεται με σχετικό μήνυμα.

```
if (this.l_strWebFiles.Count == 0)
{
      this.Cursor = Cursors.Arrow;
      MessageBox.Show("Δεν υπάρχουν νέα
      ερωτηματολόγια.", "Delta Client",
      MessageBoxButton.OK,
      MessageBoxImage.Information,
      MessageBoxResult.OK);
}
else
{
```

```
this.Cursor = Cursors.Arrow;
if ((MessageBox.Show("Βρέθηκαν " +
this.l_strWebFiles.Count + " ερωτηματολόγια."
+ Environment.NewLine + "Θέλετε να τα
κατεβάσετε;", "Delta Client",
MessageBoxButton.YesNo,
MessageBoxImage.Question) ==
MessageBoxResult.Yes))
{
```

Αν υπάρχουν ερωτηματολόγια προς λήψη και ο επιθεωρητής επιβεβαιώσει τη λήψη τους καλείται η Web Method l_strGetQuestionnaires του Web Service. Η μέθοδος δέχεται ως όρισμα ένα Array με τους κωδικούς των επιθεωρήσεων και επιστρέφει ένα Array με τα ερωτηματολόγια.

```
//DOWNLOAD QUESTIONNAIRES
string[] l_strAuditCodes = new
string[this.l_strWebFiles.Count];
for (int i = 0; i <=
this.l_strWebFiles.Count - 1; i++)
{
      l_strAuditCodes[i] =
      (string)this.l_strWebFiles[i];
}
string[] l_strQuestionnaires = new
string[l_strAuditCodes.Length];
l_strQuestionnaires =
l_ws.strGetQuestionnaires(l_strAuditCod
es);
```

Για κάθε ερωτηματολόγιο που έχει επιστραφεί γίνεται αποκρυπτογράφηση μέσω της μεθόδου Decrypt του αντικειμένου clDecryption από το dll Delta Library. Αυτό είναι απαραίτητο για να διαβαστεί από το ερωτηματολόγιο σε «καθαρή» μορφή ο κωδικός της επιθεώρησης, το κατάστημα της επιθεώρησης και το είδος της επιθεώρησης.

```
//CREATE QUESTIONNAIRES
XmlDocument l_xml = new XmlDocument();
clDecryption l_clDecryption = new
clDecryption();
for (int i = 0; i <=
l_strQuestionnaires.Length - 1; i++)
{
      //READ QUESTIONNAIRE
      l_xml.LoadXml(l_clDecryption.
      Decrypt(l_strQuestionnaires[i],
      winLogin.g_strPathQuestionnaires)
      );
      // create a writer and open the
      file
```

Το κάθε ερωτηματολόγιο αποθηκεύεται στο φάκελο των ερωτηματολογίων. Το αρχείο είναι κρυπτογραφημένο σε itps μορφή και το όνομα του περιλαμβάνει τον κωδικό, το κατάστημα και το είδος της επιθεώρησης χωρισμένα με παύλα '-' (π.χ. AUD-001-10-Biocert-Κρέας.itps).

```
TextWriter l_tw = new
StreamWriter(winLogin.g_strPathQu
estionnaires + "\\" +
l_xml.GetElementsByTagName("CODE"
)[0].InnerText + "-" +
l_xml.GetElementsByTagName("PLACE
")[0].InnerText + "-" +
```

```
                                    l_xml.GetElementsByTagName("TITLE
                                    ")[0].InnerText + ".itps");
                                    // write a line of text to the
                                    file
                                    l_tw.WriteLine(
                                    l_strQuestionnaires[i]);
                                    // close the stream
                                    l_tw.Close();
                              }
```

Με την ολοκλήρωση της λήψης η εφαρμογή ενημερώνει με μήνυμα τον επιθεωρητή.

```
                              MessageBox.Show("Η λήψη των
                              ερωτηματολογίων ολοκληρώθηκε
                              επιτυχώς.", "Delta Client",
                              MessageBoxButton.OK,
                              MessageBoxImage.Information);
                        }
                  }
            }
      }
```

Με το κουμπί ‘Αποστολή’ γίνεται η αποστολή των αναφορών των επιθεωρήσεων στην εταιρεία συμβούλων, εφ’ όσον υπάρχουν. Η δομή της μοιάζει αρκετά με την προηγούμενη της λήψης. Κι εδώ αρχικά καλείται η μέθοδος CheckForInternet η οποία ελέγχει αν το τερματικό που εκτελεί την εφαρμογή έχει πρόσβαση στο Διαδίκτυο. Στη συνέχεια από τον φάκελο των αναφορών διαβάζονται τα αρχεία (αναφορές).

```
DirectoryInfo l_diRep = new
DirectoryInfo(winLogin.g_strPathReports);
FileInfo[] l_fiFilesRep = l_diRep.GetFiles("*.itps");
if (l_fiFilesRep.Length == 0)
{
      MessageBox.Show("Δεν υπάρχουν αναφορές.", "Delta Client",
      MessageBoxButton.OK, MessageBoxImage.Information,
      MessageBoxResult.OK);
      return;
}
```

Αν βρεθούν αναφορές, για την κάθε μία γίνεται αποκρυπτογράφηση ώστε να διαβαστεί ο κωδικός και η ημερομηνία της επιθεώρησης.

```
if ((MessageBox.Show("Βρέθηκαν " + l_fiFilesRep.Length + "
αναφορές." + Environment.NewLine + "Θέλετε να τις
αποστείλετε;", "Delta Client", MessageBoxButton.YesNo,
MessageBoxImage.Question) == MessageBoxResult.Yes))
{
      using (DeltaClientService l_ws = new
      DeltaClientService())
      {
            l_ws.Url =
            DeltaClient.Properties.Settings.Default.DeltaClient
            _DeltaServices_DeltaClientService;
            //READ REPORTS
            XmlDocument l_xml = new XmlDocument();
            clDecryption l_clDecryption = new clDecryption();
            String l_strResult = "ok";
            foreach (FileInfo l_fiRep in l_fiFilesRep)
            {
                  //READ QUESTIONNAIRE
                  string l_strFileNew =
                  winLogin.g_strPathReports + "\\" +
                  l_fiRep.Name;
```

```
FileStream FsNew = new
FileStream(l_strFileNew,
FileMode.OpenOrCreate);
BinaryReader bnNew = new BinaryReader(FsNew,
Encoding.UTF8);
String l_str;
ASCIIEncoding l_enc = new ASCIIEncoding();
l_str =
l_enc.GetString(bnNew.ReadBytes(Convert.ToInt
32(bnNew.BaseStream.Length)));
bnNew.Close();
FsNew.Close();
FsNew.Dispose();
l_xml.LoadXml(l_clDecryption.Decrypt(l_str,
winLogin.g_strPathReports));
//SEND REPORTS
string[] l_strDate = new string[3];
l_strDate =
l_xml.GetElementsByTagName("DATE")[0].InnerTe
xt.Split('/');
if (l_strDate[0].Length == 1)
{
      l_strDate[0] = "0" + l_strDate[0];
}
if (l_strDate[1].Length == 1)
{
      l_strDate[1] = "0" + l_strDate[1];
}
string l_strFinalDate = String.Join("/",
l_strDate);
```

Τέλος, καλείται η Web Method l_strSendReports του Web Service η οποία και αποστέλλει στην εταιρεία συμβούλων την αναφορά (αποθηκεύει στη βάση).

```
l_strResult =
l_ws.strSendReports(l_xml.GetElementsByTagNam
e("CODE")[0].InnerText.ToString(), l_str,
l_strFinalDate);
```

Αν η μέθοδος δεν επιστρέψει ως αποτέλεσμα το αλφαριθμητικό "ok" τότε παρουσιάστηκε κάποιο πρόβλημα και ενημερώνεται με σχετικό μήνυμα ο επιθεωρητής.

```
if (!l_strResult.Equals("ok"))
{
      this.Cursor = Cursors.Arrow;
      MessageBox.Show("Ο Application Server
      δεν είναι διαθέσιμος." +
      Environment.NewLine + "Παρακαλώ,
      δοκιμάστε αργότερα.", "Delta Client",
      MessageBoxButton.OK,
      MessageBoxImage.Exclamation,
      MessageBoxResult.OK);
      return;
}
if (File.Exists(l_strFileNew))
{
      File.Delete(l_strFileNew);
}
    }
}
```

Διαφορετικά ενημερώνεται για την ολοκλήρωση της αποστολής των αναφορών.

```
            MessageBox.Show("Η αποστολή των αναφορών ολοκληρώθηκε
            επιτυχώς.", "Delta Client", MessageBoxButton.OK,
            MessageBoxImage.Information);
      }
```

Τελευταίο είναι το κουμπί ‘Επιθεώρηση’. Με το πάτημά του από τον επιθεωρητή ο πρώτος έλεγχος που γίνεται είναι αν υπάρχει ήδη ανοικτό ερωτηματολόγιο. Αυτός ο έλεγχος είναι υποχρεωτικός διότι δεν μπορούν να ανοίξουν δύο ερωτηματολόγια ταυτόχρονα αφού ένας επιθεωρητής δεν μπορεί να διεξάγει δύο επιθεωρήσεις την ίδια στιγμή.

```
      if (this.tabWindows.Items.Count > 0)
      {
            if ((MessageBox.Show("Το ερωτηματολόγιο που έχετε ανοίξει
            πρόκειται να κλείσει." + Environment.NewLine + "Θέλετε να
            συνεχίσετε;", "Delta Client", MessageBoxButton.YesNo,
            MessageBoxImage.Question) == MessageBoxResult.Yes))
            {
                  for (int i = this.tabWindows.Items.Count - 1;
                  i >= 0; i--)
                  {
                        this.tabWindows.Items.RemoveAt(i);
                  }
            }
            else
            {
                  return;
            }
      }
```

Αλλιώς, ξεκινά η διαδικασία επιλογής ερωτηματολογίου. Πρώτα διαβάζονται όλα τα αρχεία από τον φάκελο των ερωτηματολογίων και συγκρατιούνται τα ονόματα των ερωτηματολογίων, χωρίς την κατάληξη itps.

```
      DirectoryInfo l_di = new
      DirectoryInfo(winLogin.g_strPathQuestionnaires);
      FileInfo[] l_fiFiles = l_di.GetFiles("*.itps");
      int l_intFilesCount = 0;
      ArrayList l_strFiles = new ArrayList();
      foreach (FileInfo l_fi in l_fiFiles)
      {
            l_intFilesCount += 1;
            l_strFiles.Add(l_fi.Name.Substring(0, l_fi.Name.Length -
            5));
      }
      if (l_intFilesCount == 0)
      {
            MessageBox.Show("Δεν υπάρχουν διαθέσιμα ερωτηματολόγια!"
            + Environment.NewLine + "Ελέγξε από τη 'Λήψη' για τυχόν
            καινούρια.", "Delta Client", MessageBoxButton.OK,
            MessageBoxImage.Information, MessageBoxResult.OK);
      }
      else
      {
```

Αν βρεθούν ερωτηματολόγια εμφανίζεται το παράθυρο winSelectFiles.

```
            winSelectFiles l_winSelectFiles = new
            winSelectFiles(this);
            l_winSelectFiles.g_strFiles = l_strFiles;
            l_winSelectFiles.ShowDialog();
```

Στην εκκίνηση του παραθύρου και αφού ζωγραφιστούν όλα τα controls που περιλαμβάνει, δημιουργείται μια λίστα με τα ερωτηματολόγια που βρέθηκαν και με κουμπί RadioButton δίπλα σε κάθε ένα.

```
for (int i = 0; i <= g_strFiles.Count - 1; i++)
{
      RadioButton l_rbFile = new RadioButton();
      l_rbFile.FontSize = 14;
      l_rbFile.Content = " " +
      g_strFiles[i].ToString().Substring(11);
      l_rbFile.Tag = g_strFiles[i];
      l_rbFile.Checked += new
      RoutedEventHandler(l_rbFile_Checked);
      l_rbFile.Margin = new Thickness(10, 5, 0, 5);
      g_stackPanel.Children.Add(l_rbFile);
}
```

Ο επιθεωρητής επιλέγοντας ερωτηματολόγιο, καλείται το event του παραθύρου `private void l_rbFile_Checked(object sender, RoutedEventArgs e)` μέσω του οποίου εκχωρείται σε global μεταβλητή το όνομα του αρχείου που επιλέχθηκε.

```
RadioButton l_rb = (RadioButton)sender;
g_strFileName = l_rb.Tag.ToString();
```

Με το πάτημα του κουμπιού 'OK' το όνομα του αρχείου που επιλέχθηκε αποθηκεύεται στη μεταβλητή g_strFileName του κεντρικού παραθύρου (winMain).

```
if (g_strFileName == "")
{
      MessageBox.Show("Η έναρξη της επιθεώρησης απέτυχε!" +
      Environment.NewLine + "Παρακαλώ, επιλέξτε
      ερωτηματολόγιο.", "Delta Client", MessageBoxButton.OK,
      MessageBoxImage.Exclamation, MessageBoxResult.OK);
}
else
{
      g_winMain.g_blnIsOK = true;
      g_winMain.g_strFileName = g_strFileName + ".itps";
      this.Close();
}
```

Η μεταβλητή που συγκρατεί το όνομα του αρχείου θα εκχωρηθεί στην αντίστοιχη μεταβλητή του UserControl ucAudit, αφού πρώτα δημιουργηθεί.

```
            ucAudit l_ucAudit = new ucAudit();
            l_ucAudit.g_strFileName = g_strFileName;
}
```

Στη δημιουργία του ucAudit ως καρτέλα στο κεντρικό παράθυρο, διαβάζεται το itps αρχείο (ερωτηματολόγιο) από το φάκελο των ερωτηματολογίων. Το όνομα του αρχείου συγκρατείται στη μεταβλητή l_strFileName, όπως αναλύθηκε παραπάνω. Στο πάνω μέρος εμφανίζεται επεξηγηματικό κείμενο σχετικά με τα βήματα της επιθεώρησης.

```
l_txtSubtitle.Text = "Επιλέξτε απάντηση για την κάθε ερώτηση
και στη συνέχεια ορίστε σημαία για τον κάθε τομέα. Επίσης
μπορείτε να γράψετε παρατηρήσεις για τον κάθε τομέα. Με την
ολοκλήρωση της επιθεώρησης υπογράψτε στο τέλος του
ερωτηματολογίου και πατήστε το κουμπί 'Λήξη Επιθεώρησης'.";
```

Στη συνέχεια ξεκινά η δημιουργία του ερωτηματολογίου.

```
//CREATE QUESTIONNAIRE
XmlDocument l_xml = new XmlDocument();
clDecryption l_clDecryption = new clDecryption();
string l_strFileNew = winLogin.g_strPathQuestionnaires + "\\" +
g_strFileName.Substring(0, g_strFileName.Length);
```

```
FileStream FsNew = new FileStream(l_strFileNew,
FileMode.OpenOrCreate);
BinaryReader bnNew = new BinaryReader(FsNew, Encoding.UTF8);
String l_str;
ASCIIEncoding l_enc = new ASCIIEncoding();
l_str =
l_enc.GetString(bnNew.ReadBytes(Convert.ToInt32(bnNew.BaseStrea
m.Length)));
bnNew.Close();
FsNew.Close();
FsNew.Dispose();
l_xml.LoadXml(l_clDecryption.Decrypt(l_str,
winLogin.g_strPathQuestionnaires));
```

Στο συνέχεια εμφανίζονται οι πληροφορίες της επιθεώρησης. Εμφανίζεται ο πελάτης και το είδος της επιθεώρησης, στοιχεία τα οποία διαβάζονται από το όνομα του αρχείου. Επίσης εμφανίζεται η διεύθυνση του επιθεωρούμενου καταστήματος, πληροφορία που διαβάζεται από τα περιεχόμενα του ερωτηματολογίου.

```
l_txtAddress.Text =
l_xml.GetElementsByTagName("ADDRESS")[0].InnerText;
```

Στη συνέχεια ζωγραφίζονται τα περιεχόμενα, διατηρώντας τη δομή του XML αρχείου. Δηλαδή, οι ερωτήσεις εμφανίζονται κάτω από τον τομέα που ανήκουν.

```
XmlNodeList l_xmlNodeListSections =
l_xml.GetElementsByTagName("SECTION");
foreach (XmlNode l_xmlNodeSection in l_xmlNodeListSections)
{
      XmlElement l_xmlElementSection =
      (XmlElement)l_xmlNodeSection;
      XmlNodeList l_xmlNodeListQuestions =
      l_xmlElementSection.GetElementsByTagName("QUESTION");
      if (l_xmlNodeListQuestions.Count > 0)
      {
            this.AddSection(l_xmlElementSection.
            GetElementsByTagName("NAME")[0].InnerText,
            true,true);
            foreach (XmlNode l_xmlNodeQuestion in
            l_xmlNodeListQuestions)
            {
                  XmlElement l_xmlElementQuestion =
                  (XmlElement)l_xmlNodeQuestion;
                  this.AddQuestion(g_btnQuestion,
                  l_xmlElementQuestion.GetElementsByTagName("CA
                  PTION")[0].InnerText, true, ref
                  l_spQuestions);
            }
            this.AddNotes(g_btnQuestion, null, false, ref
            l_spQuestionsNotes, l_xmlNodeListQuestions.Count);
      }
      else
      {
            this.AddSection(l_xmlElementSection.GetElementsByTa
            gName("NAME")[0].InnerText, true,false);
      }
}
this.AddSignatures();
```

Η μέθοδος που εισάγει έναν τομέα είναι η `private void AddSection(String a_strText, bool a_blnHasQuestions)` Δέχεται ως παραμέτρους το όνομα του τομέα και μία boolean μεταβλητή η οποία δηλώνει αν ο τομέας έχει ερωτήσεις. Αν

δεν έχει τότε αποτελεί υπερτομέα. Δίπλα στο όνομα του κάθε τομέα (όχι των υπερτομέων) εισάγεται ένα ComboBox με τις δυνατές επιλογές (σημαίες τομέα).

```
// Add Section combo
ComboBox l_cmbSection = new ComboBox();
l_cmbSection.Name = "cmbSection_" + g_intSectionsCount;
l_cmbSection.Width = 90;
l_cmbSection.Height = 25;
l_cmbSection.Items.Add("ΠΡΑΣΙΝΗ");
l_cmbSection.Items.Add("ΚΟΚΚΙΝΗ");
l_cmbSection.Items.Add("ΜΠΛΕ");
l_cmbSection.SelectedIndex = 0;
```

Η μέθοδος που εισάγει ερώτηση είναι η `private void AddQuestion(object sender, String a_strText, ref StackPanel a_spQuestions)`. Δέχεται ως παραμέτρους το object (κουμπί) από το οποίο κλήθηκε η μέθοδος, το όνομα της ερώτησης και την αναφορά (by ref argument) στο StackPanel που θα εισαχθεί. Δίπλα στο όνομα της κάθε ερώτησης εισάγεται ένα ComboBox με τις δυνατές επιλογές.

```
// Add Question combo
ComboBox l_cmbQuestion = new ComboBox();
l_cmbQuestion.Name = "cmbQuestion_" +
l_btnCurrent.Name.Split('_').ElementAt(1) + "_" +
g_intQuestionsCount;
l_cmbQuestion.Width = 60;
l_cmbQuestion.Height = 25;
l_cmbQuestion.Items.Add("ΝΑΙ");
l_cmbQuestion.Items.Add("ΟΧΙ");
l_cmbQuestion.Items.Add("Δ/Α");
l_cmbQuestion.SelectedIndex = 2;
```

Δίπλα στις ερωτήσεις κάθε τομέα, εισάγεται και πλαίσιο στο οποίο ο επιθεωρητής μπορεί να γράψει τις παρατηρήσεις που εντόπισε στον τομέα. Η μέθοδος αυτή είναι η `private void AddNotes(object sender, ref StackPanel a_spQuestionsNotes, int a_intQuestionsCount)` Δέχεται ως παραμέτρους το object (κουμπί) από το οποίο κλήθηκε η μέθοδος, την αναφορά (by ref argument) στο StackPanel που θα εισαχθεί και τον αριθμό των ερωτήσεων του τομέα. Η τελευταία παράμετρος χρησιμοποιείται στη δήλωση του ύψους του StackPanel των παρατηρήσεων, αφού το ύψος του είναι δυναμικό και ισούται με το ύψος των πεδίων κειμένου των ερωτήσεων.

```
l_border.Height = a_intQuestionsCount * 25;
```

Τέλος, καλείται η μέθοδος `private void AddSignatures()` η οποία εισάγει στο κάτω μέρος του ερωτηματολογίου τα δύο StackPanels εντός των οποίων υπογράφουν ο επιθεωρητής και ο πελάτης.

```
InkCanvas l_signAuditor = new InkCanvas();
l_signAuditor.Name = "signAuditor";
InkCanvas l_signCustomer = new InkCanvas();
l_signCustomer.Name = "signCustomer";
```

Τα StackPanels των υπογραφών αλλά και των παρατηρήσεων, περιέχουν αντικείμενα τύπου InkCanvas. Έτσι είναι δυνατή η χρησιμοποίηση του pen από το tablet PC για την εισαγωγή κειμένου ή υπογραφών. Επίσης υπάρχει δυνατότητα διόρθωσης των περιεχομένων τους, ενεργοποιώντας το CheckBox που βρίσκεται στο πάνω μέρος του παραθύρου.

```
CheckBox l_chkEraser = new CheckBox();
l_chkEraser.FontSize = 12;
l_chkEraser.Height = 30;
l_chkEraser.Click += new
System.Windows.RoutedEventHandler(l_chkEraser_Click);
l_chkEraser.Content = " Διόρθωση Κειμένου";
```

```
l_chkEraser.Margin = new Thickness(20,0,0,0);
l_spButton.Children.Add(l_chkEraser);
```

Με την ενεργοποίηση ή απενεργοποίηση του παραπάνω CheckBox, καλείται το παρακάτω event το οποίο και δηλώνει αν εντός των InkCanvas θα εισάγεται κείμενο ή θα σβήνεται.

```
CheckBox l_objCurrent = (CheckBox)sender;
this.ChangeInkCanvasStatuses(g_stackPanel,   0,   l_objCurrent,
(bool)l_objCurrent.IsChecked);
```

Η μέθοδος ChangeInkCanvasStatuses αλλάζει την κατάσταση όλων των InkCanvas που υπάρχουν στο παράθυρο.

```
InkCanvas l_inkCurrent = new InkCanvas();
string typeName = parent.GetType().Name;
if (typeName == "InkCanvas")
{
      l_inkCurrent = (InkCanvas)parent;
      if(a_blnIsChecked)
      {
            l_inkCurrent.EditingMode =
            InkCanvasEditingMode.EraseByPoint;
      }
      else
      {
            l_inkCurrent.EditingMode =
            InkCanvasEditingMode.Ink;
      }
}
else
{
      for (int i = 0; i <
      VisualTreeHelper.GetChildrenCount(parent); i++)
      {
            DependencyObject child =
            VisualTreeHelper.GetChild(parent, i);
            ChangeInkCanvasStatuses(child, level + 1,
            a_btnSender, a_blnIsChecked);
      }
}
```

Για την ολοκλήρωση της επιθεώρησης ο επιθεωρητής πατάει το κουμπί ‘Λήξη Επιθεώρησης’.

```
if ((MessageBox.Show("Πρόκειται να λήξετε την επιθεώρηση." +
Environment.NewLine + "Μετά τη λήξη της δεν μπορείτε να
επανέλθετε." + Environment.NewLine + "Θέλετε να συνεχίσετε;",
"Delta Client", MessageBoxButton.YesNo, MessageBoxImage.Question)
== MessageBoxResult.Yes))
```

Αρχικά διαβάζεται το ερωτηματολόγιο ώστε στη συνέχεια να προστεθούν οι απαντήσεις των ερωτήσεων, οι σημαίες των τομέων και οι υπογραφές.

```
// CREATE XML REPORT
XmlDocument l_xml = new XmlDocument();
clDecryption l_clDecryption = new clDecryption();
string l_strFileNew = winLogin.g_strPathQuestionnaires + "\\" +
g_strFileName;
FileStream FsNew = new FileStream(l_strFileNew,
FileMode.OpenOrCreate);
BinaryReader bnNew = new BinaryReader(FsNew, Encoding.UTF8);
String l_str;
ASCIIEncoding l_enc = new ASCIIEncoding();
l_str = l_enc.GetString(bnNew.ReadBytes(
Convert.ToInt32(bnNew.BaseStream.Length)));
```

```
bnNew.Close();
FsNew.Close();
FsNew.Dispose();
l_xml.LoadXml(l_clDecryption.Decrypt(l_str,
winLogin.g_strPathQuestionnaires));
```

Στη συνέχεια εισάγεται η τρέχουσα ημερομηνία, στο πεδίο που συγκρατεί την ημερομηνία λήξης της επιθεώρησης.

```
l_xml.GetElementsByTagName("DATE")[0].InnerText =
(DateTime.Now.Date.Day + "/" + DateTime.Now.Date.Month + "/" +
DateTime.Now.Date.Year).ToString();
```

Για κάθε τομέα διαβάζεται η σημαία που εισήγαγε ο επιθεωρητής και εκχωρείται στο αντίστοιχο πεδίο.

```
// ADD ANSWERS TO SECTIONS AND QUESTIONS
XmlNodeList l_xmlNodeListSections =
l_xml.GetElementsByTagName("SECTION");
int l_intSectionNumber = 0;
int l_intQuestionNumber = 0;
bool l_blnAuditorSign = true;
bool l_blnCustomerSign = true;
foreach (XmlNode l_xmlNodeSection in l_xmlNodeListSections)
{
  l_intSectionNumber++;
  XmlElement l_xmlElementSection = (XmlElement)l_xmlNodeSection;
  // FIND ALL SECTIONS COMBOBOXES
  for (int i = 0; i <
  VisualTreeHelper.GetChildrenCount(g_stackPanel); i++)
  {
    DependencyObject child =
    VisualTreeHelper.GetChild(g_stackPanel,
    i);
    StackPanel l_sp = new StackPanel();
    ComboBox l_cmbCurrent = new ComboBox();
    InkCanvas l_inkCurrent = new InkCanvas();
    string typeName = child.GetType().Name;
    if (typeName == "StackPanel")
    {
      l_sp = (StackPanel)child;
      for (int j = 0; j <= l_sp.Children.Count - 1; j++)
      {
        typeName = l_sp.Children[j].GetType().Name;
        if (typeName == "ComboBox")
        {
          l_cmbCurrent = (ComboBox)l_sp.Children[j];
          if (l_cmbCurrent.Name == "cmbSection_" +
          l_intSectionNumber)
          {
            l_xmlElementSection.
            GetElementsByTagName("FLAG")[0].InnerText =
            l_cmbCurrent.SelectedValue.ToString();
          }
        }
        else if (typeName == "Border")
        {
```

Τα περιεχόμενα των πεδίων των παρατηρήσεων καθώς και των πεδίων που συγκρατούν τις υπογραφές, αποθηκεύονται στα αντίστοιχα πεδία της αναφοράς, μετά τη μετατροπή τους από τη μέθοδο ConvertCanvasToText.

```
          Border l_border = (Border)l_sp.Children[j];
          l_inkCurrent = (InkCanvas)l_border.Child;
```

```
if (l_inkCurrent.Name == "inkCanvas_" +
l_intSectionNumber)
{
  l_xmlElementSection.
  GetElementsByTagName("NOTES")[0].InnerText =
  this.ConvertCanvasToText(l_inkCurrent);
}
else if (l_inkCurrent.Name == "signAuditor" &&
l_blnAuditorSign)
{
  l_xml. GetElementsByTagName("AUDITOR")[0].InnerText =
  this.ConvertCanvasToText(l_inkCurrent);
  l_blnAuditorSign = false;
}
else if (l_inkCurrent.Name == "signCustomer" &&
l_blnCustomerSign)
{
  l_xml. GetElementsByTagName("CUSTOMER")[0].InnerText =
  this.ConvertCanvasToText(l_inkCurrent);
  l_blnCustomerSign = false;
}
```

Η μέθοδος `private string ConvertCanvasToText(InkCanvas a_inkCurrent)` δέχεται ως όρισμα το αντικείμενο InkCanvas και αρχικά δημιουργεί μια προσωρινή εικόνα του περιεχομένου του.

```
//CREATE TEMP IMAGE
MemoryStream ms = new MemoryStream();
FileStream fs = new FileStream(winLogin.g_strPathQuestionnaires
+ "\\temp.jpg", FileMode.OpenOrCreate);
RenderTargetBitmap rtb = new
RenderTargetBitmap((int)a_inkCurrent.Width,
(int)a_inkCurrent.Height, 96d, 96d, PixelFormats.Default);
rtb.Render(a_inkCurrent);
JpegBitmapEncoder encoder = new JpegBitmapEncoder();
encoder.Frames.Add(BitmapFrame.Create(rtb));
encoder.Save(fs);
fs.Close();
```

Έπειτα η εικόνα, μέσω της κωδικοποίησης base64, εξάγεται σε κείμενο ώστε να μπορεί να εκχωρηθεί σε XML αρχείο.

```
//CONVERT JPEG IMAGE TO BASE64 ENCODING
System.IO.FileStream inFile = default(System.IO.FileStream);
byte[] binaryData = null;
inFile = new
System.IO.FileStream(winLogin.g_strPathQuestionnaires +
"\\temp.jpg", System.IO.FileMode.Open,
System.IO.FileAccess.Read);
binaryData = new byte[inFile.Length];
int inFileLength = (int)inFile.Length;
string bytesRead = Convert.ToString(inFile.Read(binaryData, 0,
inFileLength));
inFile.Close();
return Convert.ToBase64String(binaryData);
```

Για κάθε ερώτηση εισάγεται η απάντηση.

```
XmlNodeList l_xmlNodeListQuestions =
l_xmlElementSection.GetElementsByTagName("QUESTION");
if (l_xmlNodeListQuestions.Count > 0)
{
  foreach (XmlNode l_xmlNodeQuestion in l_xmlNodeListQuestions)
  {
    l_intQuestionNumber++;
```

```
            XmlElement l_xmlElementQuestion =
            (XmlElement)l_xmlNodeQuestion;
            // FIND ALL QUESTIONS COMBOBOXES
            for (int i = 0; i <
            VisualTreeHelper.GetChildrenCount(g_stackPanel); i++)
            {
              DependencyObject child =
              VisualTreeHelper.GetChild(g_stackPanel, i);
              StackPanel l_sp = new StackPanel();
              StackPanel l_sp2 = new StackPanel();
              StackPanel l_sp3 = new StackPanel();
              ComboBox l_cmbCurrent = new ComboBox();
              string typeName = child.GetType().Name;
              if (typeName == "StackPanel")
              {
                l_sp = (StackPanel)child;
                for (int j = 0; j <= l_sp.Children.Count - 1; j++)
                {
                  typeName = l_sp.Children[j].GetType().Name;
                  if (typeName == "StackPanel")
                  {
                    l_sp2 = (StackPanel)l_sp.Children[j];
                    for (int k = 0; k <= l_sp2.Children.Count - 1; k++)
                    {
                      typeName = l_sp2.Children[k].GetType().Name;
                      if (typeName == "StackPanel")
                      {
                        l_sp3 = (StackPanel)l_sp2.Children[k];
                        for (int l = 0; l <= l_sp3.Children.Count - 1;
                        l++)
                        {
                          typeName = l_sp3.Children[l].GetType().Name;
                          if (typeName == "ComboBox")
                          {
                            l_cmbCurrent = (ComboBox)l_sp3.Children[l];
                            if (l_cmbCurrent.Name == "cmbQuestion_" +
                            l_intSectionNumber + "_" +
                            l_intQuestionNumber)
                            {
                              l_xmlElementQuestion.GetElementsByTagName(
                              "ANSWER")[0].InnerText =
                              l_cmbCurrent.SelectedValue.ToString();
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
```

Με την ολοκλήρωση της εκχώρησης των αποτελεσμάτων, τα δεδομένα κρυπτογραφούνται και αποθηκεύονται στο φάκελο των αναφορών. Το όνομα του αρχείου παραμένει ίδιο έχοντας προστεθεί στο τέλος η ημερομηνία λήξης.

```
  // CREATE ENCRYPTED FILE
  clEncryption l_clEncryption = new clEncryption();
```

```
l_clEncryption.Encrypt(l_xml.InnerXml, winLogin.g_strPathReports);
File.Move(winLogin.g_strPathReports + "\\temp.itps",
winLogin.g_strPathReports + "\\" + g_strFileName.Substring(0,
g_strFileName.Length - 5) + "-" +
l_xml.GetElementsByTagName("DATE")[0].InnerText.Replace('/','.') +
".itps");
```

Στη συνέχεια, διαγράφεται το προσωρινό αρχείο της κρυπτογράφησης και το ερωτηματολόγιο από τον αντίστοιχο φάκελο.

```
if (File.Exists(winLogin.g_strPathReports + "\\temp.xml"))
{
  File.Delete(winLogin.g_strPathReports + "\\temp.xml");
}
// DELETE QUESTIONNAIRE
if (File.Exists(winLogin.g_strPathQuestionnaires + "\\" +
g_strFileName))
{
  File.Delete(winLogin.g_strPathQuestionnaires + "\\" +
  g_strFileName);
}
```

Τέλος, κλείνει από το βασικό παράθυρο η καρτέλα που ερωτηματολογίου.

```
CloseableTabItemDemo.CloseableTabItem l_CloseableTabItem =
(CloseableTabItemDemo.CloseableTabItem)this.Parent;
TabControl l_TabControl = (TabControl)l_CloseableTabItem.Parent;
l_TabControl.Items.RemoveAt(0);
```

Τα κουμπιά ‘Καρτέλες’, ‘Περί…’ και ‘Έξοδος’ έχουν την ίδια λειτουργικότητα με τα αντίστοιχα κουμπιά του Delta Server (Ενότητα 6.2).

Τέλος, στο αρχείο app.config ορίζονται οι ρυθμίσεις λειτουργία της εφαρμογής. Με την εγκατάστασή της στον υπολογιστή του επιθεωρητή, μέσω του αρχείου μπορούμε να αλλάξουμε τη διεύθυνση μέσω της οποίας καλούνται τα Web Services.

```
<setting name="DeltaClient_DeltaServices_DeltaClientService"
serializeAs="String">
      <value>
            http://79.166.212.169:7101/DeltaServices-
            DeltaService-context-root/DeltaClientPort
      </value>
</setting>
```

Αυτό είναι πολύ χρήσιμο στην περίπτωση που αλλάξει η διεύθυνση του Application Server των Web Services. Σε διαφορετική περίπτωση θα έπρεπε να δημιουργηθεί καινούρια έκδοση της εφαρμογής με ενσωματωμένη την αλλαγή.

6.4 Delta Services

Τα Web Services αποτελούν το μέσο ολοκλήρωσης των επιμέρους εφαρμογών. Είναι η «καρδιά» της Service-Oriented Architecture, πάνω στην οποία αναπτύχθηκε το σύστημα. Λόγω της ετερογενούς φύσης του συστήματος, επιτυγχάνουν την επικοινωνία των εφαρμογών του διαχειριστή και των επιθεωρητών με τη βάση δεδομένων. Οι μεν πρώτες λειτουργούν σε Windows λειτουργικό σύστημα, η δε βάση λειτουργεί σε Linux περιβάλλον. Αναπτύχθηκαν σε Linux περιβάλλον με χρήση του Oracle JDeveloper 11g Update2 IDE σε γλώσσα Java και «τρέχουν» σε Oracle WebLogic Server 11g R1 Application Server επίσης σε Linux περιβάλλον (Linux openSUSE 11.1).

Είναι απόφαση του εκάστοτε Project Manager αν θα αναπτυχθούν διαφορετικά Web Services με μόνο μία Web Method το καθένα ή αν θα αναπτυχθεί ένα ενσωματώνοντας όλες τις μεθόδους. Το θετικό στην πρώτη περίπτωση είναι ότι το κάθε Web Service κάνει μόνο μία εργασία και έτσι είναι πιο ξεκάθαρη η λειτουργία του. Στο σύστημά μας επιλέχθηκε η δεύτερη περίπτωση. Αφ' ενός λόγω του γεγονότος ότι όλες οι μέθοδοι αφορούσαν το ίδιο σύστημα, άρα δεν υπήρχε λόγος να βαρύνουμε τον Application Server στο να διαχειρίζεται πολλά services, αφ' ετέρου το κάθε Web Service δεσμεύει (λειτουργεί) σε μία θύρα (port). Αυτό συνεπάγεται ότι θα έπρεπε να δεσμευτούν τέσσερις θύρες αντί για μία, κάτι που γενικά είναι επίφοβο και θα πρέπει να αποφεύγεται επειδή τα Firewalls, που θα λειτουργούν στους υπολογιστές οι οποίοι μέσω των εφαρμογών θα καλούν τα Web Services, θα έπρεπε να διαμορφωθούν ώστε να μην μπλοκάρουν τις συγκεκριμένες θύρες.

Η πρώτη Web Method καλείται από το Delta Client και επιστρέφει, αν υπάρχουν, τους κωδικούς των επιθεωρήσεων στις οποίες θα υλοποιηθούν από κάποιον επιθεωρητή. `public String strCheckForAudits (String a_strAuditorCode)` Δέχεται ως παράμετρο το κωδικό του επιθεωρητή, τον οποίο πληκτρολογεί ο επιθεωρητής κατά τη σύνδεσή του στο Delta Client, όπως είδαμε παραπάνω. Η επιστροφή της είναι αλφαριθμητικό και οι τιμές που επιστρέφει είναι "error" σε περίπτωση που προκύψει κάποια εξαίρεση, "zero" αν δεν βρεθούν διαθέσιμα ερωτηματολόγια ή τον κωδικό της επιθεώρησης. Σε περίπτωση που βρεθούν περισσότερες από μία επιθεωρήσεις για τον συγκεκριμένο επιθεωρητή, επιστρέφει ένα αλφαριθμητικό με όλους τους κωδικούς χωρίζοντας τους με το σύμβολο ; (ελληνικό ερωτηματικό) π.χ. "AUD-001-10;AUD-002-10".

```
catch(Exception e)
{
	return "error";
}

if (l_intAuditsCount == 0)
{
	return "zero";
}
else
{
	return l_strAuditsCodes;
}
```

Αρχικά μέσω του JDBC συνδεόμαστε στη βάση.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Με χρήση ενός αντικειμένου τύπου statement, ελέγχουμε τον πίνακα των επιθεωρητών ώστε να μας επιστραφούν, σε αντικείμενο τύπου ResultSet, τα δεδομένα του επιθεωρητή με βάση την παράμετρο της Web Method (τον κωδικό του επιθεωρητή).

```
rs = s.executeQuery("select * from TAUDITORS where S_AUDITORCODE =
'"+ a_strAuditorCode +"'");
```

Επιλέγουμε το κλειδί του πίνακα για τον συγκεκριμένο επιθεωρητή και με βάση αυτό αναζητούμε στον πίνακα των αναθέσεων τους κωδικούς των επιθεωρήσεων.

```
//GET AUDITOR ID
while(rs.next())
{
	l_intAuditorID = rs.getInt("PKI_AUDITORID");
}
rs = s.executeQuery("select * from TAUDITORAUDIT where FKI_AUDITORID
= "+ l_intAuditorID);
```

Στη συνέχεια από των πίνακα των επιθεωρήσεων επιλέγονται οι κωδικοί των επιθεωρήσεων.

```
//GET AUDITS' COUNT & AUDIT'S CODE
while(rs.next())
{
	rs2 = s2.executeQuery("select * from TAUDITS where PKI_AUDITID
	= "+ rs.getInt("FKI_AUDITID"));

	while (rs2.next())
	{
		if (rs2.getInt("I_AUDITAUDCOUNT") >
		rs2.getInt("I_AUDITREPCOUNT"))
		{
			if (l_intAuditsCount == 0)
			{
				l_strAuditsCodes =
				rs2.getString("S_AUDITCODE");
			}
			else
			{
				l_strAuditsCodes += ";" +
				rs2.getString("S_AUDITCODE");
			}
			l_intAuditsCount++;
		}
	}
}
```

Όπως μπορούμε να δούμε στο παραπάνω block κώδικα, επιλέγονται οι κωδικοί των επιθεωρήσεων για τους οποίους το πεδίο `I_AUDITAUDCOUNT` (αριθμός επιθεωρητών) είναι μεγαλύτερο από το πεδίο `I_AUDITREPCOUNT` (αριθμός αναφορών). Αυτός ο έλεγχος είναι απαραίτητος γιατί θα πρέπει να αγνοηθούν οι επιθεωρήσεις για τις οποίες ο αριθμός των αναφορών ισούται με τον αριθμό των επιθεωρητών (επιθεωρήσεις που έχουν ολοκληρωθεί).

Η δεύτερη Web Method καλείται κι αυτή από το Delta Client και επιστρέφει έναν πίνακα αλφαριθμητικών που συγκρατούν τα ερωτηματολόγια των επιθεωρήσεων.

```
public String[] strGetQuestionnaires (String[] a_strAuditCodes)
```

Δέχεται ως παράμετρο έναν πίνακα αλφαριθμητικών με τους κωδικούς των επιθεωρήσεων:

```
	//GET QUESTIONNAIRE
	for (int i = 0; i <= a_strAuditCodes.length - 1; i++)
	{
		rs = s.executeQuery("select * from TAUDITS where
		S_AUDITCODE = '"+ a_strAuditCodes[i] +"'");
		while(rs.next())
		{
			l_strResults[i] =
			rs.getString("XML_QUESTIONNAIRE");
		}
```

```
        }

        return l_strResults;
```

Η επόμενη Web Method καλείται από το Delta Client για να στείλουν οι επιθεωρητές τις αναφορές των επιθεωρήσεων στην εταιρεία συμβούλων. `public String strSendReports (String a_strAuditCode, String a_strXMLReport, String a_strAuditDate)`
Δέχεται ως παράμετρο τον κωδικό της επιθεώρησης, την αναφορά ως αλφαριθμητικό σε κρυπτογραφημένη μορφή (itps αρχείο) και την ημερομηνία επιθεώρησης. Η τελευταία διαβάζεται από το όνομα του αρχείο. Αυτό συμβαίνει γιατί όταν ο επιθεωρητής λήγει μια επιθεώρηση, στο όνομα του itps αρχείου που δημιουργείται και συγκρατεί τα αποτελέσματα της επιθεώρησης, αναγράφεται και η ημερομηνία διεξαγωγής της.

Αρχικά, ανανεώνεται το πεδίο που συγκρατεί την ημερομηνία επιθεώρησης.

```
        int l_intResult = s.executeUpdate("update TAUDITS set
        D_AUDITDATE = to_date('" + a_strAuditDate + "','DD/MM/YYYY'),
        I_AUDITREPCOUNT = " + l_intAuditRepCount + " where S_AUDITCODE
        = '"+ a_strAuditCode +"'");
```

Στη συνέχεια για την συγκεκριμένη επιθεώρηση, εισάγεται η αναφορά της στο ανάλογο πεδίο.

```
        int l_intResult2 = s.executeUpdate("update TAUDITS set
        XML_REPORT = concat(XML_REPORT,'" + a_strXMLReport + "') where
        S_AUDITCODE = '"+ a_strAuditCode +"'");
```

Αν όλα πάνε καλά επιστρέφεται το αλφαριθμητικό «ok», το οποίο και ελέγχεται από την εφαρμογή του επιθεωρητή και τον ενημερώνει ανάλογα.

```
        return "ok";
```

Η μοναδική Web Method που δεν καλείται από το Delta Client project είναι η `public String[] strCheckCustomers (String a_strCustomerPwd)` η οποία και καλείται από το Delta Web.
Δέχεται ως παράμετρο τον κωδικό του πελάτη και επιστρέφει έναν πίνακα αλφαριθμητικών με τις αναφορές των επιθεωρήσεων των υποκαταστημάτων και των προμηθευτών του πελάτη:

Με βάση τον κωδικό του πελάτη αναζητούνται αρχικά όλα τα υποκαταστήματα του.

```
        //LOOK FOR CUSTOMER BY PASSWORD
        rs = s.executeQuery("select * from TCUSTOMERS where
        S_CUSTOMERPWD = '"+ a_strCustomerPwd +"'");
        while(rs.next())
        {
              l_intCustomerID = rs.getInt("PKI_CUSTOMERID");
        }
        rs2 = s2.executeQuery("select * from TBRANCHES where
        FKI_CUSTOMERID = "+ l_intCustomerID);
        while(rs2.next())
        {
              l_intBranchID = rs2.getInt("PKI_BRANCHID");
```

Συλλέγονται οι κωδικοί των επιθεωρήσεων οι οποίες διεξήχθηκαν στα υποκαταστήματα του πελάτη.

```
rs3 = s3.executeQuery("select * from TAUDITS where
FKI_BRANCHID = "+ l_intBranchID);
while(rs3.next())
{
      if (l_strResult == "")
      {
            l_strResult += rs3.getString("S_AUDITCODE");
      }
      else
      {
            l_strResult += ";" +
            rs3.getString("S_AUDITCODE");
      }
}
```

Για το κάθε υποκατάστημα, από τις προμήθειές του, επιλέγονται οι κωδικοί των προμηθευτών που προμηθεύουν το κατάστημα.

```
rs4 = s4.executeQuery("select * from TBRANCHSUPPLIER
where FKI_BRANCHID = "+ l_intBranchID);
while(rs4.next())
{
```

Για τον κάθε προμηθευτή συλλέγονται οι κωδικοί των επιθεωρήσεων που συμμετείχαν.

```
      l_intSupplierID = rs4.getInt("FKI_SUPPLIERID");
      rs5 = s5.executeQuery("select * from TAUDITS where
      FKI_SUPPLIERID = "+ l_intSupplierID);
      while(rs5.next())
      {
            if (l_strResult == "")
            {
                  l_strResult +=
                  rs5.getString("S_AUDITCODE");
            }
            else
            {
                  l_strResult += ";" +
                  rs5.getString("S_AUDITCODE");
            }
      }
}
```

Δημιουργείται ο πίνακας με πλήθος το πλήθος των κωδικών επιθεωρήσεων. Για τον κάθε κωδικό επιθεώρησης εισάγεται στον πίνακα η αναφορά της επιθεώρησης.

```
if (l_strResult.split(";").length > 0)
{
      String[] l_strAuditCodes = l_strResult.split(";");
      l_strResults = new String[l_strResult.split(";").length];
      for(int i=0; i<=l_strAuditCodes.length - 1; i++)
      {
            rs6 = s6.executeQuery("select * from TAUDITS where
            S_AUDITCODE = '"+ l_strAuditCodes[i] + "'");
            while(rs6.next())
            {
                  l_strResults[i] =
                  rs6.getString("XML_REPORT");
            }
      }
}
```

```
return l_strResults;
```

Τέλος, επιστρέφεται στο Delta Web ο πίνακας με τις αναφορές. Το Delta Web πλέον είναι υπεύθυνο να ανακτήσει τα αποτελέσματα των επιθεωρήσεων και να τα προβάλει στο χάρτη μέσω του Google Maps API, όπως θα δούμε στη συνέχεια (ενότητα 6.5).

6.5 Delta Web

Το project αποτελείται μόνο από μία aspx σελίδα. Αυτό επιτυγχάνεται μέσω των δυνατοτήτων που παρέχει το Ajax Framework (ASP.NET Ajax). Σε διαφορετική περίπτωση θα έπρεπε να είχαμε συνεχή ανταλλαγή δεδομένων με τον Server, ενώ τώρα η εφαρμογή εκτελείται στην πλευρά του πελάτη (client). Επίσης, υπάρχει αναφορά (reference) στο Delta Service για τη λήψη των αποτελεσμάτων των επιθεωρήσεων του πελάτη μέσω των μεθόδων του Web Service, οι οποίες και αναλύθηκαν στην ενότητα 6.4. Με τη λήψη των αποτελεσμάτων (αναφορών) των επιθεωρήσεων, γίνεται η αποκρυπτογράφησή τους μέσω του Delta Library (ενότητα 6.1). Είναι γραμμένο σε ASP.NET και C# σε .NET Framework 3.5 SP1 και έχει αναπτυχθεί με χρήση του Visual Studio 2008 SP1 IDE. Το web project φιλοξενείται σε Microsoft Internet Information Services 7.5 web server σε λειτουργικό σύστημα Windows Server 2008 R2.

Στον κώδικα ASP.NET της σελίδας (index.aspx) η λειτουργικότητα είναι σε JavaScript script το οποίο είναι εντός του head html tag.
Η συνάρτηση που καλείται για να ζωγραφιστεί ο χάρτης είναι η load(). Αρχικά εισάγουμε τις συντεταγμένες στις οποίες θα είναι κεντραρισμένος ο χάρτης.

```
var map = new GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(38.214495, 22.941861), 9);
```

Τα αποτελέσματα των επιθεωρήσεων του πελάτη, διαβάζονται από τις αναφορές και εκχωρούνται σε κρυφό πεδίο εντός της σελίδας.

```
<asp:HiddenField ID="hidXML" runat="server" />
```

Η τιμή αυτού του πεδίου θα εκχωρηθεί σε μεταβλητή της μεθόδου load() και στη συνέχεια θα μετατραπεί σε XML object.

```
var xmlDoc;
var text = '<%= hidXML.Value %>';
if (window.DOMParser)
{
      parser = new DOMParser();
      xmlDoc = parser.parseFromString(text, "text/xml");
}
else // Internet Explorer
{
      xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
      xmlDoc.async = "false";
      xmlDoc.loadXML(text);
}
```

Τέλος, καλείται η συνάρτηση του Google Maps API GDownloadUrl() [26] στην οποία ως παραμέτρους δέχεται το XML. Είναι η συνάρτηση, η οποία είναι υπεύθυνη να χρωματίσει με τα κατάλληλα χρώματα τις καρφίτσες (pins) του χάρτη. Η κάθε μία

υποδηλώνει και μία επιθεώρηση η οποία πραγματοποιήθηκε σε υποκατάστημα ή σε προμηθευτή του πελάτη.

Στο body της σελίδας στο event onunload υπάρχει η μέθοδος GUnload(). Αυτή αποτελεί ενσωματωμένη μέθοδο του Google Maps API και η ύπαρξη της είναι για να αποφευχθούν διαρροές μνήμης (memory leaks) [26].

```
<body onunload="GUnload()">
```

Στον κώδικα C# της σελίδας αρχικά δημιουργείται ένα αντικείμενο τύπου Delta Service μέσω του οποίου θα επιστραφούν οι αναφορές του πελάτη, αν υπάρχουν. Ως παράμετρο δέχεται τον κωδικό πελάτη που εισήχθηκε από το χρήστη. Ο κώδικας είναι εντός του event του κουμπιού «Προβολή» και καλείται όταν πατηθεί από το χρήστη.

```
string[] l_strAudits;
string l_strMap = "<markers>";
using (DeltaClientService l_ws = new DeltaClientService())
{
      l_strAudits =
      l_ws.strCheckCustomers(this.txtCustomerPWD.Text);
```

Το Web Service επιστρέφει έναν πίνακα αλφαριθμητικών (string array) με τις αναφορές. Για όλες τις αναφορές διαβάζονται τα αποτελέσματα αφού αποκρυπτογραφηθούν με χρήση της μεθόδου Decrypt του Delta Library.

```
for (int i = 0; i <= l_strAudits.Length - 1; i++)
{
      l_xml.LoadXml(l_clDecryption.Decrypt(l_strAudits[i],
      System.IO.Path.GetFullPath(Environment.GetFolderPath(
      Environment.SpecialFolder.InternetCache))));
```

Κατόπιν, γίνεται έλεγχος για το αποτέλεσμα (σημαία) της επιθεώρησης.

```
//CHECK FOR RED OR BLUE FLAGS
XmlNodeList l_xmlNodeListSections =
l_xml.GetElementsByTagName("SECTION");
foreach (XmlNode l_xmlNodeSection in l_xmlNodeListSections)
{
      XmlElement l_xmlElementSection =
      (XmlElement)l_xmlNodeSection;
      if(l_xmlElementSection.GetElementsByTagName(
      "FLAG")[0].InnerText.ToString() == "KOKKINH")
      {
            l_intRedFlags++;
            if (l_strRedFlags == "")
            {
                  l_strRedFlags =
                  l_xmlElementSection.GetElementsByTagName("NAM
                  E")[0].InnerText;
            }
            else
            {
                  l_strRedFlags += ";" +
                  l_xmlElementSection.GetElementsByTagName("NAM
                  E")[0].InnerText;
            }
      }
      else if(l_xmlElementSection.GetElementsByTagName(
      "FLAG")[0].InnerText.ToString() == "ΜΠΛΕ")
      {
            l_intBlueFlags++;
```

```
                    if (l_strBlueFlags == "")
                    {
                          l_strBlueFlags =
                          l_xmlElementSection.GetElementsByTagName("NAM
                          E")[0].InnerText;
                    }
                    else
                    {
                          l_strBlueFlags += ";" +
                          l_xmlElementSection.GetElementsByTagName("NAM
                          E")[0].InnerText;
                    }
              }
        }
```

Όπως αναλύθηκε και στο κεφάλαιο 2, μία επιθεώρηση η οποία έχει τουλάχιστον έναν κόκκινο τομέα θεωρείτε ως κόκκινη. Αντίστοιχα χαρακτηρίζεται ως μπλε και μόνο όταν όλοι οι τομείς έχουν πράσινη σημαία θεωρείτε χωρίς προβλήματα. Ανάλογα το πλήθος των χρωμάτων των τομέων, εισάγεται στη μεταβλητή όπου κρατά το XML των αποτελεσμάτων, τα δεδομένα που χρειάζονται για να προβληθούν στο χάρτη.

```
if (l_intRedFlags > 0)
{
      l_strMap += "<marker name=\"" + l_strheader + "\"
      address=\"" + l_strRedFlags.Replace(";", ", ") + "\"
      lat=\"" + l_strLATITUDE + "\" lng=\"" + l_strLONGITUDE +
      "\" />";
}
```

Σε περίπτωση που για τον συγκεκριμένο πελάτη δεν υπάρχουν αναφορές εμφανίζεται σχετικό μήνυμα.

```
else
{
      this.txtErrorMsg.Visible = true;
      return;
}
```

Με τη ολοκλήρωση της διαδικασίας το XML των αποτελεσμάτων εκχωρείτε στην κρυφή μεταβλητή της ASP.NET σελίδας, που είδαμε παραπάνω. Έπειτα, καλείται η JavaScript μέθοδος load() μέσω της οποίας τα αποτελέσματα που συγκρατεί το XML ζωγραφίζονται στο χάρτη.

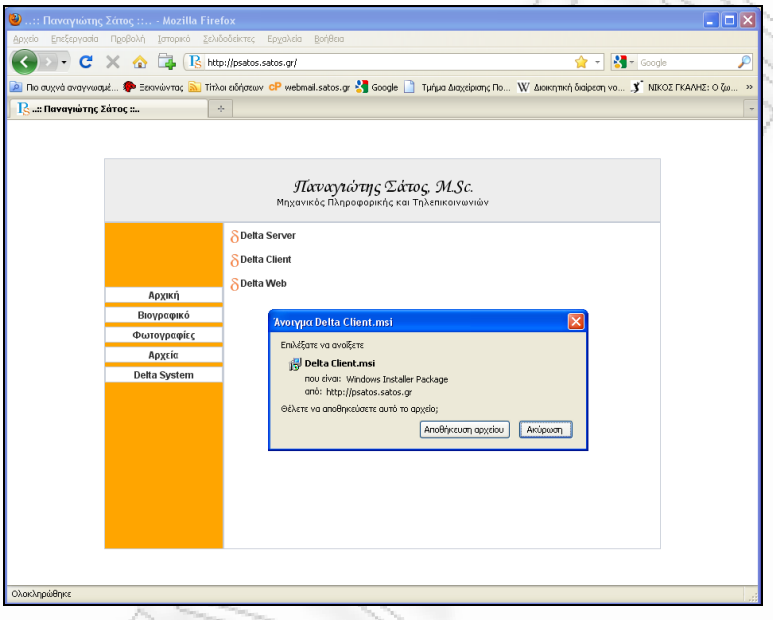
```
hidXML.Value = l_strMap;
if (!ClientScript.IsStartupScriptRegistered("alert"))
{
      Page.ClientScript.RegisterStartupScript
      (this.GetType(), "alert", "load();", true);
}
```

7 ΛΕΙΤΟΥΡΓΙΑ ΣΥΣΤΗΜΑΤΟΣ

Για τη λειτουργία των εφαρμογών του συστήματος απαιτείται η εγκατάστασή τους. Η εφαρμογή διαχείρισης του συστήματος καθώς και η εφαρμογή του επιθεωρητή είναι διαθέσιμες από ιστοσελίδα (http://psatos.satos.gr/) που φιλοξενείται σε Apache HTTP Server 2.2.15 Web Server.

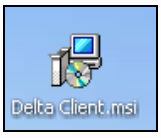
DELTA CLIENT

Η εγκατάσταση της εφαρμογής του επιθεωρητή επιτυγχάνεται μέσω Windows Installer (αρχείο msi), το οποίο οι επιθεωρητές μπορούν να το κατεβάσουν από τον παραπάνω σύνδεσμο (Εικόνα 7.1).



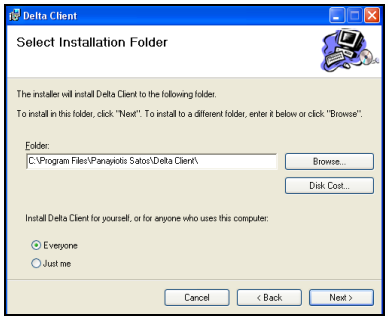
Εικόνα 7.1 Λήψη του εκτελέσιμου αρχείου (εγκαταστάτη) της εφαρμογής Delta Client.

Αφού ολοκληρωθεί η λήψη του αρχείου (Εικόνα 7.2), ο επιθεωρητής το εκτελεί ώστε να ξεκινήσει η διαδικασία εγκατάστασης της εφαρμογής.



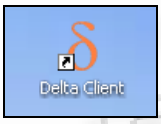
Εικόνα 7.2 Το εικονίδιο του εκτελέσιμου αρχείου της εφαρμογής Delta Client.

Με την εκτέλεση του αρχείου θα προβληθεί ο οδηγός εγκατάστασης (Εικόνα 7.3), μέσω του οποίου η εφαρμογή εγκαθίσταται τοπικά στον υπολογιστή του επιθεωρητή. Η εγκατάσταση γίνεται απλά σε τέσσερα βήματα χωρίς να απαιτείται οποιαδήποτε είδους παραμετροποίηση από τον επιθεωρητή.

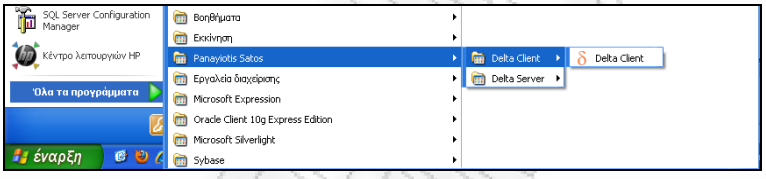


Εικόνα 7.3 Οδηγός εγκατάστασης (wizard) της εφαρμογής Delta Client.

Στην επιφάνεια εργασίας θα δημιουργηθεί το εικονίδιο της εφαρμογής (Εικόνα 7.4), όπως επίσης και στα προγράμματα του λειτουργικού συστήματος (Εικόνα 7.5).

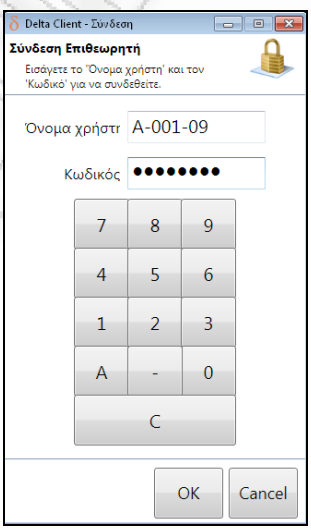


Εικόνα 7.4 Εικονίδιο της εφαρμογής Delta Client.



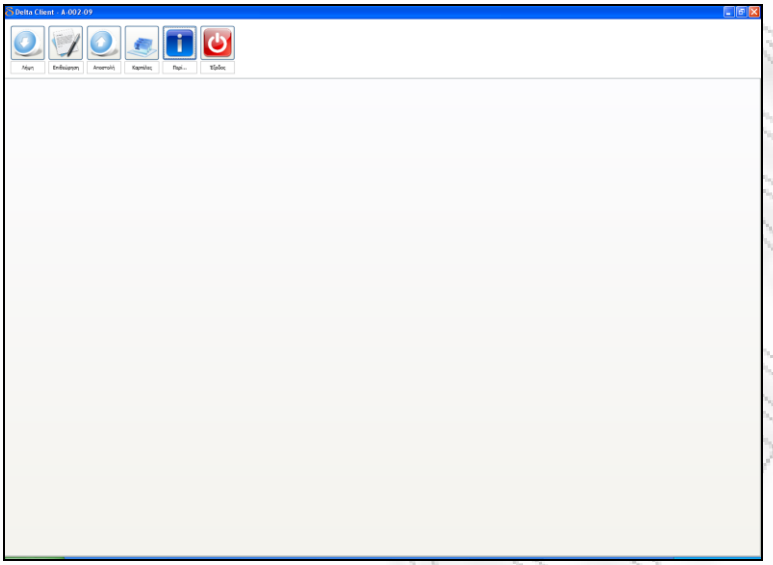
Εικόνα 7.5 Φακελοδομή της εφαρμογής Delta Client,
στα προγράμματα του λειτουργικού.

Μόλις ο επιθεωρητής εκκινήσει την εφαρμογή, προβάλλεται το παράθυρο σύνδεσης (Εικόνα 7.6).



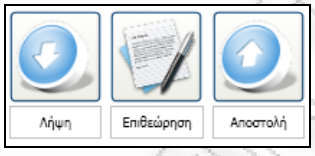
Εικόνα 7.6 Παράθυρο σύνδεσης.

Με την επιτυχή σύνδεση του επιθεωρητή προβάλλεται το κύριο παράθυρο της εφαρμογής (Εικόνα 7.7).



Εικόνα 7.7 Κύριο παράθυρο.

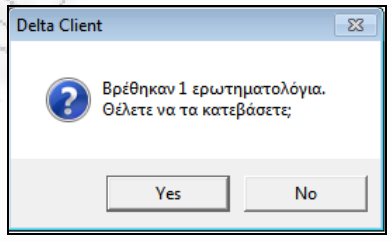
Η εφαρμογή περιλαμβάνει έξι κουμπιά. Τα τρία πρώτα είναι κουμπιά λειτουργιών (Εικόνα 7.8), ενώ τα επόμενα τρία είναι βοηθητικά (Εικόνα 7.9). Τα βοηθητικά κουμπιά παρέχουν λειτουργικότητα που αφορά τη λειτουργία της εφαρμογής και όχι τη λειτουργία του συστήματος.



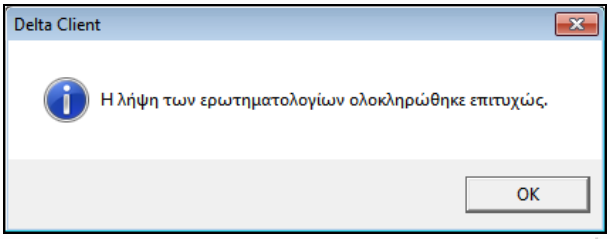
Εικόνα 7.8 Κουμπιά λειτουργιών.

Εικόνα 7.9 Βοηθητικά κουμπιά.

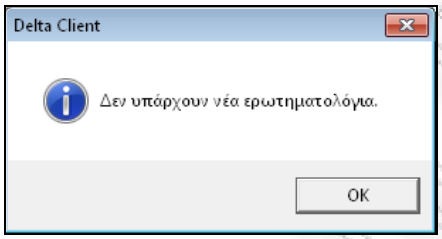
Με το πάτημα του πρώτου κουμπιού ο επιθεωρητής κατεβάζει τα ανατεθειμένα σε αυτόν ερωτηματολόγια (Εικόνες 7.10 & 7.11). Σε περίπτωση που δεν υπάρχουν ή τα έχει ήδη κατεβάσει, ενημερώνεται με μήνυμα (Εικόνα 7.12).



Εικόνα 7.10 Λήψη ερωτηματολογίων.

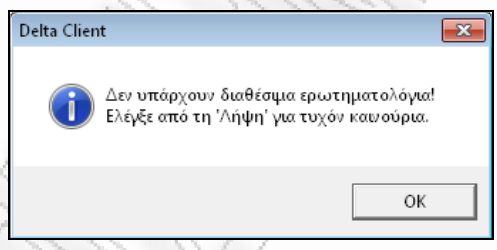


Εικόνα 7.11 Ολοκλήρωση λήψης ερωτηματολογίων.

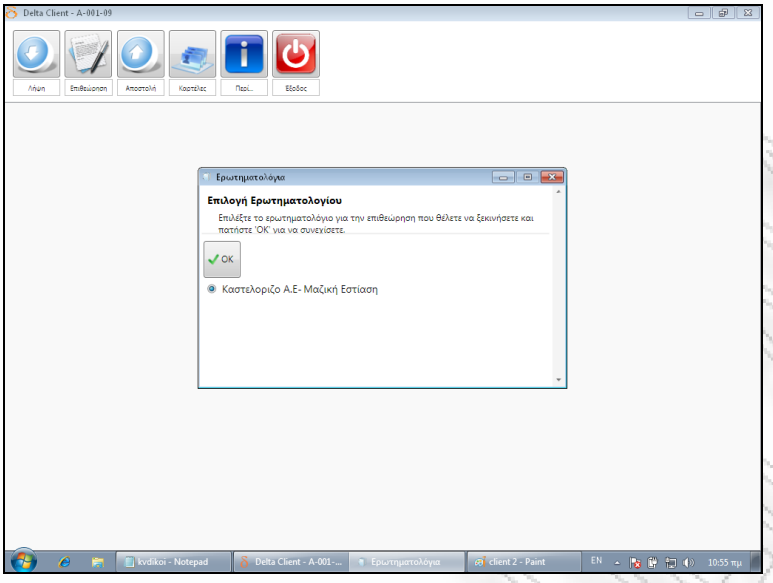


Εικόνα 7.12 Ενημερωτικό μήνυμα στη μη εύρεση ερωτηματολογίων από το κουμπί ‘Λήψη’.

Με το κουμπί ‘Επιθεώρηση’ η εφαρμογή εμφανίζει λίστα με τα ερωτηματολόγια που έχουν ληφθεί (Εικόνα 7.14). Ο επιθεωρητής επιλέγει το ερωτηματολόγιο και πατάει το κουμπί ‘OK’ για να προβληθεί. Σε περίπτωση που δεν υπάρχουν διαθέσιμα ερωτηματολόγια εμφανίζεται ανάλογο μήνυμα (Εικόνα 7.13).

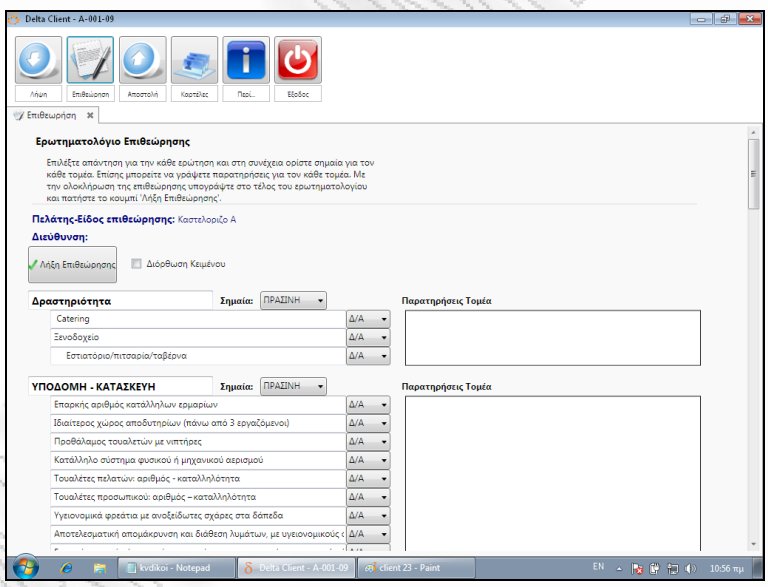


Εικόνα 7.13 Ενημερωτικό μήνυμα στη μη εύρεση ερωτηματολογίων από το κουμπί ‘Επιθεώρηση’.



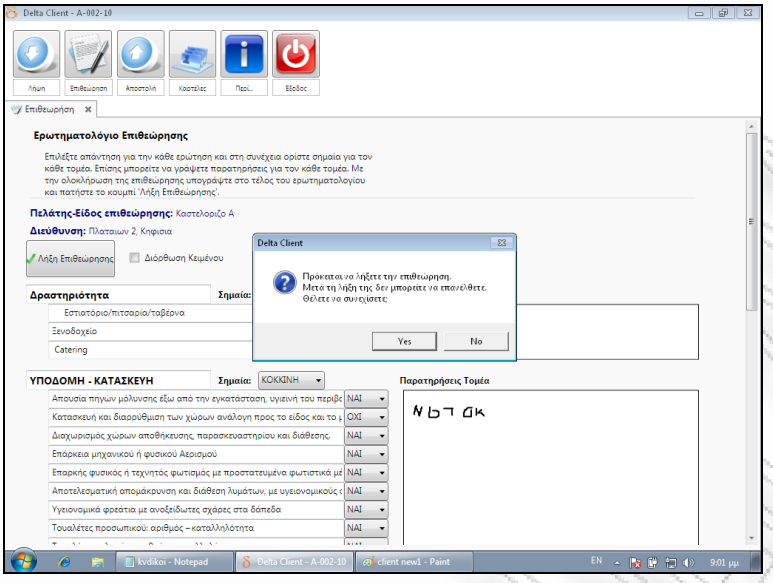
Εικόνα 7.14 Επιλογή ερωτηματολογίου.

Με την προβολή του ερωτηματολογίου, ο επιθεωρητής είναι έτοιμος για τη διεξαγωγή της επιθεώρησης (Εικόνα 7.15). Επιλέγει απαντήσεις για τις ερωτήσεις και ορίζει τη σημαία για τον κάθε τομέα. Για τον κάθε τομέα υπάρχει η δυνατότητα προσθήκης παρατηρήσεων.



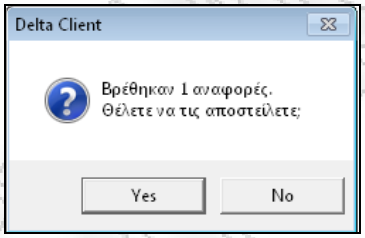
Εικόνα 7.15 Προβολή ερωτηματολογίου.

Με την ολοκλήρωση της επιθεώρησης, ο επιθεωρητής τη λήγει με το πάτημα του κουμπιού ‘Λήξη Επιθεώρησης’ (Εικόνα 7.16).



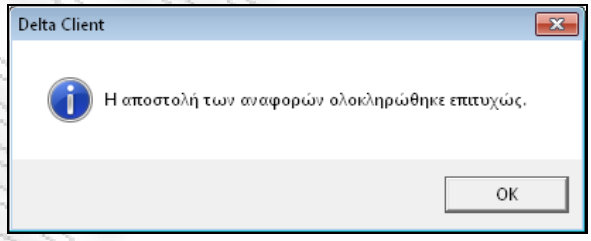
Εικόνα 7.16 Λήξη επιθεώρησης.

Ο επιθεωρητής, για τις επιθεωρήσεις που έχει λήξει, μπορεί να στείλει τις αναφορές στην εταιρεία συμβούλων πατώντας στο κουμπί 'Αποστολή' (Εικόνα 7.17).



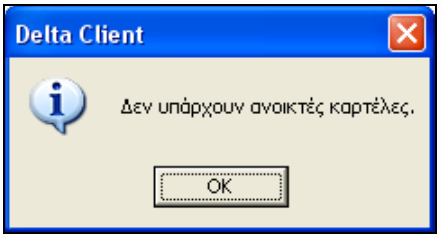
Εικόνα 7.17 Αποστολή αναφοράς επιθεώρησης.

Αν όλα πάνε καλά ενημερώνεται μέσω μηνύματος (Εικόνα 7.18).



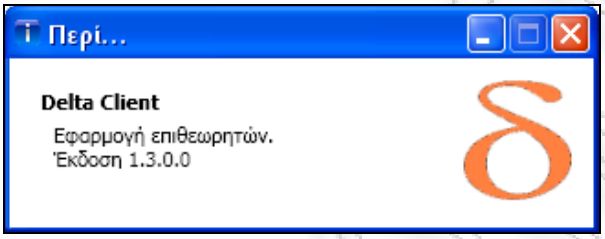
Εικόνα 7.18 Ολοκλήρωση αποστολής αναφοράς.

Το πρώτο βοηθητικό κουμπί κλείνει τις ανοικτές καρτέλες της εφαρμογής, εφ' όσον υπάρχουν. Σε διαφορετική περίπτωση ενημερώνει τον επιθεωρητή με σχετικό μήνυμα (Εικόνα 7.19).



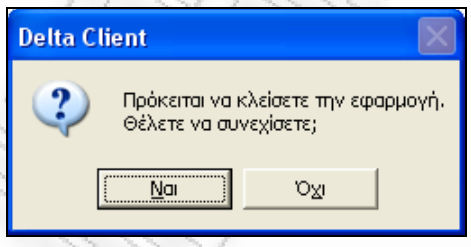
Εικόνα 7.19 Μήνυμα στο πάτημα του κουμπιού Καρτέλες.

Το δεύτερο βοηθητικό κουμπί προβάλει πληροφορίες σχετικά με την έκδοση της εφαρμογής (Εικόνα 7.20).



Εικόνα 7.20 Προβολή της έκδοσης.

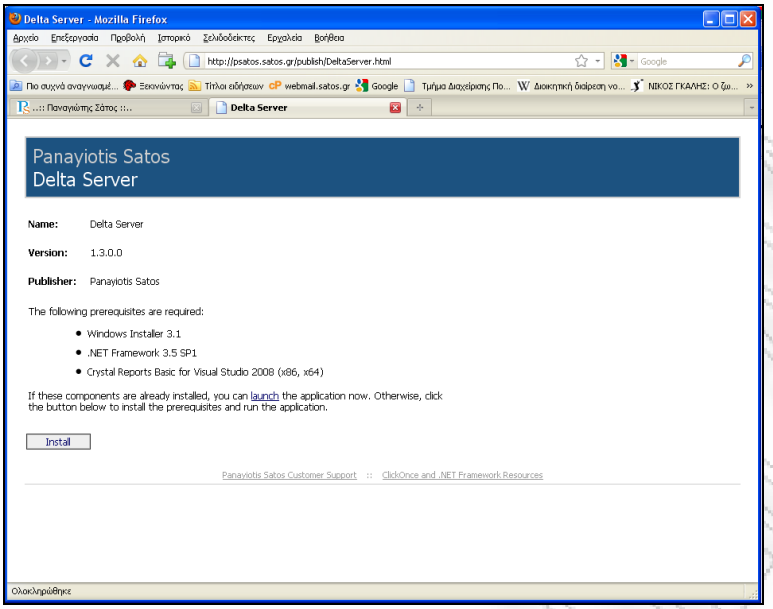
Το τρίτο από τα βοηθητικά κουμπιά είναι το κουμπί για την έξοδο από την εφαρμογή. Αρχικά εμφανίζεται μήνυμα ώστε να επιβεβαιωθεί η ενέργεια από τον επιθεωρητή (Εικόνα 7.21).



Εικόνα 7.21 Μήνυμα επιβεβαίωσης.

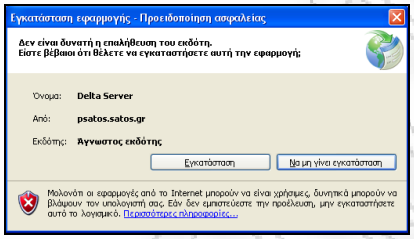
DELTA SERVER

Η εγκατάσταση της εφαρμογής του διαχειριστή επιτυγχάνεται διαδικτυακά (online) μέσω της ιστοσελίδας (http://psatos.satos.gr/) και της τεχνολογίας ClickOnce (Εικόνα 7.22).

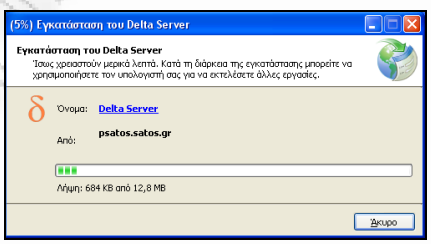


Εικόνα 7.22 Σελίδα εγκατάστασης Delta Server.

Πατώντας το κουμπί 'Install' γίνεται εγκατάσταση όλων των προαπαιτούμενων προγραμμάτων που χρειάζεται η εφαρμογή για να λειτουργήσει. Με την ολοκλήρωση της εγκατάστασή των, ξεκινά η διαδικασία λήψης και εγκατάστασης της εφαρμογής του διαχειριστή (Εικόνες 7.23 & 7.24). Εναλλακτικά, μπορεί να ξεκινήσει αμέσως η λήψη και η εγκατάσταση πατώντας στο σύνδεσμο 'launch' (μόνο στην περίπτωση όπου όλα τα προαπαιτούμενα είναι εγκαταστημένα στον υπολογιστή).



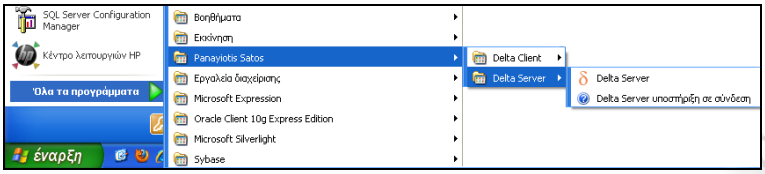
Εικόνα 7.23 Ερώτηση εγκατάστασης.



Εικόνα 7.24 Διαδικασία εγκατάστασης.

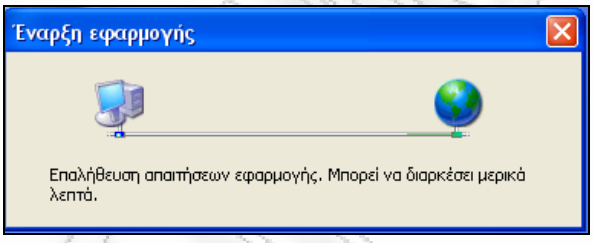
Στην επιφάνεια εργασίας θα δημιουργηθεί το εικονίδιο της εφαρμογής (Εικόνα 7.25), όπως επίσης και στα προγράμματα του λειτουργικού συστήματος (Εικόνα 7.26).

Εικόνα 7.25 Εικονίδιο της εφαρμογής Delta Server.



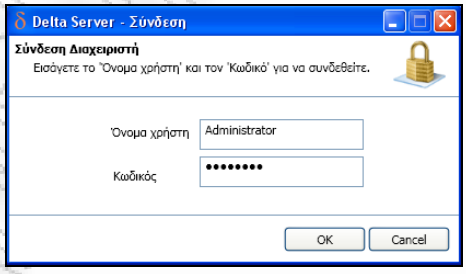
Εικόνα 7.26 Φακελοδομή της εφαρμογής Delta Server,
στα προγράμματα του λειτουργικού.

Μόλις ο διαχειριστής εκκινήσει την εφαρμογή, αρχικά προβάλλεται το παράθυρο έναρξης της εφαρμογής, μέσω του οποίου ελέγχεται η ύπαρξη κάποιας νέας έκδοσης (Εικόνα 7.27). Αυτή η λειτουργικότητα παρέχεται μέσω της τεχνολογίας ClickOnce και δίνει τη δυνατότητα στους υπολογιστές που «τρέχουν» την εφαρμογή να λάβουν αυτόματα τις νέες εκδόσεις. Αυτό επιτυγχάνεται «ανεβάζοντας» τη νέα έκδοση στο web server από όπου ο διαχειριστής εγκατέστησε αρχικά την εφαρμογή στον υπολογιστή του (http://psatos.satos.gr/). Η δυνατότητα αυτόματων ενημερώσεων δεν συμβαίνει στην εφαρμογή του επιθεωρητή μιας και λειτουργεί κυρίως offline. Σε περίπτωση νέας έκδοσης, οι επιθεωρητές θα πρέπει να απεγκαταστήσουν την τρέχουσα έκδοση, να κάνουν λήψη της νέας από την ιστοσελίδα και να ακολουθήσουν το οδηγό εγκατάστασης, όπως περιγράφηκε στην αρχή της τρέχουσας ενότητας.



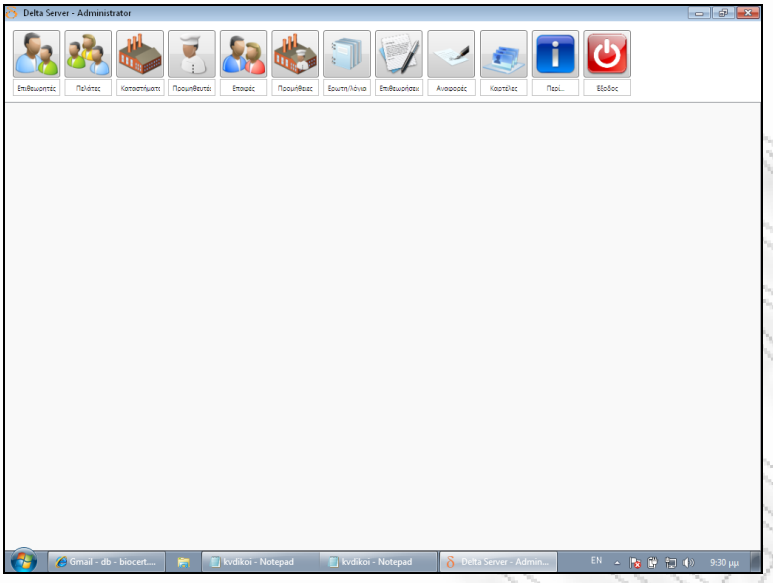
Εικόνα 7.27 Παράθυρο έναρξης της εφαρμογής.

Αν όλα πάνε καλά, εμφανίζεται το παράθυρο σύνδεσης (Εικόνα 7.28).



Εικόνα 7.28 Παράθυρο σύνδεσης.

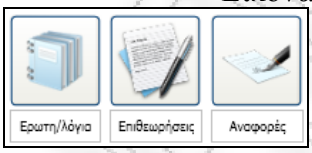
Με την επιτυχή σύνδεση του διαχειριστή προβάλλεται το κύριο παράθυρο της εφαρμογής (Εικόνα 7.29).



Εικόνα 7.29 Κύριο παράθυρο.

Η εφαρμογή περιλαμβάνει δώδεκα κουμπιά τα οποία και χωρίζονται σε τρεις κατηγορίες. Τα κουμπιά διαχείρισης (Εικόνα 7.30), τα κουμπιά των λειτουργιών (Εικόνα 7.31) και τα βοηθητικά (Εικόνα 7.32). Τα πρώτα αφορούν τη διαχείριση των εγγραφών στη βάση ενώ αυτά των λειτουργιών αφορούν τη διαχείριση των επιθεωρήσεων. Τα βοηθητικά κουμπιά παρέχουν λειτουργικότητα που αφορά τη λειτουργία της εφαρμογής και όχι τη λειτουργία του συστήματος.

Εικόνα 7.30 Κουμπιά διαχείρισης.

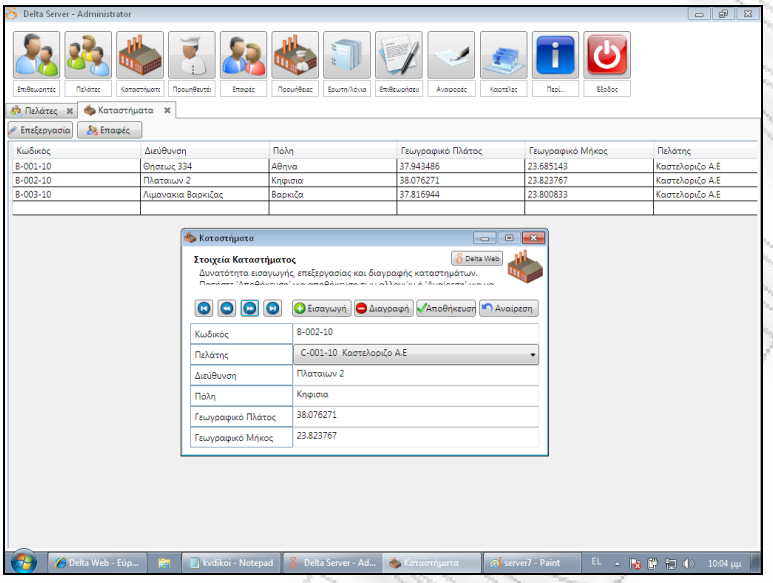


Εικόνα 7.31 Κουμπιά λειτουργιών.

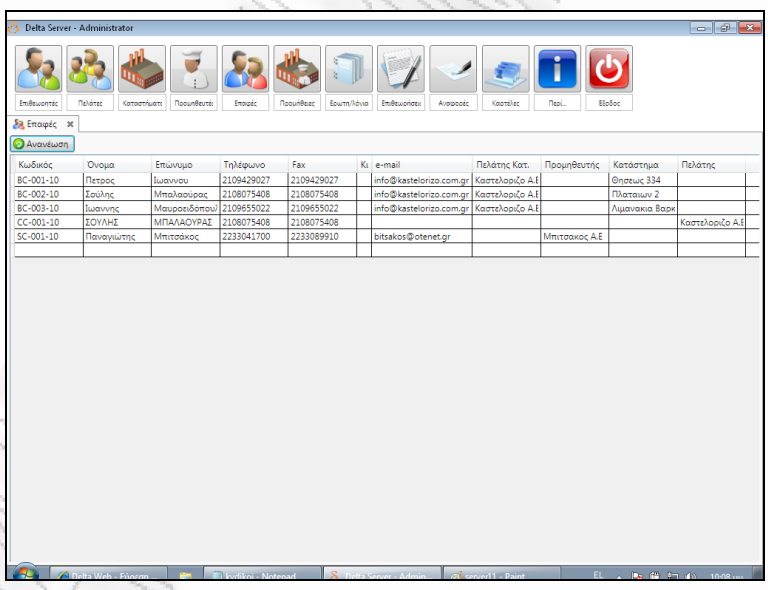
Εικόνα 7.32 Βοηθητικά κουμπιά.

Τα κουμπιά διαχείρισης έχουν όλα την ίδια συμπεριφορά, προβάλλοντας στο πάτημά τους μία φόρμα (Grid) με τις εγγραφές της οντότητας. Στο πάνω μέρος υπάρχει το κουμπί ‘Επεξεργασία’ που στο πάτημά του προβάλει φόρμα για την επεξεργασία των εγγραφών (Εικόνα 7.33). Εξαίρεση αποτελεί η φόρμα των επαφών όπου δεν υπάρχει το κουμπί ‘Επεξεργασία’ αλλά κουμπί ανανέωσης, μιας και η διαχείριση των επαφών γίνεται μέσω των πελατών, καταστημάτων και προμηθευτών (Εικόνα 7.34). Αυτό είναι αναγκαίο γιατί η επαφή είναι οντότητα δευτέρου επιπέδου και πρέπει υποχρεωτικά να ανήκει σε έναν πελάτη, κατάστημα ή προμηθευτή. Στις φόρμες επεξεργασίας των εγγραφών των καταστημάτων και των προμηθευτών υπάρχει στο πάνω δεξιά μέρος το κουμπί ‘Delta Web’ (Εικόνα 7.33). Με το πάτημα του κουμπιού ανοίγει στον περιηγητή (browser) μία σελίδα, στην οποία, εισάγοντας τη διεύθυνση

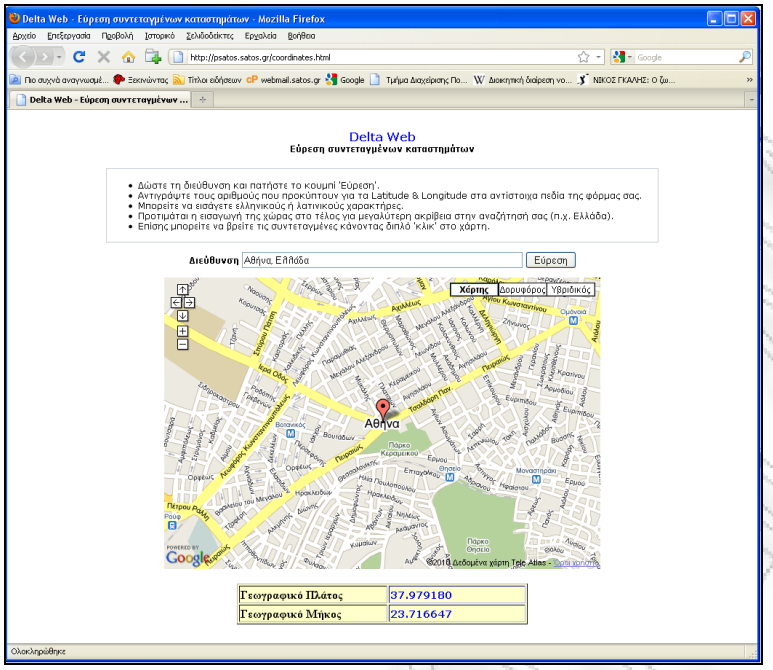
του καταστήματος ή του προμηθευτή, υπολογίζονται οι γεωγραφικές συντεταγμένες (Εικόνα 7.35). Οι τιμές του γεωγραφικού μήκους και πλάτους εισάγονται στη φόρμα της εφαρμογής και αποθηκεύονται στη βάση δεδομένων. Χρησιμοποιούνται για την εύρεση της ακριβής θέσης του καταστήματος και την προβολή στο χάρτη (Google Map) του project Delta Web, η λειτουργία του οποίου αναλύεται στη συνέχεια.



Εικόνα 7.33 Φόρμα επεξεργασίας εγγραφών των καταστημάτων.

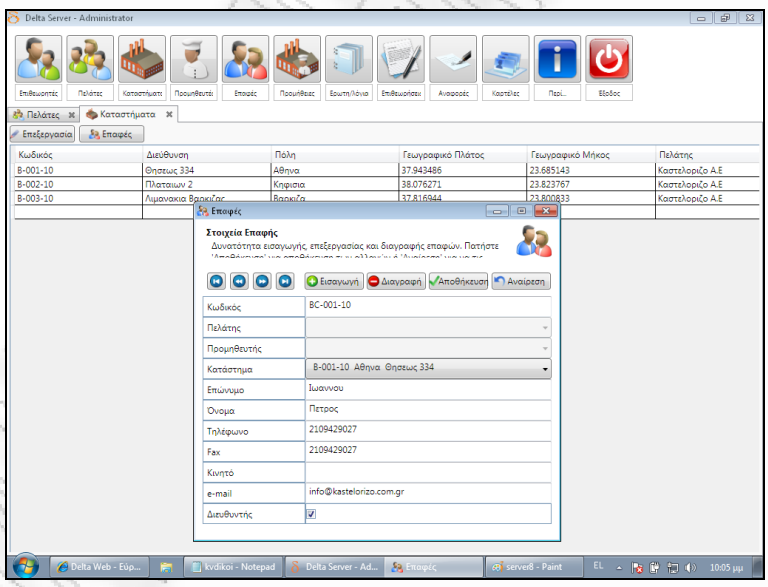


Εικόνα 7.34 Προβολή επαφών.



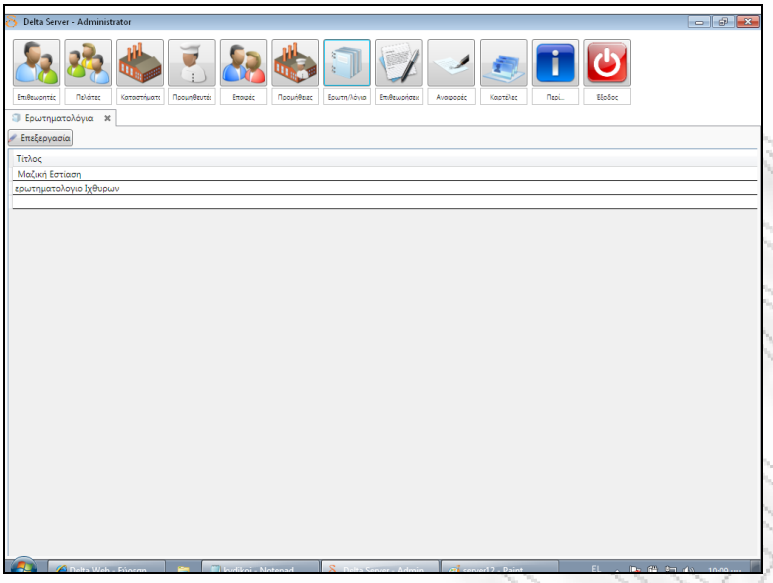
Εικόνα 7.35 Σελίδα εύρεσης συντεταγμένων.

Στις φόρμες των πελατών, καταστημάτων και προμηθευτών εκτός του κουμπιού της επεξεργασίας, υπάρχει το κουμπί ‘Επαφές’ για τη διαχείριση των επαφών (Εικόνα 7.36).

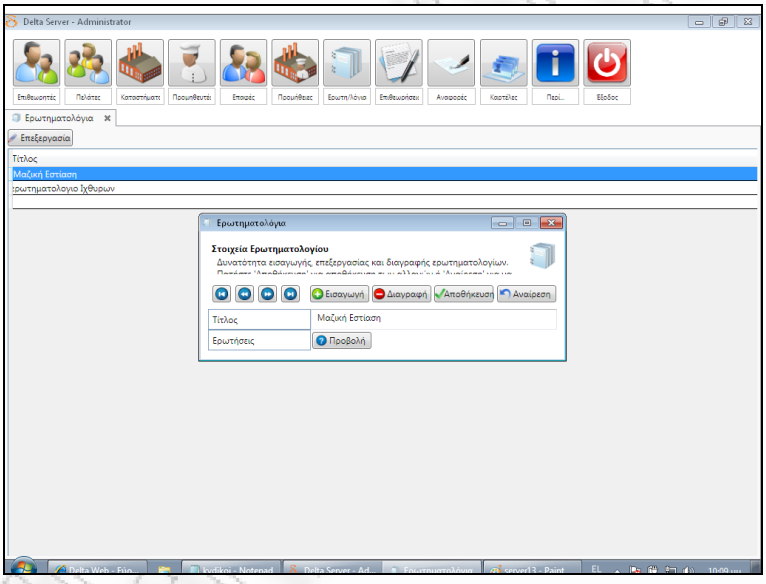


Εικόνα 7.36 Διαχείριση των επαφών των καταστημάτων.

Από τα κουμπιά των λειτουργιών (Εικόνα 7.31), το κουμπί των ‘Ερωτηματολογίων’ προβάλει τα πρότυπα ερωτηματολόγια (Εικόνα 7.37) με δυνατότητα διαχείρισής τους (Εικόνα 7.38).

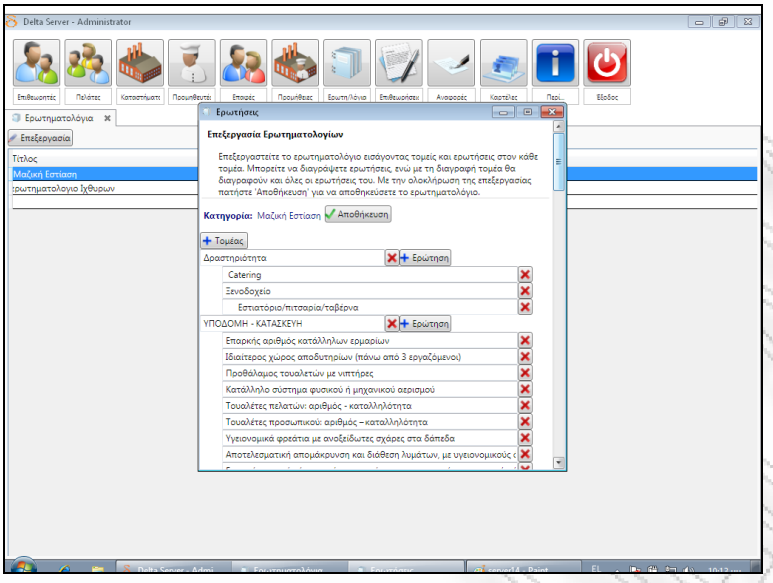


Εικόνα 7.37 Προβολή ερωτηματολογίων.



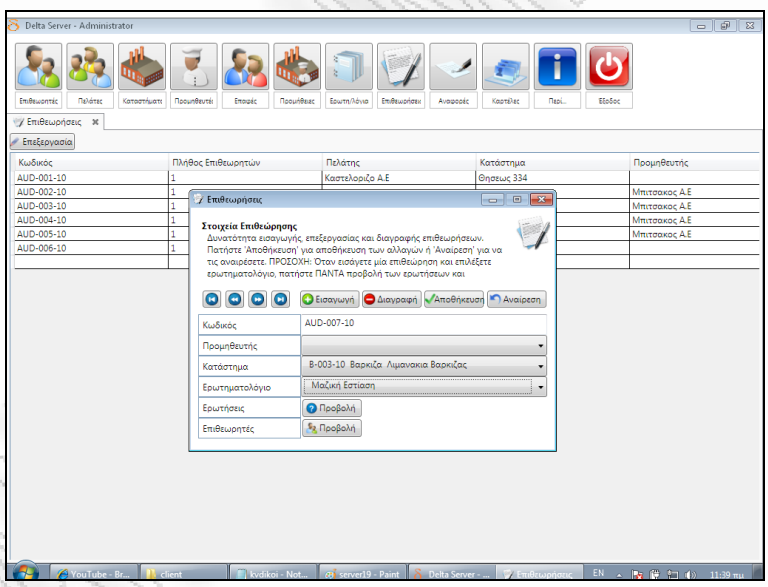
Εικόνα 7.38 Επεξεργασία ερωτηματολογίων.

Όπως αναφέρθηκε και στην ανάλυση (Κεφάλαιο 2) κάθε ερωτηματολόγιο αποτελείται από ερωτήσεις που ανήκουν σε τομείς. Με το κουμπί ‘Προβολή’ ανοίγει η φόρμα που εμφανίζει τους τομείς και τις ερωτήσεις του ερωτηματολογίου (Εικόνα 7.39). Με τη διαγραφή ενός τομέα διαγράφονται και οι ερωτήσεις του.



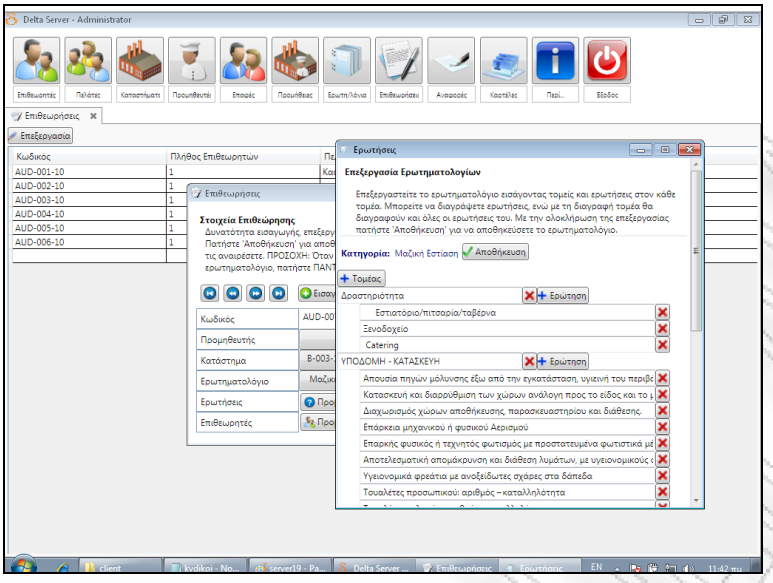
Εικόνα 7.39 Επεξεργασία ερωτήσεων του ερωτηματολογίου.

Με το κουμπί ‘Επιθεωρήσεις’ προβάλλονται οι επιθεωρήσεις. Για τη δημιουργία μιας επιθεώρησης απαιτείται η ύπαρξη τουλάχιστον ενός ερωτηματολογίου. Για την κάθε επιθεώρηση επιλέγεται το κατάστημα ή ο προμηθευτής και το ερωτηματολόγιο (Εικόνα 7.40).



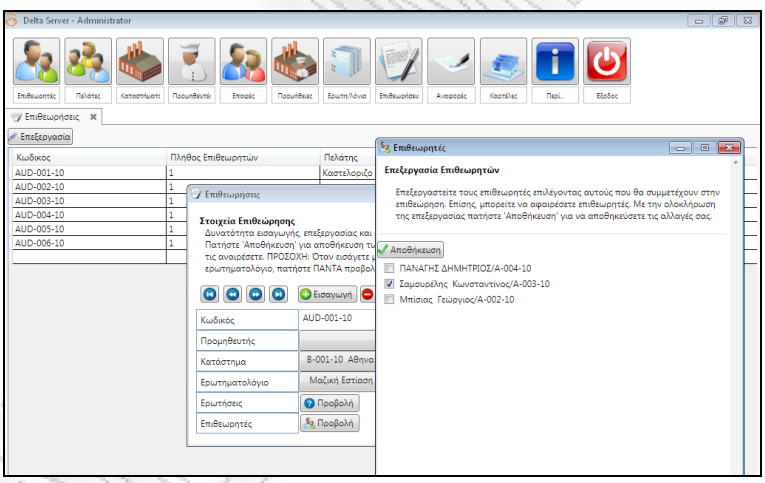
Εικόνα 7.40 Επεξεργασία επιθεώρησης.

Με το κουμπί ‘Προβολή’ δίπλα στην ετικέτα ‘Ερωτήσεις’ προβάλλονται οι ερωτήσεις του ερωτηματολογίου με δυνατότητα προσθαφαίρεσης τομέων και ερωτήσεων (Εικόνα 7.41). Να σημειωθεί ότι το ερωτηματολόγιο που παράγεται για την επιθεώρηση προέρχεται από το αντίστοιχο πρότυπο το οποίο όμως παραμένει ανεξάρτητο. Δηλαδή ακόμη κι αν σβηστεί ή αλλάξει το πρότυπο, το ερωτηματολόγιο της επιθεώρησης δεν θα επηρεαστεί.



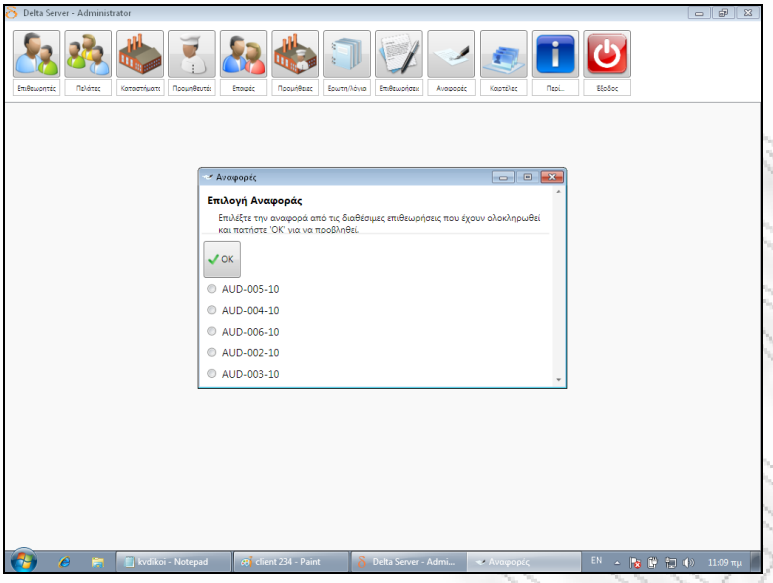
Εικόνα 7.41 Προβολή και επεξεργασία ερωτήσεων,
από ερωτηματολόγιο επιθεώρησης.

Με το κουμπί ‘Προβολή’ δίπλα στην ετικέτα ‘Επιθεωρητές’ προβάλλονται οι επιθεωρητές της εταιρείας συμβούλων με δυνατότητα επιλογής αυτών που θα συμμετέχουν στην επιθεώρηση (Εικόνα 7.42).



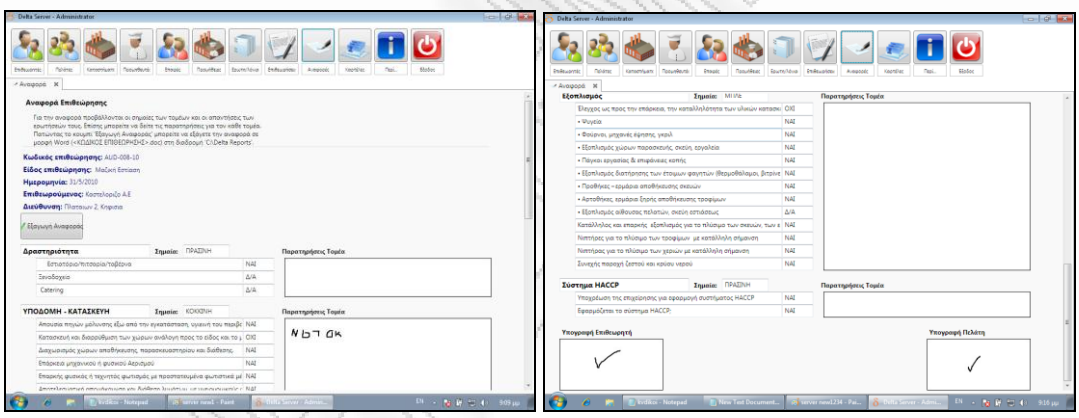
Εικόνα 7.42 Επιλογή επιθεωρητών για την επιθεώρηση.

Με το κουμπί ‘Αναφορές’ ανοίγει η φόρμα επιλογής αναφοράς, από τις αναφορές που έχουν στείλει οι επιθεωρητές (Εικόνα 7.43).



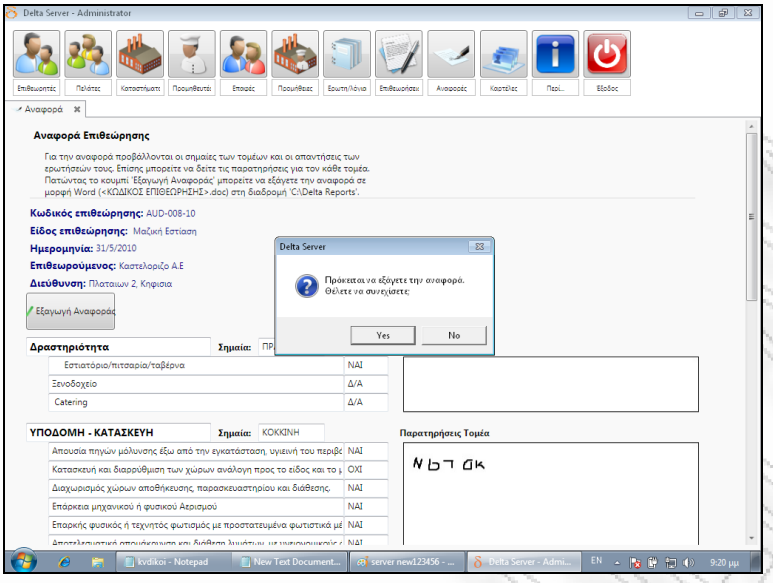
Εικόνα 7.43 Επιλογή αναφοράς επιθεώρησης.

Ο διαχειριστής της εταιρείας συμβούλων επιλέγει μια αναφορά, η οποία και ανοίγει σε ξεχωριστή καρτέλα (Εικόνα 7.44). Ο διαχειριστής εκτός από τις απαντήσεις των ερωτήσεων και τις σημαίες των τομέων, μπορεί να βλέπει και τις παρατηρήσεις που έχει γράψει ο επιθεωρητής.

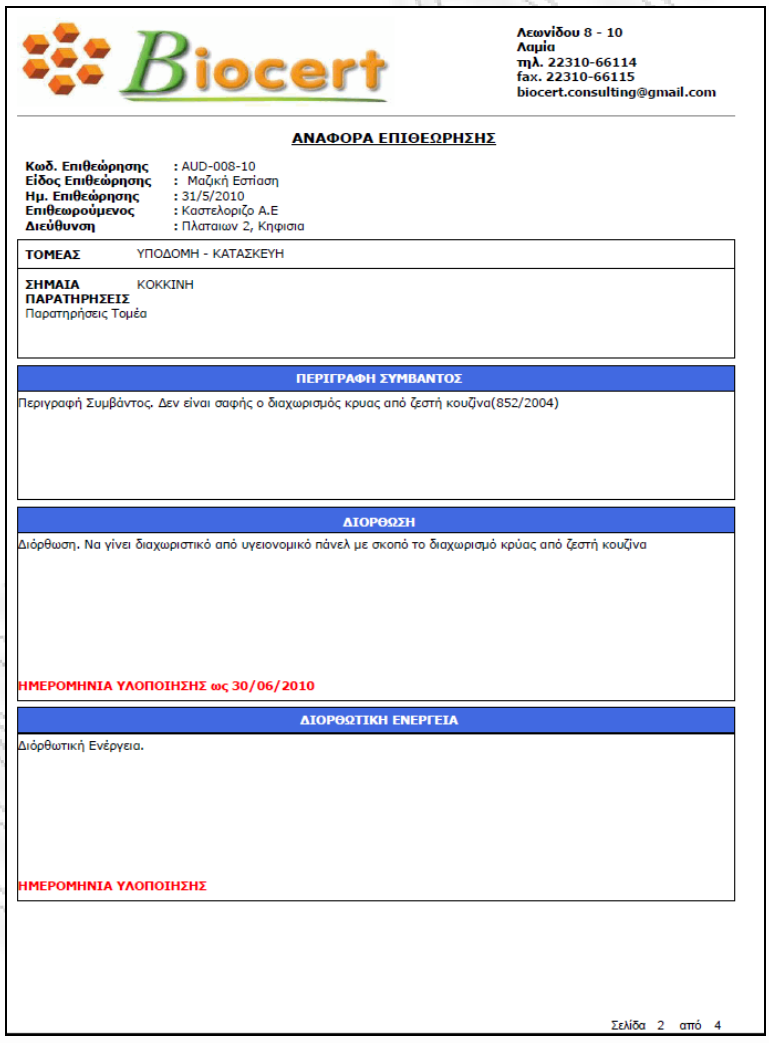


Εικόνα 7.44 Προβολή αναφοράς.

Τέλος, μπορεί να εξάγει την τελική αναφορά της επιθεώρησης (Εικόνα 7.45) σε μορφή κειμένου (doc). Η τελική αναφορά περιέχει τα ονόματα και τις σημαίες των τομέων δίνοντας τη δυνατότητα στο διαχειριστή της εταιρείας συμβούλων να γράψει τις δικές του παρατηρήσεις και τις διορθώσεις που θα πρέπει να προβεί ο πελάτης (Εικόνα 7.46).



Εικόνα 7.45 Εξαγωγή τελικής αναφοράς.

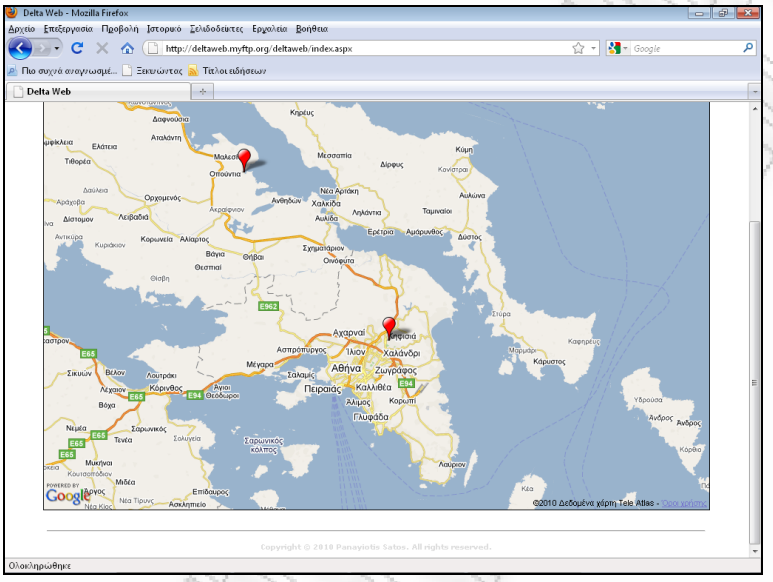


Εικόνα 7.46 Τελική αναφορά σε μορφή doc.

Τα βοηθητικά κουμπιά έχουν τις ίδιες λειτουργίες με τα αντίστοιχα της εφαρμογής του επιθεωρητή, που περιγράφηκαν παραπάνω.

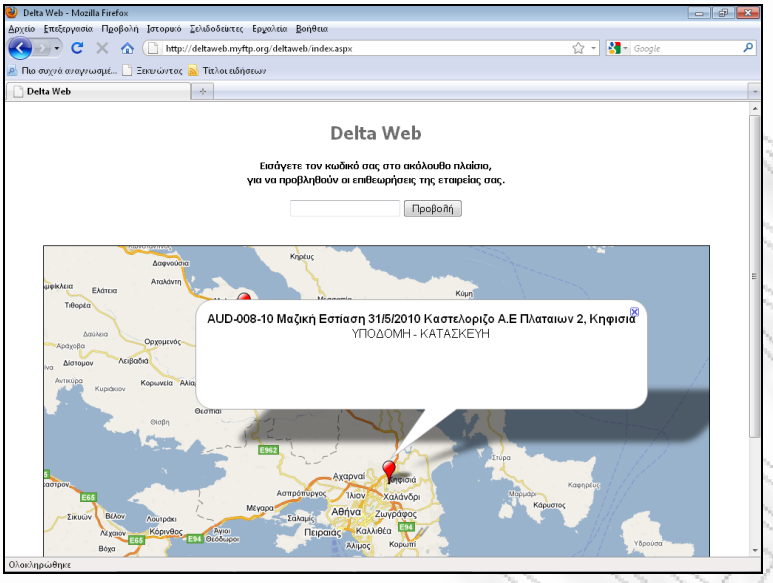
DELTA WEB

Το Delta Web όπως έχουμε αναφέρει είναι ένα web project μέσω του οποίου ο πελάτης ενημερώνεται για τα αποτελέσματα των επιθεωρήσεων. Αρχικά εισάγει τον κωδικό που του έχει αποσταλεί από την εταιρεία συμβούλων. Αν υπάρχουν ολοκληρωμένες επιθεωρήσεις (έχει αποσταλεί η αναφορά από τον επιθεωρητή), τότε για κάθε επιθεώρηση εμφανίζεται μία καρφίτσα (pin) στο χάρτη (Εικόνα 7.47).



Εικόνα 7.47 Προβολή ολοκληρωμένων επιθεωρήσεων.

Πατώντας πάνω στη κάθε καρφίτσα εμφανίζονται με έντονα γράμματα οι πληροφορίες της επιθεώρησης (κωδικός, είδος, ημερομηνία, πελάτης, κατάστημα, διεύθυνση) και από κάτω οι προβληματικοί τομείς στις περιπτώσεις κόκκινης ή μπλε σημαίας (Εικόνα 7.48).



Εικόνα 7.48 Λεπτομέρειες επιθεώρησης.

8 ΑΝΑΤΡΟΦΟΔΟΤΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

Η λειτουργία του συστήματος πραγματοποιήθηκε στην εταιρεία Biocert με έδρα τη Λαμία, για τέσσερις (4) εβδομάδες.
Με την ανατροφοδότηση (feedback) μπόρεσαν να εξαχθούν ασφαλή συμπεράσματα για την αξιολόγηση της αποτελεσματικότητας (λειτουργικό κομμάτι) και της απόδοσης του συστήματος (τεχνικό κομμάτι).

Η αξιολόγηση του συστήματος, σχετικά με την αποτελεσματικότητά του, πραγματοποιήθηκε μέσω συνεντεύξεων με τους εμπλεκόμενους του συστήματος. Συγκεκριμένα την εταιρεία συμβούλων, τους επιθεωρητές και του πελάτες.
Τα θετικά και αρνητικά σημεία που αναφέρθηκαν ήταν:

Αποτελεσματικότητα συστήματος

Πλεονεκτήματα	Μειονεκτήματα
Εταιρεία συμβούλων	
• Χρήση περισσότερων επιθεωρητών. • Γρήγορη απόκριση στον πελάτη. • Μηδενική εκπαίδευση επιθεωρητών. • Μείωση αναλωσίμων (γραφική ύλη). • Άμεση ενημέρωση επιθεωρητή χωρίς άμεση επικοινωνία. • Γρήγορη σύνταξη ερωτηματολογίου για επιθεώρηση. • Γρήγορη και εύκολη ανάθεση επιθεώρησης σε επιθεωρητές. • Αυτόματη εξαγωγή τελικών αναφορών.	• Ανάγκη δύο εξυπηρετητών (Linux & Windows Server). • Παροχή ειδικού (ακριβού) εξοπλισμού (Tablet PC) στους επιθεωρητές.
Επιθεωρητές	
• Εύκολος χειρισμός ερωτηματολογίου. • Απουσία εκτυπώσεων. • Δυνατότητα ηλεκτρονικής αρχειοθέτησης αναφορών. • Άμεση ενημέρωση για αλλαγές σε ερωτηματολόγιο. • Λήψη ερωτηματολογίων και αποστολή αναφορών εν κινήσει. • Γρήγορος έλεγχος για νέες επιθεωρήσεις. • Εύκολη σύνταξη αναφοράς. • Εύκολη διαδικασία αποστολής της αναφοράς.	• Έλλειψη δυνατότητας λήψης νέας έκδοσης ερωτηματολογίου για την ίδια επιθεώρηση.

Πελάτες	
• Άμεση ενημέρωση για τα αποτελέσματα της επιθεώρησης. • Γρήγορη απόκριση σε περιπτώσει αποκλίσεων – μη συμμορφώσεων. • Ηλεκτρονική αρχειοθέτηση αναφορών επιθεώρησης. • Προβολή όλων των επιθεωρήσεων.	• Μη δυνατότητα ενημέρωσης μέσω κινητού, με αποτέλεσμα την ανάγκη συνεχούς παρακολούθησης της ιστοσελίδας.

Απόδοση συστήματος

Στο τεχνικό κομμάτι καταγράφηκαν τα παρακάτω προβλήματα:

- Έλλειψη αυτόματης εξαγωγής αντιγράφου ασφαλείας της βάσης (δεν υποστηρίζεται από την express έκδοση της Oracle DB).
- Προβλήματα συμβατότητας στην εγκατάσταση του Delta Server, λόγω της μη συμβατότητας κάποιων περιηγητών (browsers) με την τεχνολογία ClickOnce (πλην του Microsoft Internet Explorer).
- Υποχρεωτική εγκατάσταση του Oracle Client 10g στον υπολογιστή του διαχειριστή για την επικοινωνία με τη βάση δεδομένων.
- Μη υποστήριξη μεθόδου που χρησιμοποιεί το Google Maps API από τον Microsoft Internet Explorer.

Συμπερασματικά τα πλεονεκτήματα που επέφερε η χρήση του συστήματος, όπως καταγράφηκαν από την εταιρεία Biocert, ήταν:

- Προσέλκυση πελατών με μεγάλο αριθμό υποκαταστημάτων ανά την επικράτεια.
- Χρησιμοποίηση επιθεωρητών που διαμένουν σε διαφορετικές πόλεις από την έδρα της εταιρείας.
- Άμεση γνώση των αποτελεσμάτων των επιθεωρήσεων και ταυτόχρονη αντιμετώπιση τυχόν προβλημάτων.
- Αποφυγή αποστολών αναφορών με ταχυδρομείο ή με άλλο τρόπο.
- Δημιουργία βάσης δεδομένων με τα αποτελέσματα και δυνατότητα στατιστικής ανάλυσης αυτών τόσο στον τελικό πελάτη όσο και στην συνεργαζόμενη εταιρεία συμβούλων.
- Άμεση ενημέρωση επιθεωρητών για αλλαγές σε αναφορά και λοιπά εργαλεία επιθεώρησης.
- Ελαχιστοποίηση του χρόνου ολοκλήρωσης της επιθεώρησης αφού πριν την χρήση του προγράμματος χρειάζονταν 3 ανθρωποημέρες ενώ μετά την χρήση χρειάζεται μια ανθρωποημέρα.
- Καλύτερη διαχείριση επιθεωρητών σε περιπτώσεις όπου σε μια επιθεώρηση παρίστανται περισσότεροι του ενός επιθεωρητή με δυνατότητα να καθορίσει η εταιρεία ποιο κομμάτι επιθεώρησης θα κάνουν οι επιθεωρητές και έτσι αποφεύγεται η επιθεώρηση ιδίων κομματιών και από τους δυο επιθεωρητές και βέβαια η κακή εικόνα της εταιρείας στον πελάτη.
- Αύξηση τζίρου εταιρείας κατά 1,5%
- Μείωση εξόδων κατά 2,5%

Από τα παραπάνω και σε συνδυασμό ότι το σύστημα περιελάμβανε όλες τις αρχικές απαιτήσεις της ανάλυσης εξάγεται το συμπέρασμα ότι αποτέλεσε ένα πετυχημένο πληροφοριακό σύστημα. Σημαντικό είναι το γεγονός ότι τηρήθηκε πλήρως το χρονοδιάγραμμα που είχε οριστεί πριν την έναρξη του κύκλου ζωής. Το τελευταίο αποκτά μεγαλύτερη σημασία λόγω του γεγονότος ότι αναπτύχθηκε ένα πραγματικό σύστημα με πραγματικές απαιτήσεις που εφαρμόστηκε σε αληθινές συνθήκες.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

9 ΣΥΜΠΕΡΑΣΜΑΤΑ

Κατά την διάρκεια ανάπτυξης του συστήματος, μελετήθηκε η ολοκλήρωση ετερογενών κατανεμημένων συστημάτων σε υπηρεσιοστρεφή αρχιτεκτονική. Τα Web Services ως εργαλείο ολοκλήρωσης αποδείχτηκαν στην πράξη πολύ εύχρηστα λόγω της απλής κλήσης τους. Αυτό οφείλεται στο γεγονός ότι η πλευρά που τα καλεί δεν χρειάζεται να ξέρει τίποτα για τη λειτουργία τους παρά μόνο την τεκμηρίωσή (περιγραφή) τους. Δηλαδή τα ορίσματα που δέχεται η μέθοδος, και την επιστροφή της. Επίσης αποδείχτηκαν πολύ γρήγορα στην απόκρισής τους, όπου στις περισσότερες περιπτώσεις ο χρόνος απόκρισής τους ήταν κάτω του ενός δευτερολέπτου. Αυτό αποκτά ιδιαίτερη σημασία αν αναλογιστούμε ότι όλες οι κλήσεις γινόταν μέσω διαδικτύου σε απομακρυσμένο γεωγραφικά Server και όχι σε τοπικό δίκτυο. Η αρχιτεκτονική αποδείχτηκε η πλέον κατάλληλη για την ολοκλήρωση των επιχειρησιακών διαδικασιών, σε ένα περιβάλλον με αρκετούς περιορισμούς και προβλήματα όπως είναι το σενάριο στο οποίο αναπτύχθηκε (επιθεωρήσεις HACCP). Επίσης μελετήθηκε η απόδοση της XML ως μέσο αποθήκευσης (ερωτηματολόγια, αναφορές επιθεώρησης) και επικοινωνίας, φτάνοντας σε υψηλά επίπεδα απόδοσης. Αφ' ενός για τη μεταφορά δομημένης πληροφορίας με δυνατότητα χειρισμού τους από όλες τις πλατφόρμες (Java, .NET) και αφ' ετέρου για την αποφυγή χρήσης βάσης δεδομένων σε κάθε client. Το μεγαλύτερο πλεονέκτημα του τελευταίου είναι η ευκολίας στην αλλαγή της δομής της πληροφορίας, αφού το νέο schema αποτελεί μέρος του αρχείου ενώ στην περίπτωση της βάσης θα έπρεπε να εκτελεστεί SQL Script σε όλες τις βάσεις σε όλους τους client. Η εναλλακτική επιλογή θα ήταν οι clients να επικοινωνούν απευθείας με την κεντρική βάση, κάτι που από τις απαιτήσεις του συστήματος δεν ήταν εφικτό λόγω της υποστήριξης offline λειτουργίας από του clients (πρόβλημα συγχρονισμού δεδομένων). Συγκριτικά με παραδοσιακές τεχνολογίες ολοκλήρωσης (Java RMI, CORBA) η SOA δείχνει να αποτελεί σημείο αναφοράς στην ολοκλήρωση συστημάτων καθιστώντας την XML όχι μόνο μέσο ολοκλήρωσης αλλά και μέσο αποθήκευσης δεδομένων. Παρέχει στο σύστημα μας δυνατότητα μελλοντικών επεκτάσεων χωρίς την απαίτηση αλλαγών στις ήδη υπάρχουσες εφαρμογές. Για παράδειγμα, στην περίπτωση που το σύστημά μας θα ενσωμάτωνε και mobile συσκευές, για τα επιμέρους τμήματα που αποτελούν το σύστημα (Βάση δεδομένων, Web Services, Delta Server, Delta Client) δεν θα ήταν απαραίτητη καμία αλλαγή, αφού η ανάπτυξη ενός client σε mobile περιβάλλον μπορεί να υποστηρίξει κλήση σε Web Services και ασφαλώς διαχείριση XML αρχείων.

ΠΗΓΕΣ

ΞΕΝΟΓΛΩΣΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]. D. Chappell, T. Jewell, Java Web Services. O'Reilly, 2002.

[2]. I. Taylor, From P2P to Web Services and Grids: Peers in a Client / Server World (Computer Communications & Networks). Springer – Verlag London Ltd, 2004.

[3]. E. Vander Veer, R. Mengle, XML® Your visual blueprint for building expert Web pages. maranGraphics Inc., 2000.

[4]. I. Griffiths, C. Sells, Programming Windows Presentation Foundation. O'Reilly, 2005.

[5]. A. Danesh, Mastering Linux Premium Edition. SYBEX Inc., 1999.

[6]. Oracle Database Express Edition. ORACLE, 2006.

ΕΛΛΗΝΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

[7]. Ι. Κάβουρας, Ι. Μήλης, Γ. Ξυλωμένος, Α. Ρουκουνάκη, Κατανεμημένα Συστήματα με Java. Κλειδάριθμος, 2005.

[8]. Γ. Βασιλακόπουλος, Σχεδιασμός Βάσεων Δεδομένων. 2009.

[9]. Σ. Κάτσικας, Σ. Γκρίτζαλης, Δ. Γκρίτζαλης, Ασφάλεια Δικτύων Υπολογιστών. Παπασωτηρίου, 2003.

[10]. Β. Λαοπόδης, Ανάλυση και Σχεδιασμός Συστημάτων. 2001.

[11]. Α. Πλαστήρας, Σ. Καλαφατούδης, Δ. Μπαθούλης, Δημιουργία Ασφαλών Διαδικτυακών Τόπων σε περιβάλλον Linux. Εκδόσεις Νέων Τεχνολογιών, 2006.

[12]. Β. Σταυρουλάκη, Κατανεμημένα Συστήματα, Πανεπιστημιακές Σημειώσεις Π.Μ.Σ. Δικτυοκεντρικών Συστημάτων. Πανεπιστήμιο Πειραιώς, 2009.

ΠΗΓΕΣ ΔΙΑΔΙΚΤΥΟΥ

[13]. Lecture slides and activities on XMLRPC.
Διαθέσιμο από: http://perisic.com/xmlrpc

[14]. XML Schema: Formal Description.
Διαθέσιμο από: http://www.w3.org/TR/2001/WD-xmlschema-formal-20010925

[15]. XML Encryption Requirements. Διαθέσιμο από:
http://www.w3.org/TR/2002/NOTE-xml-encryption-req-20020304

[16]. Web Services Description Requirements.
Διαθέσιμο από: http://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028/

[17]. Web Services Architecture.
Διαθέσιμο από: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[18]. Web Service Management: Service Life Cycle. Διαθέσιμο από:
http://www.w3.org/TR/2004/NOTE-wslc-20040211/

[19]. AJAX Tutorial. Διαθέσιμο από: http://www.w3schools.com/ajax/default.asp

[20]. XML Tutorial. Διαθέσιμο από: http://www.w3schools.com/xml/default.asp

[21]. Web Services Tutorial.
Διαθέσιμο από: http://www.w3schools.com/webservices/default.asp

[22]. SOAP Tutorial. Διαθέσιμο από: http://www.w3schools.com/soap/default.asp

[23]. Very Dynamic Web Interfaces. Διαθέσιμο από:
http://www.xml.com/pub/a/2005/02/09/xml-http-request.html

[24]. Κατανεμημένα Συστήματα.
Διαθέσιμο από: http://epikouros.unipi.gr/eclass/DTPS130/

[25]. Service-oriented architecture. Διαθέσιμο από: http://en.wikipedia.org/wiki/Service-oriented_architecture
[26]. Google Maps API Reference. Διαθέσιμο από: http://code.google.com/intl/el-GR/apis/maps/documentation/javascript/v2/reference.html
[27]. Windows Presentation Foundation. Διαθέσιμο από: http://www.codeproject.com/KB/WPF/
[28]. Βιοτεχνικό Επιμελητήριο Αθηνών. Διαθέσιμο από: http://212.205.122.34/xrisimi/haccp.htm

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ