ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων

Πτυχιακή εργασία

Ολοκληρωμένη Εφαρμογή Διαχείρισης Πολυκατοικιών & Έκδοσης Κοινοχρήστων

Κύρτσης Αθανάσιος Α.Μ. 107

Επιβλέπων καθηγητής: Αγγελάκης Γεώργιος

Ηράκλειο, Ιούνιος 2008

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πρωτίστως τον καθηγητή κ. Αγγελάκη Γεώργιο για την ανάθεση της εργασίας, την στήριξη και συνεργασία που συνέβαλλαν στην ολοκλήρωση της.

Επίσης θα ήθελα να ευχαριστήσω θερμά όλους τους συναδέλφους και φίλους για την στήριξη και βοήθεια που μου παρείχαν.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την ηθική και υλική υποστήριξη που μου παρείχαν κατά την διάρκεια των σπουδών μου.

Πίνακας περιεχομένων

Εισαγωγή

Οι εφαρμογές διαχείρισης αποτελούν ίσως το μεγαλύτερο ποσοστό των εμπορικών εφαρμογών που κυκλοφορούν σήμερα στην αγορά από τις διάφορες εταιρίες πληροφορικής. Σχεδόν κάθε επιχείρηση, όποιο και να είναι το αντικείμενο της, έχει πλέων την δυνατότητα να βρει στην αγορά ένα πλήθος από εφαρμογές οι οποίες θα είναι κατάλληλες για την διαχείρισή της.

Η συγκεκριμένη πτυχιακή εργασία έχει ως σκοπό την μελέτη και την δημιουργία μίας εφαρμογής η οποία θα είναι κατάλληλη για την διαχείριση μιας εταιρίας έκδοσης κοινοχρήστων. Στόχοι της πτυχιακής είναι:

- η ανεύρεση των δυσκολιών που παρουσιάζονται κατά την μελέτη και την ανάπτυξη μιας εμπορικής εφαρμογής
- η δημιουργία μιας ευέλικτης εφαρμογής η οποία να καλύπτει όσο το δυνατών περισσότερες από τις απαιτήσεις και ιδιαιτερότητες των εταιριών έκδοσης κοινοχρήστων
- η δυνατότητα μελλοντικής χρήσης μερών της εφαρμογής σε άλλες εμπορικές εφαρμογές διαχείρισης με διαφορετικό αντικείμενο
- η δυνατότητα βελτίωσης της εφαρμογής χωρίς να χρειαστεί να ξαναγραφτεί μεγάλο μέρος από τον κώδικα

Η εφαρμογή αυτή αναπτύχθηκε στο περιβάλλον του Microsoft Visual Studio 2005 χρησιμοποιώντας ως γλώσσα προγραμματισμού την Visual Basic. Η επιλογή του περιβάλλοντος εργασίας και της γλώσσας προγραμματισμού έγινε με γνώμονα την ταχύτητα και την ευελιξία που μας δίνεται από την Microsoft μέσα από το .Net Framework. Το .Net Framework της Microsoft είναι μια ολοκληρωμένη λύση ανάπτυξής εφαρμογών προσφέροντας στον προγραμματιστή πληθώρα έτοιμων εργαλείων που καλύπτουν τις περισσότερες από τις βασικές απαιτήσεις που μπορεί να έχει μια εφαρμογή. Συγκεκριμένα κατά την ανάπτυξη της εφαρμογής μας χρησιμοποιήσαμε πολλά από αυτά τα εργαλεία, από τα ποιο απλά όπως το Button και το TextBox control του Namespace System.Windows.Forms, έως τα ποιο περίπλοκα και ολοκληρωμένα εργαλεία περιλαμβάνονται στα Namespace System.Data.OleDb, System.Xml και System.Drawing.

Για την δημιουργία της Βάσης Δεδομένων χρησιμοποιήθηκε η Microsoft Access κυρίως για την ευκολία λειτουργίας της και την ταχύτητα με την οποία μπορούμε να δημιουργήσουμε και να μεταβάλουμε μια βάση δεδομένων μέσα από το περιβάλλων της. Εξαιτίας του ότι θα θέλαμε η εφαρμογή μας να μπορεί να επικοινωνήσει και με άλλους τύπους βάσεων δεδομένων έτσι ώστε στο μέλλον να μπορούμε εύκολα να αλλάξουμε την Microsoft Access με κάποια άλλη Open Source λύση δημιουργήσαμε μία κλάση που να

παρεμβάλλεται ανάμεσα στην εφαρμογή και την βάση δεδομένων αποκρύβοντας από την εφαρμογή το είδος της βάσης δεδομένων.

1 – Εμπορικές εφαρμογές

Η επιτυχία μιας εμπορικής εφαρμογής βασίζετε κυρίως στην αρχική μελέτη των απαιτήσεών της. Για να επιτευχθεί κάτι τέτοιο θα πρέπει να υπάρχει σωστή επικοινωνία ανάμεσα στην εταιρία ανάπτυξης της εφαρμογής και στην ενδιαφερόμενη ή ενδιαφερόμενες επιχειρήσεις, αναλόγως εάν η εφαρμογή έχει στόχο μία μεμονωμένη επιχείρηση ή ένα πλήθος επιχειρήσεων οι οποίες ασχολούνται με κάποιο συγκεκριμένο αντικείμενο.

Κάθε εμπορική εφαρμογή, είτε αυτή αφορά μία μεμονωμένη επιχείρηση είτε έχει σαν στόχο ένα πλήθος επιχειρήσεων, θα πρέπει να αναπτυχθεί με τρόπο τέτοιο έτσι ώστε να καλύπτει κάποιους βασικούς στόχους από την πλευρά της εταιρίας που θα την αναπτύξει. Δύο από τους βασικότερους στόχους είναι η εύκολη αναβάθμιση της εφαρμογής και η δυνατότητα επαναχρησιμοποίησης μέρους του κώδικα της εφαρμογής σε άλλες εφαρμογές με διαφορετικό αντικείμενο.

Κατά την μελέτη των απαιτήσεων μιας εφαρμογής επιβάλετε η επικοινωνία με την ενδιαφερόμενη επιχείρηση ή με επιχειρήσεις οι οποίες πραγματεύονται το αντικείμενο της εφαρμογής. Στην περίπτωση που η εφαρμογή προορίζετε για κάποια μεμονωμένη επιχείρηση καλό είναι να υπάρχει και μία ευρεία μελέτη της αγοράς για τις απαιτήσεις που θα μπορούσαν να έχουν άλλες επιχειρήσεις που ασχολούνται με το ίδιο αντικείμενο.

Σε κάθε περίπτωση η επικοινωνία ανάμεσα σε ανθρώπους της πληροφορικής και σε ανθρώπους που δεν έχουν καμία σχέση με αυτήν δεν είναι εύκολη υπόθεση.

1.1.1 – Προβλήματα επικοινωνίας

Συνήθως είναι πολύ δύσκολο να εξηγήσεις και να δώσεις να καταλάβει σε κάποιον τι είναι αυτό που θέλεις και πώς ακριβός το θέλεις εάν αυτός δεν έχει καμία γνώση για το αντικείμενο για το οποίο του μιλάς. Είναι φανερό λοιπόν ότι κάθε επιχείρηση πρόκειται να αντιμετωπίσει σίγουρα κάποια προβλήματα επικοινωνίας μέχρι να εξηγήσει επακριβώς τις απαιτήσεις και τις ιδιαιτερότητες που έχει για την διαχείρισή της.

Αντίστροφα, προβλήματα επικοινωνίας μπορούν να παρουσιαστούν και από την πλευρά της εταιρίας ανάπτυξης της εφαρμογής. Είναι πολύ πιθανών κατά την μελέτη των απαιτήσεων να υπάρξουν κάποιες αποφάσεις που θα πρέπει να παρθούν οι οποίες θα απαιτούν κάποιες γνώσεις πάνω στην πληροφορική.

Ένας τρόπος αντιμέτωπης αυτών των προβλημάτων επικοινωνίας, ίσως ο καλύτερος, είναι η μεταφορά γνώσης ανάμεσα στην εταιρία ανάπτυξης της εφαρμογής και στους ενδιαφερόμενους. Αυτό όμως είναι πολύ δύσκολο να γίνει. Ειδικά στην περίπτωση

που η εφαρμογή δεν αφορά μία μεμονωμένη επιχείρηση αλλά ένα πλήθος επιχειρήσεων είναι αδύνατον να ζητηθεί από κάποια επιχείρηση, η οποία θα είναι ένας πιθανός αγοραστής της εφαρμογής, να εκπαιδεύσει το προσωπικό της πάνω στην πληροφορική και θα πρέπει να βρεθούνε άτομα που να γνωρίζουν κάποια βασικά πράγματα για την πληροφορική μέσα στις επιχειρήσεις αυτές. Αλλά και στην περίπτωση που η εφαρμογή προορίζετε για μία μεμονωμένη επιχείρηση το βάρος της εκμάθησης του αντικειμένου του άλλου πέφτει κυρίως στην εταιρία που θα αναπτύξει την εφαρμογή και όχι στην ενδιαφερόμενη επιχείρηση. Αυτό γίνετε γιατί η ενδιαφερόμενη επιχείρηση είναι ο πελάτης ο οποίος θα πρέπει να εξυπηρετηθεί χωρίς να υποχρεούται, ίσως να μην έχει καν την διάθεση να το κάνει, να μάθει περισσότερα πάνω στην πληροφορική πέρα από τον τρόπο χρήσης της εφαρμογής.

Συμπερασματικά μπορούμε να πούμε ότι η λύση στο πρόβλημα της επικοινωνίας βαραίνει κυρίως την εταιρία ανάπτυξης της εφαρμογής με το να εκπαιδευτούν κατάλληλα οι αναλυτές της επάνω στο αντικείμενο με το οποίο πρόκειται να ασχοληθούν.

1.1.2 – Καθορισμός των απαιτήσεων

Κατά τον καθορισμό των απαιτήσεων μιας εμπορικής εφαρμογής θα πρέπει να ληφθούν υπόψη όσο το δυνατών περισσότερες από τις ιδιαιτερότητες των επιχειρήσεων που θα είναι ο στόχος της εφαρμογής. Είναι σχεδόν βέβαιο ότι κάποιες από τις απαιτήσεις που θα έχει μία επιχείρηση να είναι αδιάφορες για κάποια άλλη. Αυτό δεν θα πρέπει να βάλει στο περιθώριο και να μειώσει την σημαντικότητα αυτών των απαιτήσεων εάν θέλουμε η εφαρμογή να μπορεί να εξυπηρετήσει επαρκώς και σωστά το μεγαλύτερο μέρος της αγοράς. Ακόμα και εάν κάποια απαίτηση προβάλετε από μία και μοναδική επιχείρηση, ίσως αυτή η ιδιαιτερότητα της επιχείρησης αυτής να φανεί χρήσιμη και καινοτόμα και σε άλλες επιχειρήσεις και ίσως να είναι αυτή που θα οδηγήσει την εφαρμογή μας στην επιτυχία.

Από την αντίθετη πλευρά κάποιες από τις απαιτήσεις και ιδιαιτερότητες ίσως να είναι πολύ εξεζητημένες και να μειώνουν την χρηστικότητα της εφαρμογής σε βαθμό που να μην είναι αποδεκτή από άλλες επιχειρήσεις. Επίσης είναι δυνατόν να εμφανιστούν αντικρουόμενες απαιτήσεις που να μην μπορούν να συνυπάρξουν στην ίδια εφαρμογή. Σε αυτές τις περιπτώσεις θα πρέπει να μελετηθεί πάρα πολύ καλά το εάν θα είναι συνετό να ενσωματωθούν στην εφαρμογή και σε ποιο βαθμό καθώς και ποια από τις αντικρουόμενες απαιτήσεις θα επικρατήσει.

Μετά την συγκέντρωση όλων των δυνατών ιδιαιτεροτήτων και απαιτήσεων που θα επηρεάσουν την υλοποίηση και την ανάπτυξη της εφαρμογής θα πρέπει αυτές να μελετηθούν και να διατυπωθούν με τρόπο τέτοιο ώστε να μην υπάρχει αμφιβολία για το νόημά τους. Στην περίπτωση που η εφαρμογή προορίζεται για μία συγκεκριμένη

επιχείρηση, αυτές οι αναδιατυπωμένες απαιτήσεις θα αποτελέσουν στο σύνολό τους το συμβόλαιο που θα υπογραφεί από την εταιρία ανάπτυξης της εφαρμογής και την ενδιαφερόμενη επιχείρηση για την ανάθεση της υλοποίησης της εφαρμογής από την πρώτη.

Επόμενο βήμα μετά την μελέτη των απαιτήσεων μιας εμπορικής εφαρμογής αποτελεί ο σχεδιασμός της. Κατά τον σχεδιασμό μιας εφαρμογής δημιουργείται ο σκελετός πάνω στον οποίο θα αναπτυχθεί η εφαρμογή. Σε αυτό το στάδιο θα πρέπει να παρθούν κάποιες σημαντικές αποφάσεις όπως η γλώσσα στην οποία θα γραφτεί η εφαρμογή, τις απαιτήσεις σε υλικό που θα έχει, το εάν θα χρησιμοποιεί κάποια βάση δεδομένων και το είδος αυτής, το εάν η εφαρμογή θα υποστηρίζει περισσότερους από έναν χρήστες, η δυνατότητα απομακρυσμένης χρήσης της καθώς και άλλα παρόμοια ζητήματα.

Οι περισσότερες από αυτές τις αποφάσεις εξαρτώνται από τις απαιτήσεις της εφαρμογής. Ορισμένες βέβαια από τις αποφάσεις έχουν ήδη παρθεί ή είναι δύσκολο να αλλάξουν, όπως παραδείγματος χάρη η γλώσσα στην οποία θα υλοποιηθεί η εφαρμογή, καθώς συνήθως οι εταιρίες ανάπτυξης εφαρμογών έχουν κάποια καθιερωμένα πρότυπα πάνω στα οποία δουλεύουν.

Όπως προαναφέρθηκε, εκτός από την επιτυχία της εφαρμογής, δύο είναι η βασικότεροι στόχοι κατά την ανάπτυξη μιας εμπορικής εφαρμογής. Η εύκολη αναβάθμισή της και η δυνατότητα επαναχρησιμοποίησης μέρους του κώδικά της σε άλλες εφαρμογές με διαφορετικό αντικείμενο.

1.3.1 – Επαναχρησιμοποιήσιμος κώδικας

Οι περισσότερες εταιρίες ανάπτυξης λογισμικού συνηθίζουν να επαναχρησιμοποιούν κομμάτια κώδικα από προηγούμενες εφαρμογές που έχουν αναπτύξει. Η κλάση διασύνδεσης της εφαρμογής με την βάση δεδομένων της είναι το κυριότερο παράδειγμα επαναχρησιμοποιήσιμου κώδικα. Άλλα παραδείγματα επαναχρησιμοποιήσιμου κώδικα μπορούν να είναι κάποιες κλάσεις διαχείρισης των ρυθμίσεων της εφαρμογής, κλάσεις διαχείρισης αριθμητικών μονάδων, κλάσεις κωδικοποίησης και αποκωδικοποίησης καθώς και κάποια εργαλεία χρήστη για είσοδο ή έξοδο συγκεκριμένων αντικειμένων.

Πάνω σε αυτήν την βάση κάθε νέα εφαρμογή που υλοποιείτε από μία εταιρία ανάπτυξης λογισμικού θα πρέπει να χρησιμοποιεί, εάν αυτό είναι απαραίτητο, κλάσεις και εργαλεία χρήστη από προηγούμενες εφαρμογές που έχει αναπτύξει η εταιρία βελτιώνοντας

τα παράλληλα για μελλοντική χρήση, καθώς επίσης κάθε νέα κλάση ή εργαλείο χρήστη που αναπτύσσεται για αυτήν την νέα εφαρμογή θα πρέπει να υλοποιείται με τέτοιον τρόπο ώστε να υπάρχει η δυνατότητα μελλοντικής επαναχρησιμοποίησης του, εφόσον αυτό αφορά κάτι γενικό και υπάρχει αυτή η δυνατότητα.

1.3.2 – Εύκολη αναβάθμιση

Καθημερινός βλέπουμε να κυκλοφορούν στην αγορά όλο και ποιο νέες εκδόσεις γνωστών εφαρμογών. Είναι λογικό αυτές οι νέες εκδόσεις να μην έχουν ξαναγραφτεί από την αρχή αλλά να έχουν απλά βελτιωθεί σε επιμέρους κομμάτια τους και συνήθως να περιέχουν και κάποιες νέες δυνατότητες. Έτσι λοιπόν μία εμπορική εφαρμογή θα πρέπει να αναπτυχθεί με τέτοιον τρόπο ώστε πέρα από την επαναχρησιμοποίηση μέρους του κώδικά της να μπορεί να αναβαθμιστεί εύκολα βελτιώνοντας τον τρόπο λειτουργίας της και προσθέτοντας νέες δυνατότητες και λειτουργίες.

Για να επιτευχθεί κάτι τέτοιο θα πρέπει πριν από την ανάπτυξη και την υλοποίηση της εφαρμογής να έχει προηγηθεί σωστή και σε βάθος μελέτη καθώς και ένας πολύ καλός σχεδιασμός της. Ένας γενικά αποδεκτός τρόπος για την επίτευξη αυτού του στόχου είναι ο διαχωρισμός της εφαρμογής σε πολλά μικρά τμήματα κώδικα τα οποία συνήθως αφορούνε το καθένα ένα διαφορετικό αντικείμενο ή οντότητα ή εκτελούνε το καθένα μία διαφορετική εργασία. Καθένα από αυτά τα τμήματα της εφαρμογής λειτουργεί ανεξάρτητα από τα υπόλοιπα χωρίς να δίνει πληροφορίες προς τα έξω για τον τρόπο με τον οποίο επιτυγχάνει να φέρει εις πέρας την αποστολή του. Το μόνο που αφήνει να φανεί προς τα έξω είναι κάποιες μεθόδους και κάποιες ιδιότητες με τις οποίες κάποιο άλλο κομμάτι του κώδικα χρησιμοποιώντας τες παίρνει τα αποτελέσματα ή εκτελεί την λειτουργία με την οποία έχει επιφορτιστεί το συγκεκριμένο τμήμα της εφαρμογής. Κάθε τέτοιο κομμάτι κώδικα αποτελεί ένα αντικείμενο και ονομάζετε κλάση.

Γίνετε λοιπόν φανερό ότι η υλοποίηση μιας εμπορικής εφαρμογής, είτε αυτή είναι διαχειριστική είτε όχι, είναι κάτι το οποίο απαιτεί αρκετό χρόνο και πολύ μελέτη για να γίνει σωστά. Στην περίπτωση που η εταιρία ανάπτυξης της εφαρμογής έχει στο ενεργητικό της αρκετές εφαρμογές τα πράγματα γίνονται ποιο εύκολα και εξαιτίας της εμπειρίας που έχει ήδη αποκτήσει η εταιρία στην ανάπτυξη εφαρμογών αλλά και εξαιτίας του ότι θα έχει στην διάθεσή της κάποια έτοιμα κομμάτια κώδικα τα οποία θα μπορεί να τα επαναχρησιμοποιήσει σε αυτήν την νέα εφαρμογή που θέλει να αναπτύξει.

Συνήθεις πρόβλημα στην ανάπτυξη εμπορικών εφαρμογών είναι ο περιορισμένος χρόνος που έχει η εταιρία για την παράδοση της εφαρμογής στην επιχείρηση που την ζήτησε εξαιτίας του ότι οι επιχειρήσεις συνήθως βιάζονται να αποκτήσουν την εφαρμογή

γιατί έχουν ήδη επενδύσει κάποια χρήματα σε αυτήν καθώς και γιατί με αυτήν έχουν σαν στόχο την ποιο αποδοτική λειτουργία τους με σκοπό τον πολλαπλασιασμό των κερδών τους. Στην περίπτωση που η εφαρμογή δεν είναι παραγγελία μιας επιχείρησης αλλά πρόκειται να κυκλοφορήσει στην αγορά με στόχο επιχειρήσεις που ασχολούνται με κάποιο συγκεκριμένο αντικείμενο η πίεση για την άμεση παράδοση της εφαρμογής δεν έρχεται έξω από την εταιρία αλλά μέσα από τα ίδια τα στελέχη της και αυτό γιατί η εταιρία έχει επενδύσει χρήμα και χρόνο για την υλοποίηση της εφαρμογής και περιμένει κάποιο κέρδος από αυτήν το συντομότερο δυνατόν.

2 – Η Μελέτη και Ο Σχεδιασμός της Εφαρμογής

Κατά την Μελέτη μιας Διαχειριστικής εφαρμογής γίνετε φανερό από την αρχή ότι θα χρειαστεί να αναπτυχτούν κάποιες κλάσεις βασικές και παρόμοιες σε κάθε είδους εφαρμογή διαχείρισης. Κλάσεις καταγραφής γεγονότων και ρυθμίσεων καθώς και κλάσεις διασύνδεσης με την βάση δεδομένων συναντούνται σε κάθε είδους εφαρμογή διαχείρισης.

Συνήθως γίνετε φανερή η ανάγκη δημιουργίας συγκεκριμένων εργαλείων και αντικειμένων από την αρχή της Μελέτης μιας εφαρμογής. Παραδείγματος χάριν στην δικιά μας Εφαρμογή φάνηκε από την αρχή ότι θα χρειαστούμε ένα αντικείμενο προβολής και διαχείρισης λίστας καθώς και μία κλάση για την αυτόματη εκτύπωση αυτής της λίστας. Ακόμα ένα παράδειγμα εδώ είναι η μπάρα προόδου που αναπτύχθηκε λόγο της σχετικά μεγάλης χρονικής διάρκειας που χρειάζεται για να γίνει ο καταμερισμός των δαπανών κατά την έκδοση των κοινοχρήστων.

Μετά από την βασική μελέτη για την εφαρμογή και τον αρχικό σχεδιασμό έρχεται η στιγμή να παρθούν αποφάσεις για την καθαυτού λειτουργία της εφαρμογής. Οι αποφάσεις αυτές θα πρέπει να καλύπτουν καταρχάς τις βασικές προϋποθέσεις μιας Διαχειριστικής Εφαρμογής, ευκολία αναβάθμισης και επαναχρησιμοποιήσιμος κώδικας. Πέρα αυτών των βασικών προϋποθέσεων και ποιο συγκεκριμένα η δικιά μας εφαρμογή θα πρέπει να καλύπτει τις παρακάτω προϋποθέσεις:

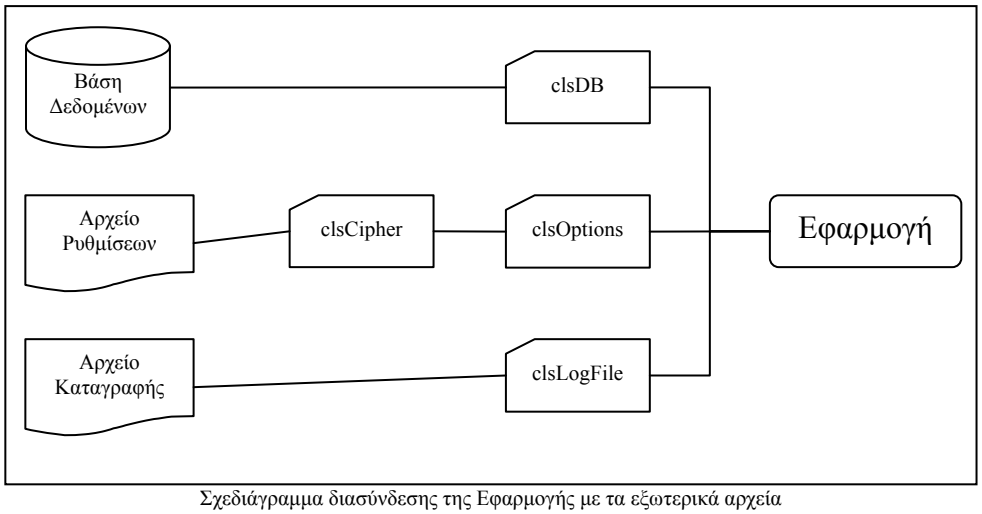
- δυνατότητα καταμερισμού δαπανών με όσο το δυνατών περισσότερους τρόπους
- δυνατότητα να αποφασίζει ο χρήστης τους τρόπους καταμερισμού των δαπανών κάθε πολυκατοικίας ξεχωριστά
- δυνατότητα δημιουργίας από τον χρήστη νέων τρόπων καταμερισμού δαπανών
- ευελιξία κατά την έκδοση των κοινοχρήστων με την δυνατότητα να επανεκδώσει ή να ακυρώσει ήδη εκδομένα κοινόχρηστα κατά όπως αυτός επιθυμεί

Παρακάτω θα παρουσιαστούν και θα αναπτυχθούν βήμα προς βήμα η βασική μελέτη, ο σχεδιασμός των χρήσιμων εργαλείων και ο σχεδιασμός της Εφαρμογής.

Όπως αναφέρθηκε και παραπάνω κάθε διαχειριστική εφαρμογή ακολουθεί έναν βασικό σχεδιασμό για τον τρόπο με τον οποίο θα διαχειρίζεται τα εξωτερικά της αρχεία. Τα εξωτερικά αρχεία που θα χρειαστούν για την δικιά μας εφαρμογή είναι τα ακόλουθα:

- η Βάση Δεδομένων στην οποία θα αποθηκεύονται τα δεδομένα της εφαρμογής
- το Αρχείο Ρυθμίσεων στο οποίο θα φυλάγονται οι ρυθμίσεις του χρήστη καθώς και οι προτιμήσεις του χρήστη για τον τρόπο προβολής της εφαρμογής
- το Αρχείο Καταγραφής Γεγονότων στο οποίο θα καταγράφετε κάθε πρωτεύων κίνηση του χρήστη καθώς και κάθε μήνυμα λάθους της εφαρμογής

Για να έχει πρόσβαση η εφαρμογή μας σε αυτά τα αρχεία θα πρέπει να αναπτυχθούν τουλάχιστον τρείς κλάσεις, κάθε μία αντίστοιχα για κάθε αρχείο. Στην προκειμένη περίπτωση εμείς αναπτύξαμε και μία επιπλέων κλάση για την κωδικοποίηση του Αρχείου Ρυθμίσεων για λόγους που θα αναπτυχθούν ποιο διεξοδικά παρακάτω.



Σχεδιάγραμμα διασύνδεσης της Εφαρμογής με τα εξωτερικά αρχεία

2.1.1 – Διασύνδεση με την Βάση Δεδομένων

Για την δημιουργία της Βάσης Δεδομένων χρησιμοποιήθηκε η Microsoft Access κυρίως για την ευκολία λειτουργίας της και την ταχύτητα με την οποία μπορούμε να δημιουργήσουμε και να μεταβάλουμε μια βάση δεδομένων μέσα από το περιβάλλων της. Θα μπορούσαμε κάλλιστα να χρησιμοποιήσουμε MySQL ή κάποια άλλη Open Source εφαρμογή για την δημιουργία της Βάσης Δεδομένων. Εξαιτίας αυτού και της πιθανότητας στο μέλλον να αποφασίσουμε να αλλάξουμε την Βάση Δεδομένων από Microsoft Access σε κάτι άλλο, η κλάση με την οποία διασυνδέεται η Εφαρμογή μας με την Βάση Δεδομένων θα πρέπει να κρύβει τελείως το είδος της Βάσης Δεδομένων και να δίνει στην Εφαρμογή μας γενικευμένες μεθόδους για την επικοινωνία με αυτήν.

Η ανεξαρτησία της εφαρμογής από τον τύπο της βάσης δεδομένων που χρησιμοποιεί επιτυγχάνετε μέσα από την κλάση clsDB. Η clsDB μέσα από τον Constructor της μας δίνει την δυνατότητα να την αρχικοποιήσουμε ανάλογα με το είδος της βάσης δεδομένων που έχουμε.

```
Public Sub New(ByVal strConnectionString As System.String)
```

Στην περίπτωσή μας δημιουργούμε ένα αντικείμενο objDB τύπου clsDB μέσα στο mdlMain. Επιλέξαμε να δημιουργήσουμε το αντικείμενο διασύνδεσης με την βάση δεδομένων μέσα σε ένα module της Εφαρμογής για να μπορούμε να έχουμε πρόσβαση σε αυτό από οποιοδήποτε σημείο της Εφαρμογής. Βεβαίως το αντικείμενο δημιουργήθηκε ως Private μέσα στο module και προσφέρετε ως ReadOnly Property DB στην υπόλοιπη Εφαρμογή.

```
    Private objDB As clsDB
    Public ReadOnly Property DB() As clsDB
        Get
            Return objDB
        End Get
    End Property 'DB
```

Κατά την έναρξη της εφαρμογής καλείται η μέθοδος InitializeApplication από το ίδιο module η οποία αρχικοποιεί τα βασικά στοιχεία της εφαρμογής και μέσα σε αυτά και το αντικείμενο objDB.

```
    Private Sub frmMain_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load
        mdlMain.InitializeApplication()
        ...
    End Sub 'frmMain_Load
```

Από την στιγμή που χρησιμοποιούμε Microsoft Access για τη Βάση Δεδομένων μας θα πρέπει στο αλφαριθμητικό του Constructor του αντικειμένου objDB να ορίσουμε ως Provider το Microsoft.Jet.OLEDB.4.0.

```
    Public Sub InitializeApplication()
         ...
         objDB = New clsDB("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
My.Application.Info.DirectoryPath & "\" & My.Application.Info.ProductName.Replace(" ", "")
& ".dbs;User Id=admin;Password=;Jet OLEDB:Engine Type=5")
         ...
    End Sub 'InitializeApplication
```

Σε αυτό το σημείο έχουμε πρόσβαση από την Εφαρμογή μας σε μία Βάση Δεδομένων μέσα από την Ιδιότητα DB χωρίς να γνωρίζουμε το είδος της βάσης δεδομένων και κυρίως χωρίς να μας ενδιαφέρει.

Στην περίπτωση που αποφασίσουμε να αλλάξουμε το είδος της βάσης δεδομένων το μόνο που έχουμε να κάνουμε είναι να αλλάξουμε κατάλληλα το αλφαριθμητικό που θα δώσουμε στον Constructor του αντικειμένου objDB μέσα στην μέθοδο InitializeApplication του module mdlMain.

2.1.2 – Διασύνδεση με το Αρχείο Ρυθμίσεων

Το Αρχείο Ρυθμίσεων της Εφαρμογής ακολουθεί το βασικό πρότυπο της XML δημιουργώντας tags για κάθε ρύθμιση και αποθηκεύοντας την τιμή της κάθε ρύθμισης μέσα στο tag. Για την πρόσβαση της Εφαρμογής στο Αρχείο Ρυθμίσεων δημιουργήθηκε η κλάση clsOptions. Μέσα στο Αρχείο Ρυθμίσεων πέρα από τις επιλογές του χρήστη αποθηκεύονται και οι τελευταίες ρυθμίσεις που έκανε ο χρήστης στον τρόπο εμφάνισης

της εφαρμογής όπως παραδείγματος χάριν οι διαστάσεις τις κάθε στήλης για κάθε αντικείμενο ctlListView που υπάρχει στην Εφαρμογή.

Εξαιτίας τις ευαισθησίας που έχουν αυτές οι πληροφορίες για την σωστή λειτουργία της εφαρμογής θεωρήσαμε σκόπιμο να τις αποκρύψουμε από τον χρήστη με κάθε δυνατό τρόπο. Άλλωστε σε μία εμπορική εφαρμογή οι πληροφορίες ενεργοποίησης του προϊόντος και τα στοιχεία άδειας χρήσης του είναι κάποια από τα στοιχεία που θα μπορούσαν να αποθηκευτούν μέσα στο Αρχείο Ρυθμίσεων μαζί με τα προσωπικά στοιχεία του χρήστη ή της εταιρίας που ενεργοποίησε την άδεια χρήσης της εφαρμογής, κάνοντας τις πληροφορίες που περιέχει το Αρχείο Ρυθμίσεων ακόμα ποιο ευαίσθητες και κρίσιμες.

Ο τρόπος που επιλέχτηκε για την απόκρυψη αυτών των πληροφοριών αναπτύχθηκε σε μία ξεχωριστή κλάση. Η κλάση αυτή θα παρεμβάλετε ανάμεσα στο Αρχείο Ρυθμίσεων και την κλάση clsOptions και θα κρυπτογραφεί τις πληροφορίες που αποθηκεύονται στο Αρχείο Ρυθμίσεων. Η κλάση αυτή είναι η clsCipher.

Η κλάση clsCipher σχεδιάστηκε έτσι ώστε να μπορεί να κρυπτογραφήσει οποιοδήποτε αλφαριθμητικό. Εξαιτίας του ότι το κρυπτογραφημένο αλφαριθμητικό μπορεί να περιέχει χαρακτήρες οι οποίοι δεν έχουν θέση σε ένα αλφαριθμητικό αποφασίσαμε να χειριστούμε το κρυπτογραφημένο αλφαριθμητικό ως ένα Base 64 String για να αποφύγουμε τυχών προβλήματα.

2.1.3 – Διασύνδεση με το Αρχείο Καταγραφής Γεγονότων

Για την διασύνδεση της Εφαρμογής με το Αρχείο Καταγραφής δημιουργήθηκε η κλάση clsLogFile. Κάθε πρωτεύων κίνηση του χρήστη, όπως η εισαγωγή ή η διαγραφή ενός κτιρίου, καταγράφετε μέσα στο Αρχείο Καταγραφής. Πέραν των κινήσεων του χρήστη στο Αρχείο Καταγραφής αποθηκεύονται κάθε μήνυμα επικοινωνίας που προβάλει η Εφαρμογή μας προς τον χρήστη καθώς και κάθε απόφαση που καλείτε να λάβει αυτός. Επίσης στο Αρχείο Καταγραφής αποθηκεύονται και όλα τα μηνύματα λάθους.

Ο λόγος για τον οποίο δημιουργήθηκε το Αρχείο Καταγραφής δεν είναι άλλος από την δημιουργία μίας αξιόπιστης πηγής πληροφοριών για τις ενέργειες που έκανε ο χρήστης ακριβός πριν από ένα μήνυμα λάθους. Με αυτόν τον τρόπο έχουμε την δυνατότητα να μάθουμε τι ήταν αυτό που οδήγησε την εφαρμογή σε λάθος έτσι ώστε να το διορθώσουμε.

Σε αυτό το σημείο υπήρχε το δίλημμα για το εάν είναι ορθό να κρυπτογραφήσουμε αυτό το αρχείο ή όχι. Εξαιτίας του ότι αυτό το αρχείο θα ζητηθεί από τον χρήστη για να ερευνήσουμε το πρόβλημα που αντιμετωπίζει η εφαρμογή, θεωρήσαμε ποιο σωστό να έχει την δυνατότητα ο χρήστης να δει εύκολα αυτό το αρχείο και τις πληροφορίες που μεταφέρει προτού το αποστείλει σε εμάς έτσι ώστε να είναι σίγουρος ότι δεν περιέχει ευαίσθητα προσωπικά δεδομένα ή πληροφορίες για το σύστημα του.

Μετά από την βασική μελέτη μιας εφαρμογής αρχίζει να γίνετε φανερό ότι θα χρειαστεί να αναπτυχθούν κάποια εργαλεία χρήστη και κάποια αντικείμενα τα οποία θα μας βοηθήσουν στην διαχείριση και την προβολή σημαντικών τμημάτων της εφαρμογής προς τον χρήστη. Η ανάπτυξη αυτών των τμημάτων του κώδικα θα πρέπει να γίνει με κύριο μέλημα την δυνατότητα χρησιμοποίησής τους και σε άλλες εφαρμογές που πραγματεύονται διαφορετικό αντικείμενο διαχείρισης από το δικό μας.

2.2.1 – Εργαλείο χρήστη για την διαχείριση λίστας

Όπως πολύ πιθανών και σε κάθε άλλη διαχειριστική εφαρμογή έτσι και εδώ θα χρειαστεί να παρουσιάσουμε στον χρήστη πληροφορίες σε μορφή λίστας πολύ περισσότερες φορές από μία. Εξαιτίας αυτού θεωρήθηκε σκόπιμο από την πλευρά μας να αναπτύξουμε ένα εργαλείο χρήστη με το οποίο θα μπορούμε εύκολα, γρήγορα και κυρίως αποτελεσματικά να παρουσιάσουμε αυτά τα στοιχεία στον χρήστη. Ένα τέτοιο εργαλείο μπορεί, και θα πρέπει, κάλλιστα να αναπτυχθεί με κύριο γνώμονα την δυνατότητα να χρησιμοποιηθεί και σε άλλες εφαρμογές στα πλαίσια του επαναχρησιμοποιήσιμου κώδικα που θέσαμε στην αρχή αυτής της μελέτης.

Οι βασικοί μας στόχοι, πέραν της δυνατότητας επαναχρησιμοποίησης του, που θέσαμε κατά την ανάπτυξη αυτού του εργαλείου χρήστη είναι οι ακόλουθοι:

- δυνατότητα ομαδοποίησης των εγγραφών
- δυνατότητα ταξινόμησης των εγγραφών ανά στήλη
- δυνατότητα επιλογής από τον χρήστη περισσότερων από μία εγγραφών
- δυνατότητα παραμετροποίησης των διαστάσεων κάθε στήλης και αποθήκευσης των προτιμήσεων του χρήστη

Για την δημιουργία αυτού του εργαλείου χρήστη χρησιμοποιήθηκε ως βάση το αντικείμενο ListView από το Namespace System.Windows.Forms του .Net Framework. Τελικό αποτέλεσμα σε αυτήν την προσπάθεια είναι το εργαλείο χρήστη ctlListView το οποίο ενσωματώνει και τους τέσσερις στόχους που θέσαμε για αυτό.

2.2.2 – Κλάση για την αυτοματοποιημένη εκτύπωσης λίστας

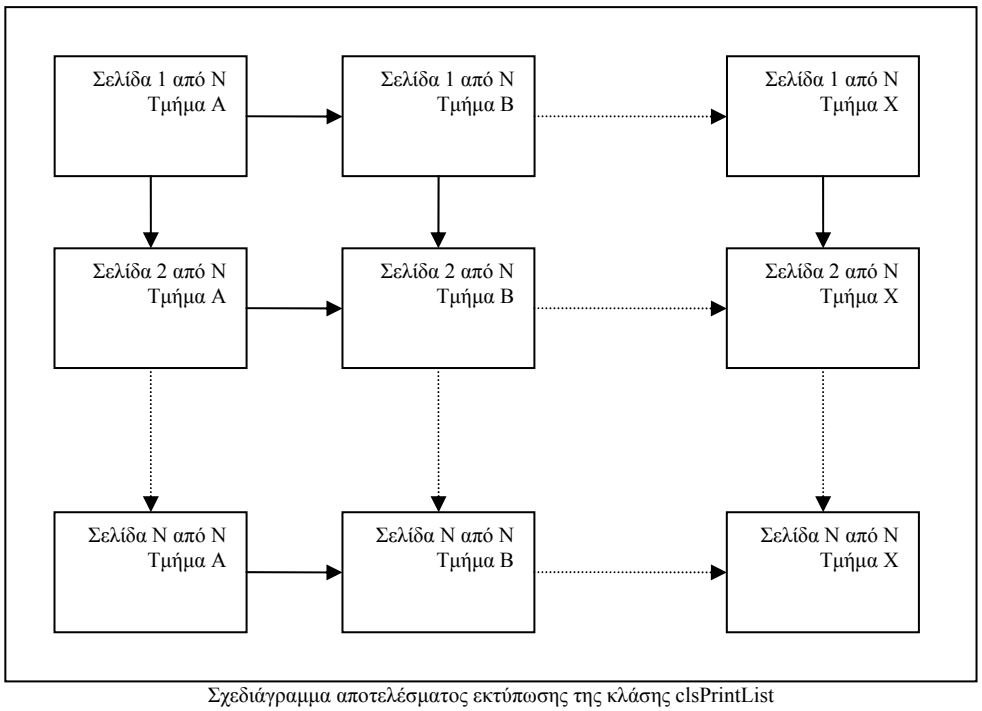
Κατά την δημιουργία του εργαλείου χρήστη ctlListView έγινε φανερή η ανάγκη να δοθεί στον χρήστη η δυνατότητα να εκτυπώνει αυτές τις λίστες όποτε αυτός επιθυμεί. Το γεγονός ότι το εργαλείο αυτό δίνει στον χρήστη την δυνατότητα να εμφανίσει την λίστα όπως αυτός επιθυμεί καθώς και το ότι η λίστα αυτή μπορεί να περιέχει οποιοδήποτε πλήθος εγγραφών και πεδίων σε κάθε εγγραφή μας οδήγησε στην απαίτηση για

δημιουργία μίας κλάσης η οποία θα μπορεί να χειριστεί και να εκτυπώσει οποιαδήποτε λίστα.

Την λύση σε αυτήν την απαίτηση την δώσαμε με την δημιουργία της κλάσης clsPrintList η οποία μπορεί να δημιουργήσει μια αξιόπιστη εκτύπωση, για κάθε λίστα που της δίνουμε, αυτόματα και προσφέροντάς μας εκτυπώσεις οι οποίες:

- διαχωρίζονται σε αριθμημένες σελίδες εάν το πλήθος των εγγραφών το απαιτεί
- διαχωρίζοντας κάθε σελίδα σε περισσότερα από ένα τμήματα εάν το πλήθος των πεδίων της κάθε εγγραφής το απαιτεί

Για να γίνει ποιο κατανοητή η λειτουργία της κλάσης παρουσιάζουμε το παρακάτω σχεδιάγραμμα.



Σχεδιάγραμμα αποτελέσματος εκτύπωσης της κλάσης clsPrintList

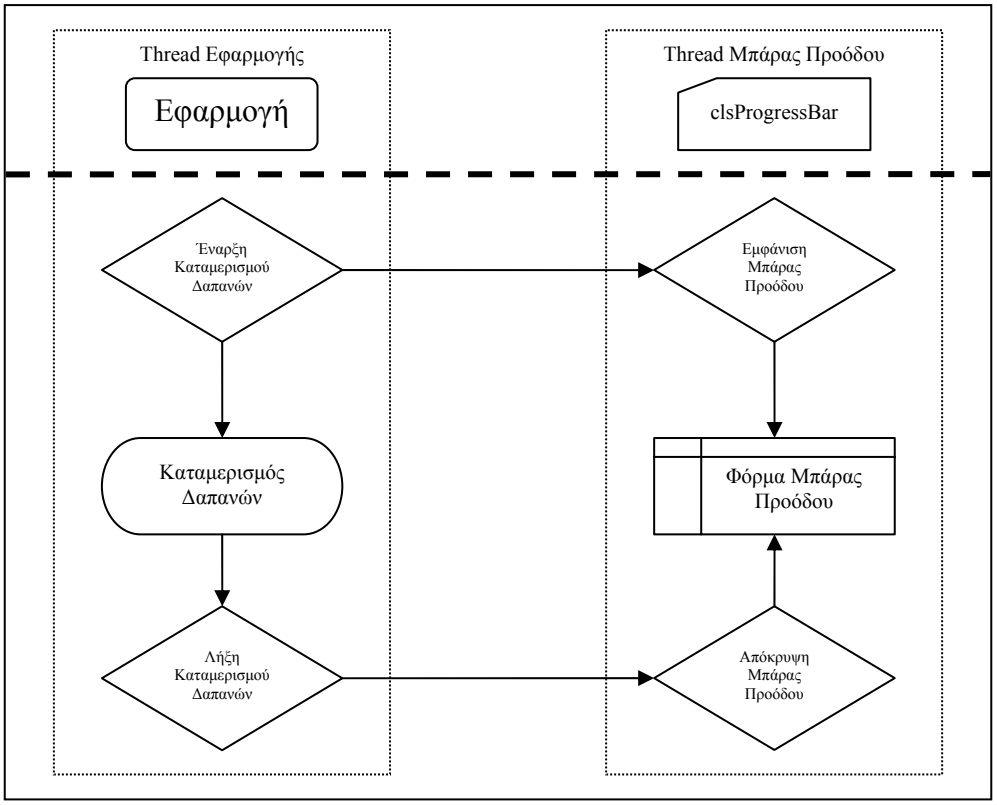
2.2.3 –Μπάρα Προόδου σε διαφορετικό Thread από την Εφαρμογή

Εξαιτίας του ότι ο καταμερισμός των δαπανών για την έκδοση κοινοχρήστων είναι μία διαδικασία η οποία απαιτεί περισσότερο χρόνο από δυο ή τρία δευτερόλεπτα και σε ορισμένες ποιο περίπλοκες περιπτώσεις μέχρι και ίσως περισσότερο από ένα λεπτό, χρειάστηκε να αντιμετωπίσουμε αυτό το πάγωμα της εφαρμογής και να δήξουμε στον χρήστη ότι εκτελείτε κάποια περίπλοκη εργασία.

Για την αντιμετώπιση αυτού του παγώματος η ποιο προφανής λύση είναι μία μπάρα προόδου η οποία θα δείχνει στον χρήστη ότι η εφαρμογή κάνει κάποια εργασία. Αυτή ακριβός την λύση διαλέξαμε και εμείς με μία μικρή διαφορά. Η μπάρα προόδου θα

πρέπει να τρέχει σε ένα διαφορετικό Thread έτσι ώστε να αποφευχθεί το πάγωμα της εφαρμογής και να της επιτραπεί να συνεχίσει τον καταμερισμό των δαπανών.

Για την δημιουργία του διαφορετικού Thread δημιουργήσαμε την κλάση clsProgressBar. Για την εμφάνιση της Μπάρας Προόδου δημιουργήσαμε την φόρμα frmProgressBar. Η κλάση clsProgressBar αναλαμβάνει να δημιουργήσει ένα νέο αντικείμενο τύπου frmProgressBar και να το εμφανίσει στον χρήστη μέσα σε ένα Thread διαφορετικό από αυτό που τρέχει η Εφαρμογή κατά την έναρξη του καταμερισμού των δαπανών. Όταν η εφαρμογή ολοκληρώσει τον καταμερισμό των δαπανών ενημερώνει την κλάση clsProgressBar που διαχειρίζεται το Thread της Μπάρας Προόδου και η τελευταία αναλαμβάνει να κλείσει την φόρμα της Μπάρας Προόδου και να τερματίσει το Thread.



Σχεδιάγραμμα

2.2.4 – Διαχείριση γεγονότων (Event) καθολικών στην εφαρμογή

Κατά την ανάπτυξη της εφαρμογής χρειάστηκε πολλές φορές να ενημερωθεί μία λίστα για ένα γεγονός που συνέβη σε ένα κομμάτι της εφαρμογής τελείως απομονωμένο από αυτήν. Επίσης σε αρκετά σημεία της εφαρμογής ενέργειες σου χρήστη οδηγούν στην ανάγκη να ενημερωθούν κάποιες λίστες με τα νέα στοιχεία που μας έδωσε ο χρήστης χωρίς να μπορούμε να γνωρίζουμε εάν αυτές οι λίστες και κυρίως ποιες από αυτές προβάλλονται αυτήν την στιγμή στον χρήστη.

Για την αντιμετώπιση αυτής της απαίτησης αναπτύξαμε την κλάση clsGlobalEvents μέσα από την οποία μπορούμε να ενεργοποιήσουμε συγκεκριμένα γεγονότα για την ανανέωση της κάθε λίστας. Μέσα από την Δημόσια Μέθοδο RaiseGlobalEvent της κλάσης, η οποία παίρνει ως όρισμα ένα αλφαριθμητικό στο οποίο περιγράφεται το γεγονός που θέλουμε να ενεργοποιήσουμε, μπορούμε από οποιοδήποτε σημείο της εφαρμογής μας να ενεργοποιήσουμε αυτό το γεγονός. Σε κάθε σημείο της εφαρμογής που έχουμε κάποια λίστα έχουμε επίσης και έναν χειριστή για το γεγονός της κλάσης clsGlobalEvents που αναφέρετε σε αυτήν την λίστα έτσι ώστε όταν αυτό ενεργοποιηθεί και η λίστα αυτή προβάλετε στον χρήστη να ανανεωθεί.

3 – Η Ανάπτυξη της Εφαρμογής Α` – Τα βασικά αντικείμενα

Σε αυτό το κεφάλαιο παρουσιάζονται τα αντικείμενα, όπως οι κλάσεις, τα εργαλεία χρήστη και οι φόρμες, που αναπτυχθήκαν για την εφαρμογή καθώς και ο τρόπος λειτουργίας τους. Τα αντικείμενα παρουσιάζονται με σειρά ανάλογη με αυτήν που παρουσιαστήκαν στο προηγούμενο κεφάλαιο, στον σχεδιασμό της εφαρμογής. Όλα αυτά τα αντικείμενα είναι τα βασικά που χρειάζονται για την ανάπτυξη μιας εμπορικής εφαρμογής η οποία θα είναι σε θέση να διαχειριστεί μία βάση δεδομένων και να παρουσιάσει στον χρήστη τα αποτελέσματα μιας στατιστικής του έρευνας.

Η κλάση διασύνδεσης με την βάση δεδομένων χρησιμοποιεί το αντικείμενο OleDbConnection από το Namespace System.Data.OleDb του .Net Framework για να συνδεθεί με την βάση δεδομένων και προσφέρει διάφορες μεθόδους για την ανάκτηση και την αποθήκευση των δεδομένων σε αυτήν. Με αυτόν τον τρόπο επιτυγχάνεται η ανεξαρτησία της εφαρμογής από τύπο της βάσης δεδομένων. Όλες οι μέθοδοι της κλάσης επιστρέφουν μία τιμή Boolean για το εάν εκτελέστηκαν σωστά ή παρουσιάστηκε κάποιο λάθος στην εκτέλεσή τους. Η ιδιότητα LastException της κλάσης μας προσφέρει την δυνατότητα να ανακτήσουμε το τελευταίο λάθος που παρουσιάστηκε κατά την εκτέλεση κάποιας μεθόδου. Η κλάση προσφέρει επίσης και την μέθοδο FilterString η οποία μπορεί να χρησιμοποιηθεί και χωρίς να έχει δημιουργηθεί κάποιο αντικείμενο της κλάσης, δηλαδή η μέθοδος είναι κοινή.

3.1.1 – Οι Constructor της κλάσης clsDB

```
Public Sub New(ByVal strConnectionString As System.String)
```

Ο Constructor της κλάσης, όπως θα ήταν αναμενόμενο, παίρνει σαν είσοδο ένα αλφαριθμητικό για να αρχικοποιήσει την κλάση. Στο αλφαριθμητικό αυτό συνήθως ορίζονται ο τύπος της βάσης δεδομένων, η θέση της, οι κωδικοί για την πρόσβαση σε αυτήν και ότι άλλο είναι απαραίτητο με τέτοιο τρόπο αναλόγως με τον τύπο της βάσης δεδομένων που θέλουμε να συνδέσουμε.

3.1.2 – Οι Ιδιότητες της κλάσης clsDB

```
Public ReadOnly Property LastException() As System.String
```

Η ιδιότητα LastException μας επιστρέφει ένα αλφαριθμητικό το οποίο περιγράφει το τελευταίο λάθος που παρουσιάστηκε κατά την εκτέλεση κάποιας από τις μεθόδους της κλάσης. Στην περίπτωση που δεν υπάρχει κανένα λάθος η ιδιότητα επιστρέφει Nothing.

Επίσης κάθε φορά που ανακτούμε το τελευταίο λάθος η κλάση φροντίζει εσωτερικά να αρχικοποιήσει την ιδιότητα με την τιμή Nothing, δηλαδή ότι δεν υπάρχει λάθος.

3.1.3 – Οι Μέθοδοι της κλάσης clsDB

```
Public Function PopulateDataSet(ByVal strTableName As System.String, ByRef objDataSet As
System.Data.DataSet) As System.Boolean
```

Η μέθοδος PopulateDataSet έχει δύο εισόδους, ένα αλφαριθμητικό που ορίζει το όνομα του πίνακα που θα ανακτήσουμε και ένα αντικείμενο τύπου DataSet στο οποίο θα αποθηκευτεί ο πίνακας και περνιέται στην μέθοδο ως διεύθυνση μνήμης έτσι ώστε κάθε αλλαγή που θα γίνει εσωτερικά στην μέθοδο μα επιστρέψει μετά το τέλος της εκτέλεσης της μεθόδου. Η μέθοδος αναλαμβάνει να ανακτήσει ολόκληρο τον πίνακα που έχουμε ορίσει και να μας τον επιστρέψει μέσω της μεταβλητής objDataSet. Η μέθοδος επιστρέφει True εάν όλα εκτελέστηκαν σωστά και False εάν υπήρξε κάποιο λάθος.

```
Public Function PopulateDataSet(ByVal strSQL As System.String, ByVal strTableName As
System.String, ByRef objDataSet As System.Data.DataSet) As System.Boolean
```

Η μέθοδος PopulateDataSet εκτός από τον ορισμό που αναφέρθηκε παραπάνω έχει και έναν άλλον ορισμό με τρεις εισόδους. Η διαφορά με πριν είναι η νέα αυτή είσοδος που είναι αλφαριθμητικό και αποτελεί ένα SQL ερώτημα το οποίο μας επιτρέπει να ανακτήσουμε περιορισμένο αριθμό εγγραφών και πεδίων ανάλογα με τι μας εξυπηρετεί καλύτερα. Η μέθοδος επιστρέφει και πάλι True/False αναλόγως με την επιτυχία ή όχι της μεθόδου.

```
Public Function UpdateDataSet(ByVal strTableName As System.String, ByVal objDataSet As
System.Data.DataSet) As System.Boolean
```

Η μέθοδος UpdateDataSet έχει δύο εισόδους, ένα αλφαριθμητικό στο οποίο ορίζετε το όνομα του πίνακα που θέλουμε να ενημερώσουμε και ένα αντικείμενο τύπου DataSet το οποίο περιέχει τον πίνακα αυτόν. Η μέθοδος αναλαμβάνει να ενημερώσει την βάση δεδομένων για τις αλλαγές που υπάρχουν στον πίνακα. Όπως κάθε άλλη μέθοδος της κλάσης επιστρέφει True/False αναλόγως με την επιτυχία της.

```
Public Function UpdateField(ByVal strTableName As System.String, ByVal strFieldName As
System.String, ByVal strFieldNewValue As System.String, ByVal strCriteria As System.String)
As System.Boolean
```

Προτελευταία μέθοδος της κλάσης είναι η UpdateField η οποία έχει τέσσερα αλφαριθμητικά για είσοδο. Το όνομα του πίνακα, το όνομα του πεδίου που θέλουμε να ενημερώσουμε, την νέα τιμή που θέλουμε να θέσουμε και τα κριτήρια με τα οποία θα γίνει η ενημέρωση. Η μέθοδος επιστρέφει True/False ανάλογα με την επιτυχία της.

```
Public Function DeleteRecord(ByVal strTableName As System.String, ByVal strCriteria As
System.String) As System.Boolean
```

Τελευταία μέθοδος της κλάσης είναι η DeleteRecord η οποία έχει δύο αλφαριθμητικά για είσοδο, το όνομα του πίνακα στον οποίο θέλουμε να σβήσουμε εγγραφές και τα κριτήρια με τα οποία θα γίνει η διαγραφή αυτών. Η μέθοδος επιστρέφει True/False ανάλογα με την επιτυχία της.

3.1.4 – Οι Κοινές Μέθοδοι της κλάσης clsDB

```
Public Shared Function FilterString(ByVal strText As String) As String
```

Η κλάση προσφέρει και μία κοινή μέθοδο η οποία μπορεί να χρησιμοποιηθεί χωρίς να έχει δημιουργηθεί κάποιο αντικείμενο της κλάση. Η μέθοδος αυτή ονομάζετε FilterString και αναλαμβάνει να αντικαταστήσει στο αλφαριθμητικό της εισόδου όλους εκείνους τους χαρακτήρες που θεωρούνται επικίνδυνοι για την SQL, όπως παραδείγματος χάρη τα μονά και διπλά εισαγωγικά, με τον χαρακτήρα που αντιπροσωπεύει το κενό. Η μέθοδος επιστρέφει το τροποποιημένο αλφαριθμητικό.

3.1.5 – Ο κώδικας της κλάσης clsDB

```
Option Explicit On
Option Strict On

Public Class clsDB

    Private _strConnectionString As String
    Private _objOleDbConnection As System.Data.OleDb.OleDbConnection
    Private _expLastException As System.Exception

    Public Sub New(ByVal strConnectionString As System.String)
        Me._objOleDbConnection = New System.Data.OleDb.OleDbConnection()
        Me._strConnectionString = strConnectionString

        Me._objOleDbConnection.ConnectionString = Me._strConnectionString

        Try
            Me._objOleDbConnection.Open()
        Catch ex As System.Exception
            mdlMain.LogFile.WriteGlobalException( _
                    ex.Source & vbNewLine & _
                    vbNewLine & _
                    ex.Message & vbNewLine & _
                    ex.StackTrace, "Error in clsDB...", True)
            End
        End Try
    End Sub 'New

    Protected Overrides Sub Finalize()
        Try
            Me._objOleDbConnection.Close()
            Me._objOleDbConnection = Nothing
        Catch ex As Exception
        End Try
    End Sub 'Finalize

#Region " Properties "

    Public ReadOnly Property LastException() As System.String
        Get
            If Not Me._expLastException Is Nothing Then
                Dim strTemp As String = Me._expLastException.Source & ControlChars.NewLine
& ControlChars.NewLine & _
                    Me._expLastException.Message & ControlChars.NewLine & _
                    Me._expLastException.StackTrace
                Me._expLastException = Nothing
                Return strTemp
            Else
                Return "No Exception"
            End If
        End Get
    End Property 'LastException

#End Region

#Region " Public "
```

```
    Public Function PopulateDataSet(ByVal strTableName As System.String, ByRef objDataSet
As System.Data.DataSet) As System.Boolean
        Dim objOleDbCommand As System.Data.OleDb.OleDbCommand
        Dim objOleDbDataAdapter As System.Data.OleDb.OleDbDataAdapter
        Try
            objOleDbCommand = New System.Data.OleDb.OleDbCommand()
            objOleDbCommand.Connection = Me._objOleDbConnection
            objOleDbCommand.CommandText = "SELECT * FROM " & strTableName
            objOleDbDataAdapter = New System.Data.OleDb.OleDbDataAdapter()
            objOleDbDataAdapter.SelectCommand = objOleDbCommand
            objDataSet = New System.Data.DataSet()
            objDataSet.Reset()
            Call objOleDbDataAdapter.FillSchema(objDataSet, SchemaType.Mapped,
strTableName)
            Call objOleDbDataAdapter.Fill(objDataSet, strTableName)
            objOleDbDataAdapter = Nothing
            objOleDbCommand = Nothing
            Return True
        Catch ex As System.Exception
            Me._expLastException = ex
            Return False
        End Try
    End Function 'PopulateDataSet

    Public Function PopulateDataSet(ByVal strSQL As System.String, ByVal strTableName As
System.String, ByRef objDataSet As System.Data.DataSet) As System.Boolean
        Dim objOleDbCommand As System.Data.OleDb.OleDbCommand
        Dim objOleDbDataAdapter As System.Data.OleDb.OleDbDataAdapter
        Try
            objOleDbCommand = New System.Data.OleDb.OleDbCommand()
            objOleDbCommand.Connection = Me._objOleDbConnection
            objOleDbCommand.CommandText = strSQL
            objOleDbDataAdapter = New System.Data.OleDb.OleDbDataAdapter()
            objOleDbDataAdapter.SelectCommand = objOleDbCommand
            objDataSet = New System.Data.DataSet()
            objDataSet.Reset()
            Call objOleDbDataAdapter.FillSchema(objDataSet, SchemaType.Mapped,
strTableName)
            Call objOleDbDataAdapter.Fill(objDataSet, strTableName)
            objOleDbDataAdapter = Nothing
            objOleDbCommand = Nothing
            Return True
        Catch ex As System.Exception
            Me._expLastException = ex
            Return False
        End Try
    End Function 'PopulateDataSet (SQL)

    Public Function UpdateDataSet(ByVal strTableName As System.String, ByVal objDataSet As
System.Data.DataSet) As System.Boolean
        Dim objOleDbCommand As System.Data.OleDb.OleDbCommand
        Dim objOleDbDataAdapter As System.Data.OleDb.OleDbDataAdapter
        Dim objOleDbCommandBuilder As OleDb.OleDbCommandBuilder
        Try
            objOleDbCommand = New System.Data.OleDb.OleDbCommand()
            objOleDbCommand.Connection = Me._objOleDbConnection
            objOleDbCommand.CommandText = "SELECT * FROM " & strTableName
            objOleDbDataAdapter = New System.Data.OleDb.OleDbDataAdapter()
            objOleDbDataAdapter.SelectCommand = objOleDbCommand
            objOleDbCommandBuilder = New OleDb.OleDbCommandBuilder()
            objOleDbCommandBuilder.DataAdapter = objOleDbDataAdapter
            objOleDbDataAdapter.Update(objDataSet, strTableName)
            objDataSet.AcceptChanges()
            objOleDbCommandBuilder = Nothing
            objOleDbDataAdapter = Nothing
            objOleDbCommand = Nothing
            objDataSet = Nothing
            Return True
        Catch ex As System.Exception
            Me._expLastException = ex
            Return False
        End Try
    End Function 'UpdateDataSet

    Public Function UpdateField(ByVal strTableName As System.String, ByVal strFieldName As
System.String, ByVal strFieldNewValue As System.String, ByVal strCriteria As System.String)
As System.Boolean
```

```
        Dim objOleDbCommand As System.Data.OleDb.OleDbCommand
        Try
            objOleDbCommand = New System.Data.OleDb.OleDbCommand()
            objOleDbCommand.Connection = Me._objOleDbConnection
            objOleDbCommand.CommandText = "UPDATE " & strTableName & " SET " & strFieldName
& "=" & strFieldNewValue & " WHERE " & strCriteria
            objOleDbCommand.ExecuteNonQuery()
            objOleDbCommand = Nothing
            Return True
        Catch ex As System.Exception
            Me._expLastException = ex
            Return False
        End Try
    End Function 'UpdateField

    Public Function DeleteRecord(ByVal strTableName As System.String, ByVal strCriteria As
System.String) As System.Boolean
        Dim objOleDbCommand As System.Data.OleDb.OleDbCommand
        Try
            objOleDbCommand = New System.Data.OleDb.OleDbCommand()
            objOleDbCommand.Connection = Me._objOleDbConnection
            objOleDbCommand.CommandText = "DELETE FROM " & strTableName & " WHERE " &
strCriteria
            objOleDbCommand.ExecuteNonQuery()
            objOleDbCommand = Nothing
            Return True
        Catch ex As System.Exception
            Me._expLastException = ex
            Return False
        End Try
    End Function 'DeleteRecord

#End Region

#Region " Shared "

    Public Shared Function FilterString(ByVal strText As String) As String
        strText = strText.Replace(Chr(34), Chr(32))
        strText = strText.Replace(Chr(39), Chr(32))
        Return strText.Trim()
    End Function 'FilterString

#End Region

End Class 'clsDB
```

Η κλάση διαχείρισης των ρυθμίσεων της εφαρμογής χρησιμοποιεί το αντικείμενο XmlDocument από το System.Xml Namespace του .Net Framework για να διαχειριστεί τις ρυθμίσεις της εφαρμογής. Οι ρυθμίσεις αποθηκεύονται σε ένα αρχείο κειμένου και υπάρχει η δυνατότητα να κωδικοποιηθούν εάν το αντικείμενο της κλάσης αρχικοποιηθεί με κάποια είσοδο που θα αποτελεί το κλειδί της κωδικοποίησης.

3.2.1 – Οι Constructor της κλάσης clsOptions

```
Public Sub New(Optional ByVal strCipherKey As String = "")
```

Ο Constructor της κλάσης έχει προαιρετική είσοδο ένα αλφαριθμητικό που αποτελεί το κλειδί της κωδικοποίησης. Εάν αυτό είναι ίσο με το κενό αλφαριθμητικό τότε η κλάση δεν κωδικοποιεί τις ρυθμίσεις αλλά τις αποθηκεύει σαν απλό κείμενο.

3.2.2 – Οι Ιδιότητες της κλάσης clsOptions

```
Public ReadOnly Property OptionFilePath() As String
```

Μοναδική ιδιότητα της κλάσης είναι η OptionFilePath και επιστρέφει την πλήρης διαδρομή της θέσεως που βρίσκετε το αρχείο που κρατά τις ρυθμίσεις.

3.2.3 – Οι Μέθοδοι της κλάσης clsOptions

```
Public Function GetOption(ByVal strOptionName As String) As String
```

Η μέθοδος GetOption μας επιστρέφει ένα αλφαριθμητικό που περιέχει την τιμή της ρύθμισης που της δώσαμε σαν είσοδο. Η είσοδος είναι αλφαριθμητικού τύπου. Στην περίπτωση που η ρύθμιση δεν βρεθεί η μέθοδος επιστρέφει το κενό αλφαριθμητικό.

```
Public Sub SetOption(ByVal strOptionName As String, ByVal strOptionValue As String)
```

Η μέθοδος SetOption έχει δύο αλφαριθμητικά ως εισόδους, το όνομα της ρύθμισης και την τιμή της. Η μέθοδος ενημερώνει το XmlDocument με την νέα τιμή της ρύθμισης ή στην περίπτωση που η ρύθμιση δεν υπάρχει αναλαμβάνει να την δημιουργήσει και να της θέση την τιμή που της δώσαμε.

3.2.4 – Ο κώδικας της κλάσης clsOptions

```
Option Explicit On
Option Strict On

Public Class clsOptions

    Private strOptionsFile As String
    Private strCipherKey As String
    Private XmlDoc As Xml.XmlDocument

    Public Sub New(Optional ByVal strCipherKey As String = "")
        Me.strOptionsFile = My.Application.Info.DirectoryPath & "\" &
My.Application.Info.ProductName.Replace(" ", "") & ".opt"
        Me.strCipherKey = strCipherKey.Trim()
        Me.XmlDoc = New Xml.XmlDocument()
        Me.LoadOptions()
    End Sub 'New

    Protected Overrides Sub Finalize()
        If IO.File.Exists(Me.strOptionsFile) Then
            Me.SaveOptions()
        End If
        MyBase.Finalize()
    End Sub 'Finalize

    Public ReadOnly Property OptionFilePath() As String
        Get
            Return Me.strOptionsFile
        End Get
    End Property 'OptionFilePath

    Public Function GetOption(ByVal strOptionName As String) As String
        Dim aXmlNodeList As Xml.XmlNodeList
        Try
            aXmlNodeList = Me.XmlDoc.GetElementsByTagName(strOptionName)
            If aXmlNodeList.Count = 1 Then
                Return aXmlNodeList.ItemOf(0).InnerText
            Else
                Return ""
            End If
        Catch ex As Exception
            Return ""
        End Try
    End Function 'GetOption

    Public Sub SetOption(ByVal strOptionName As String, ByVal strOptionValue As String)
        Dim aXmlNodeList As Xml.XmlNodeList
        Dim aXmlElement As Xml.XmlElement
```

```
        Try
            aXmlNodeList = Me.XmlDoc.GetElementsByTagName(strOptionName)
            If aXmlNodeList.Count = 1 Then
                aXmlNodeList.ItemOf(0).InnerText = strOptionValue
            Else
                aXmlElement = Me.XmlDoc.CreateElement(strOptionName)
                aXmlElement.InnerText = strOptionValue
                Me.XmlDoc.DocumentElement.AppendChild(aXmlElement)
            End If
        Catch ex As Exception
        End Try
    End Sub 'SetOption

    Private Sub SetDefaults()
        Dim aXmlElement As Xml.XmlElement
        Try
            aXmlElement = Me.XmlDoc.CreateElement("xml")
            aXmlElement.InnerText = ""
            Me.XmlDoc.AppendChild(aXmlElement)
        Catch ex As Exception
        End Try
    End Sub 'SetDefaults

    Private Sub LoadOptions()
        Me.SetDefaults()
        Try
            If IO.File.Exists(Me.strOptionsFile) Then
                Dim aFileStream As IO.FileStream = IO.File.Open(Me.strOptionsFile,
IO.FileMode.Open, IO.FileAccess.Read, IO.FileShare.ReadWrite)
                Dim bytData(CInt(aFileStream.Length) - 1) As Byte
                aFileStream.Read(bytData, 0, bytData.Length)
                aFileStream.Close()
                aFileStream = Nothing
                Dim strData As String = ""
                If Me.strCipherKey.Length > 0 Then
                    If Not clsCipher.Decrypt(Convert.ToBase64String(bytData), strData,
Me.strCipherKey) Then
                        Err.Raise(vbObjectError + 513)
                    End If
                Else
                    Dim objUnicodeEncoding As New System.Text.UnicodeEncoding()
                    strData = objUnicodeEncoding.GetString(bytData)
                End If
                Me.XmlDoc.LoadXml(strData)
            Else
                Me.SaveOptions()
            End If
        Catch ex As Exception
            Me.SetDefaults()
            mdlMain.LogFile.WriteGlobalException("Η εφαρμογή δεν μπόρεσε να διαβάσει το
αρχείο ρυθμίσεων """ & Me.strOptionsFile & """.", "Error in clsOptions...", True)
        End Try
    End Sub 'LoadOptions

    Public Sub SaveOptions()
        Try
            IO.File.Delete(Me.strOptionsFile)
            Dim bytData() As Byte
            If Me.strCipherKey.Length > 0 Then
                Dim strData As String = ""
                If Not clsCipher.Encrypt(Me.XmlDoc.InnerXml, strData, Me.strCipherKey) Then
                    Err.Raise(vbObjectError + 513)
                End If
                bytData = Convert.FromBase64String(strData)
            Else
                Dim objUnicodeEncoding As New System.Text.UnicodeEncoding()
                bytData = objUnicodeEncoding.GetBytes(Me.XmlDoc.InnerXml)
            End If
            Dim aFileStream As IO.FileStream = IO.File.Open(Me.strOptionsFile,
IO.FileMode.Create, IO.FileAccess.Write, IO.FileShare.ReadWrite)
            aFileStream.Write(bytData, 0, bytData.Length)
            aFileStream.Flush()
            aFileStream.Close()
            aFileStream = Nothing
        Catch ex As Exception
            mdlMain.LogFile.WriteGlobalException("Η εφαρμογή δεν μπόρεσε να δημιουργήσει το
αρχείο ρυθμίσεων """ & Me.strOptionsFile & """.", "Error in clsOptions...", True)
```

```
        End Try
    End Sub 'SaveOptions

End Class 'clsOptions
```

Η κλάση κωδικοποίησης και αποκωδικοποίησης της εφαρμογής χρησιμοποιεί δύο κοινές μεθόδους, μία για την κωδικοποίηση και μία για την αποκωδικοποίηση. Με αυτόν τον τρόπο οι μέθοδοι αυτές μπορούν να χρησιμοποιηθούν χωρίς να δημιουργηθεί κάποιο αντικείμενο αυτής της κλάσης. Κάθε μία από αυτές τις μεθόδους έχει την δυνατότητα να χρησιμοποιήσει οποιαδήποτε από τις εσωτερικές υποκλάσεις για να επιτύχει την κωδικοποίηση και την αποκωδικοποίηση και αυτό επιλέγετε ανάλογα με τις εισόδους της.

Στο σημείο ανάπτυξης που βρίσκετε αυτήν την στιγμή η κλάση υπάρχει η δυνατότητα να γίνει κωδικοποίηση και αποκωδικοποίηση μόνο με τον συμμετρικό αλγόριθμο Rijndael ο οποίος ορίζετε στην υποκλάση clsRijndaelCipher. Βεβαίως υπάρχει η δυνατότητα μελλοντικής ανάπτυξης της κλάσης έτσι ώστε να υποστηρίζονται και άλλοι αλγόριθμοι, είτε συμμετρικοί είτε ασύμμετροι.

Η κλάση clsCipher χρησιμοποιεί αλφαριθμητικά για να κρατήσει το κείμενο προς κωδικοποίηση και το κωδικοποιημένο κείμενο. Το αλφαριθμητικό που κρατά το κωδικοποιημένο κείμενο χρησιμοποιεί τους 64 βασικούς χαρακτήρες που ορίζονται στα Base 64 Strings, δηλαδή τους κεφαλαίους λατινικούς χαρακτήρες από το ‘A’ έως το ‘Z’, τους πεζούς λατινικούς χαρακτήρες από το ‘a’ έως το ‘z’, τους αριθμούς από το ‘0’ έως το ‘9’ και τα σύμβολα ‘+’ και ‘/’. Επίσης χρησιμοποιείτε και ο χαρακτήρας ‘=’ ο οποίος αντιμετωπίζετε σαν να μην έχει απολύτως καμία τιμή και τοποθετείτε στο τέλος του αλφαριθμητικού όταν και αν χρειάζεται έτσι ώστε αυτό να έχει την σωστή μορφή που απαιτείτε για ένα Base 62 String.

3.3.1 – Οι Κοινές Μέθοδοι της κλάσης clsCipher

```
Public Shared Function Encrypt(ByVal strToEncrypt As System.String, ByRef strEncrypted As
System.String, Optional ByVal strKey As System.String = "", Optional ByVal enmCipherMode As
CipherModeEnum = CipherModeEnum.RijndaelCipherMode) As System.Boolean
```

Η κοινή μέθοδος κωδικοποίησης Encrypt έχει δύο βασικές εισόδους, το αλφαριθμητικό που θέλουμε να κωδικοποιήσουμε και το κωδικοποιημένο αλφαριθμητικό σε μορφή Base 64 String το οποίο εισάγετε ως διεύθυνση μνήμης έτσι ώστε να μπορούμε να δούμε τις αλλαγές σε αυτό και εξωτερικά από την μέθοδο. Εκτός από αυτές τις δύο εισόδους η μέθοδος μπορεί προαιρετικά να πάρει και άλλες δύο, το κλειδί της κωδικοποίησης και τον τύπο της μεθόδου που θα χρησιμοποιηθεί για αυτήν ο οποίος ορίζετε από τον τύπο CipherModeEnum. Στην περίπτωση που το κλειδί της κωδικοποίησης δεν οριστεί ή είναι το καινό αλφαριθμητικό τότε χρησιμοποιείται ένα εσωτερικό κλειδί το οποίο έχει οριστεί μέσα στην ανάλογη υποκλάση που πρόκειται να

χρησιμοποιηθεί. Η μέθοδος επιστρέφει True/False αναλόγως εάν εκτελέστηκε με επιτυχία ή όχι.

```
Public Shared Function Decrypt(ByVal strToDecrypt As System.String, ByRef strDecrypted As
System.String, Optional ByVal strKey As System.String = "", Optional ByVal enmCipherMode As
CipherModeEnum = CipherModeEnum.RijndaelCipherMode) As System.Boolean
```

Η δεύτερη κοινή μέθοδος, και τελευταία, της κλάσης είναι η Decrypt και έχει πανομοιότυπες εισόδους και λειτουργία με την Encrypt με την μόνη διαφορά ότι τα αλφαριθμητικά εισόδου αυτήν την φορά λειτουργούν αντίθετα. Δηλαδή το αλφαριθμητικό με το κείμενο προς αποκωδικοποίηση θα πρέπει να βρίσκετε σε μορφή Base 62 String και το αποκωδικοποιημένο αλφαριθμητικό εισάγετε ως διεύθυνση μνήμης. Η μέθοδος επιστρέφει και πάλι True/False ανάλογα με την επιτυχία της.

3.3.2 – Η υποκλάση clsRijndaelCipher

Η υποκλάση clsRijndaelCipher αντιπροσωπεύει τον συμμετρικό αλγόριθμο κωδικοποίησης Rijndael με κλειδί μεγέθους 256 bits.

```
Public Sub New(Optional ByVal strKey As System.String = "")
```

Ο Constructor της υποκλάσης έχει μία και μοναδική είσοδο, η οποία είναι και προαιρετική, ενός αλφαριθμητικού το οποίο θα χρησιμοποιηθεί ως κλειδί για τη κωδικοποίηση και την αποκωδικοποίηση. Όπως προαναφέρθηκε, η υποκλάση θα χρησιμοποιήσει το εσωτερικό της κλειδί στην περίπτωση που δεν εισάγουμε κάποιο στον constructor της.

```
Public Function RijndaelEncryptor(ByVal bytToEncrypt() As System.Byte, ByRef bytEncrypted()
As System.Byte) As System.Boolean
```

Η μέθοδος RijndaelEncryptor αναλαμβάνει την κωδικοποίηση με τον συμμετρικό αλγόριθμο Rijndael. Οι δύο είσοδοί της είναι πίνακες από Byte. Ο πρώτος περιέχει τα Byte τα οποία θέλουμε να κωδικοποιήσουμε και ο δεύτερος τα κωδικοποιημένα. Εξαιτίας του ότι τα κωδικοποιημένα Byte τα θέλουμε εκτός της μεθόδου, ο δεύτερος πίνακας εισάγετε στην μέθοδο ως διεύθυνση μνήμης. Η μέθοδος μας επιστρέφει True/False αναλόγως με την επιτυχία της.

```
Public Function RijndaelDecryptor(ByVal bytToDecrypt() As System.Byte, ByRef bytDecrypted()
As System.Byte) As System.Boolean
```

Ανάλογη είναι και η δεύτερη μέθοδος της κλάσης η RijndaelDecryptor με την διαφορά οι είσοδοι λειτουργούν ανάποδα. Δηλαδή ως διεύθυνση μνήμης εισάγετε ο πίνακας που θα περιέχει τα αποκωδικοποιημένα Bytes μετά το τέλος της μεθόδου. Η μέθοδος επιστρέφει και πάλι True/False για να μας ενημερώσει για την επιτυχία της ή την αποτυχία της.

3.3.3 – Ο κώδικας της κλάσης clsCipher

```
Option Explicit On
Option Strict On
```

```
''' <summary>
''' Class Cipher:
''' Contains 2 Public Shared functions (Encrypt and Decrypt).
''' Each function gets a string and returns a encrypted or decrypted string.
''' In additional each function optional gets a string key for the encryption or decryption
''' and the encryption or decryption mode (Rijndael, etc.)
''' Modes: Rijndael
''' </summary>
''' <remarks></remarks>
Public Class clsCipher

    Public Enum CipherModeEnum
        RijndaelCipherMode
    End Enum

    ''' <summary>
    ''' Encrypt Function from Class Cipher:
    ''' Gets a plain string (System.String), a reference to base64string (System.String),
    ''' (Optional) a key (System.String) and (Optional) a encryption mode
(clsCipher.CipherModeEnum)
    ''' and returns True or False (System.Boolean) representing the success of the method.
    ''' </summary>
    ''' <param name="strToEncrypt">The plain string to encrypt (System.String).</param>
    ''' <param name="strEncrypted">A reference to base64string encrypted with the given key
and mode (System.String).</param>
    ''' <param name="strKey">(Optional, Default = "") The key for the encryption
(System.String).</param>
    ''' <param name="enmCipherMode">(Optional, Default = RijndaelCipherMode) The mode of
the encryption (clsCipher.CipherModeEnum).</param>
    ''' <returns>True if method finish successfully, False if failure
(System.Boolean).</returns>
    ''' <remarks>Remarks for the Class Cipher Encrypt Function...</remarks>
    Public Shared Function Encrypt(ByVal strToEncrypt As System.String, ByRef strEncrypted
As System.String, Optional ByVal strKey As System.String = "", Optional ByVal enmCipherMode
As CipherModeEnum = CipherModeEnum.RijndaelCipherMode) As System.Boolean
        Dim objEncoding As System.Text.Encoding
        Dim bytEncrypted() As System.Byte = {}
        Select Case enmCipherMode
            Case CipherModeEnum.RijndaelCipherMode
                Dim RijndaelCipher As clsCipher.clsRijndaelCipher
                Try
                    objEncoding = New System.Text.UnicodeEncoding()
                    RijndaelCipher = New clsCipher.clsRijndaelCipher(strKey)
                    If Not
RijndaelCipher.RijndaelEncryptor(objEncoding.GetBytes(strToEncrypt), bytEncrypted) Then
                        Return False
                    End If
                    strEncrypted = System.Convert.ToBase64String(bytEncrypted)
                    Return True
                Catch ex As Exception
                    Return False
                End Try
            Case Else
                Return False
        End Select
    End Function 'Encrypt

    ''' <summary>
    ''' Decrypt Function from Class Cipher:
    ''' Gets a encrypted base64string (System.String), a reference to plain string
(System.String),
    ''' (Optional) a key (System.String) and (Optional) a decryption mode
(clsCipher.CipherModeEnum)
    ''' and returns True or False (System.Boolean) representing the success of the method.
    ''' </summary>
    ''' <param name="strToDecrypt">The base64string to decrypt (System.String).</param>
    ''' <param name="strDecrypted">A reference to plain string decrypted with the given key
and mode (System.String).</param>
    ''' <param name="strKey">(Optional, Default = "") The key for the decryption
(System.String).</param>
    ''' <param name="enmCipherMode">(Optional, Default = RijndaelCipherMode) The mode of
the decryption (clsCipher.CipherModeEnum).</param>
    ''' <returns>True if method finish successfully, False if failure
(System.Boolean).</returns>
    ''' <remarks>Remarks for the Class Cipher Decrypt Function...</remarks>
```

```
    Public Shared Function Decrypt(ByVal strToDecrypt As System.String, ByRef strDecrypted
As System.String, Optional ByVal strKey As System.String = "", Optional ByVal enmCipherMode
As CipherModeEnum = CipherModeEnum.RijndaelCipherMode) As System.Boolean
        Dim objEncoding As System.Text.Encoding
        Dim bytDecrypted() As System.Byte = {}
        Select Case enmCipherMode
            Case CipherModeEnum.RijndaelCipherMode
                Dim RijndaelCipher As clsCipher.clsRijndaelCipher
                Try
                    objEncoding = New System.Text.UnicodeEncoding()
                    RijndaelCipher = New clsCipher.clsRijndaelCipher(strKey)
                    If Not
RijndaelCipher.RijndaelDecryptor(System.Convert.FromBase64String(strToDecrypt),
bytDecrypted) Then
                        Return False
                    End If

                    Dim strTemp As String = objEncoding.GetString(bytDecrypted)
                    strDecrypted = ""
                    For i As Integer = 0 To strTemp.IndexOf(Chr(0)) - 1
                        strDecrypted &= strTemp.Substring(i, 1)
                    Next

                    Return True
                Catch ex As Exception
                    Return False
                End Try
            Case Else
                Return False
        End Select
    End Function 'Decrypt

#Region " CipherModes "

    '################################################
    '## RijndaelCipher Class
    '#==============================================
    Private Class clsRijndaelCipher

        '################################################
        '## Constructor
        '#==============================================
        Public Sub New(Optional ByVal strKey As System.String = "")
            If strKey.Length = 0 Then
                strKey = "WiSE"
            End If

            Me.InitializeRijndael(strKey)
            Me.bytRijndaelKey = Me.GenerateRijndaelKey(strKey)
            Me.bytRijndaelIV = Me.GenerateRijndaelIV(strKey)
        End Sub 'New

        '################################################
        '## Public Methods
        '#==============================================
        Public Function RijndaelEncryptor(ByVal bytToEncrypt() As System.Byte, ByRef
bytEncrypted() As System.Byte) As System.Boolean
            Dim myRijndael As System.Security.Cryptography.RijndaelManaged
            Dim myRijndaelEncryptor As System.Security.Cryptography.ICryptoTransform
            Dim msEncryptor As System.IO.MemoryStream
            Dim csEncryptor As System.Security.Cryptography.CryptoStream
            Try
                myRijndael = New System.Security.Cryptography.RijndaelManaged()
                myRijndael.KeySize = 256
                myRijndael.BlockSize = 256
                myRijndaelEncryptor = myRijndael.CreateEncryptor(Me.bytRijndaelKey,
Me.bytRijndaelIV)
                msEncryptor = New System.IO.MemoryStream()
                csEncryptor = New System.Security.Cryptography.CryptoStream(msEncryptor,
myRijndaelEncryptor, Security.Cryptography.CryptoStreamMode.Write)
                csEncryptor.Write(bytToEncrypt, 0, bytToEncrypt.Length)
                csEncryptor.FlushFinalBlock()
                bytEncrypted = msEncryptor.ToArray()
                Return True
            Catch ex As Exception
                Return False
            End Try
```

```
        End Function 'RijndaelEncryptor

        Public Function RijndaelDecryptor(ByVal bytToDecrypt() As System.Byte, ByRef
bytDecrypted() As System.Byte) As System.Boolean
            Dim myRijndael As System.Security.Cryptography.RijndaelManaged
            Dim myRijndaelDecryptor As System.Security.Cryptography.ICryptoTransform
            Dim msDecryptor As System.IO.MemoryStream
            Dim csDecryptor As System.Security.Cryptography.CryptoStream
            Dim bytFromDencrypt(bytToDecrypt.Length - 1) As System.Byte
            Try
                myRijndael = New System.Security.Cryptography.RijndaelManaged()
                myRijndael.KeySize = 256
                myRijndael.BlockSize = 256
                myRijndaelDecryptor = myRijndael.CreateDecryptor(Me.bytRijndaelKey,
Me.bytRijndaelIV)
                msDecryptor = New System.IO.MemoryStream(bytToDecrypt)
                msDecryptor.Position = 0
                csDecryptor = New System.Security.Cryptography.CryptoStream(msDecryptor,
myRijndaelDecryptor, Security.Cryptography.CryptoStreamMode.Read)
                csDecryptor.Read(bytFromDencrypt, 0, bytFromDencrypt.Length)
                bytDecrypted = bytFromDencrypt
                Return True
            Catch ex As Exception
                Return False
            End Try
        End Function 'RijndaelEncryptor

        '################################################
        '## Private Variables
        '#===============================================
        Private bytRijndaelKey() As System.Byte
        Private bytRijndaelIV() As System.Byte

        '################################################
        '## Private Methods
        '#===============================================
        Private Sub InitializeRijndael(ByVal strKey As System.String)
            Randomize(Rnd(-1))
            For i As System.Int32 = 0 To strKey.Length - 1
                Randomize(Rnd(-Rnd() * Asc(strKey.Chars(i))))
            Next
        End Sub 'InitializeRijndael

        Private Function GenerateRijndaelKey(ByVal strKey As System.String) As
System.Byte()
            Dim bytResult(31) As System.Byte

            For i As System.Int32 = 0 To 31
                bytResult(i) = CByte(Rnd(-Asc(strKey.Chars(i Mod strKey.Length))) * 256)
            Next

            Return bytResult
        End Function 'GenerateRijndaelKey

        Private Function GenerateRijndaelIV(ByVal strKey As System.String) As System.Byte()
            Dim bytResult(31) As System.Byte

            Dim strTemp As System.String = ""
            For i As System.Int32 = 1 To strKey.Length
                strTemp &= CStr(strKey.Chars(strKey.Length - i))
            Next
            strKey = strTemp

            For i As System.Int32 = 0 To 31
                bytResult(i) = CByte(Rnd(-Asc(strKey.Chars(i Mod strKey.Length))) * 256)
            Next

            Return bytResult
        End Function 'GenerateRijndaelIV

    End Class 'clsRijndaelCipher

#End Region

End Class 'clsCipher
```

Η κλάση που αναλαμβάνει να διαχειριστεί το αρχείο καταγραφής των γεγονότων της εφαρμογής χρησιμοποιεί την μέθοδο AppendAllText από το αντικείμενο File του System.IO Namespace από το .Net Framework. Η κλάση προσφέρει έναν αριθμό από μεθόδους με τις οποίες μπορούμε να καταγράψουμε διάφορα γεγονότα που συμβαίνουν στην εφαρμογή. Επίσης μέσω της κλάσης ελέγχετε το εάν η εφαρμογή λειτουργεί σε περιβάλλον αποσφαλμάτοσης, για να επιτευχθεί αυτό θα πρέπει στον φάκελο εκτέλεσης της εφαρμογής να υπάρχει ένα συγκεκριμένο αρχείο το οποίο ορίζετε μέσα στην κλάση.

3.4.1 – Οι Constructor της κλάσης clsLogFile

```
Public Sub New()
```

Ο Constructor της κλάσης δεν έχει καμία είσοδο και απλά αρχικοποιεί το αντικείμενο της κλάσης.

3.4.2 – Οι Ιδιότητες της κλάσης clsLogFile

```
Public ReadOnly Property IsInDebugMode() As Boolean
```

Μέσο της ιδιότητας IsInDebugMode ελέγχουμε εάν η εφαρμογή λειτουργεί σε περιβάλλον αποσφαλμάτοσης ή όχι.

3.4.3 – Οι Μέθοδοι της κλάσης clsLogFile

```
Public Sub WriteAction(ByVal objOwner As IWin32Window, ByVal strText As String)
```

Η μέθοδος WriteAction έχει σαν εισόδους ένα αντικείμενο που να καλύπτει το Interface IWin32Window, όπως είναι μία φόρμα, και ένα αλφαριθμητικό που περιέχει το κείμενο που θέλουμε να καταγράψουμε. Η μέθοδος καταγράφει το κείμενο της ενέργειας σε σχέση με το αντικείμενο.

```
Public Sub WriteGlobalException(ByVal strText As String, ByVal strCaption As String, ByVal
booShowMessagebox As Boolean)
```

Η μέθοδος WriteGlobalException έχει τρεις εισόδους. Το κείμενο και την επικεφαλίδα της εγγραφής που θέλουμε να κάνουμε και μία μεταβλητή τύπου Boolean για το εάν θα φανεί κάποιο παράθυρο μηνύματος προς τον χρήστη ή όχι.

```
Public Sub WriteDBException(ByVal objOwner As IWin32Window, ByVal strDBException As String)
```

Η μέθοδος WriteDBException έχει ίδιες εισόδους με την WriteAction με την διαφορά ότι αυτή χρησιμοποιείτε εάν η καταγραφή αφορά κάποιο λάθος σε σχέση με την βάση δεδομένων και την χαρακτηρίζει ως τέτοια.

```
Public Function WriteMessageBox(ByVal objOwner As IWin32Window, ByVal strText As String,
ByVal strCaption As String, ByVal enmMessageBoxButtons As MessageBoxButtons, ByVal
enmMessageBoxIcon As MessageBoxIcon) As DialogResult
```

Η μέθοδος WriteMessageBox χρησιμοποιείτε όταν θέλουμε να προβάλουμε κάποια μηνύματα ή κάποια παράθυρα διαλόγου προς τον χρήστη. Οι είσοδοί της είναι πέντε. Ένα αντικείμενο που να καλύπτει το IWin32Window Interface, το κείμενο του μηνύματος, η επικεφαλίδα του, τα κουμπιά που θα έχει και η εικόνα που θα φαίνεται. Η μέθοδος επιστρέφει την απάντηση του χρήστη στο παράθυρο διαλόγου.

```
Public Sub WriteNotFountMessageBox(ByVal objOwner As IWin32Window, ByVal intMessageBoxID As
Integer)
```

Τελευταία μέθοδος είναι η WriteNotFoyntMessageBox που καταγράφει και παρουσιάζει στον χρήστη ένα παράθυρο διαλόγου με ένα έτοιμο κείμενο που τον ενημερώνει ότι το μήνυμα με τον συγκεκριμένο κωδικό δεν βρέθηκε. Η μέθοδος έχει δύο εισόδους, ένα αντικείμενο του IWin32Window Interface και έναν Integer. Η μέθοδος έχει χρησιμότητα μόνο εάν η εφαρμογή χρησιμοποιεί κωδικούς για τα παράθυρα διαλόγου.

3.4.4 – Οι Κοινές Σταθερές της κλάσης clsLogFile

```
Public Const STR_CONTACT_PHONE As System.String = "6949207539"
```

Η κλάση περιέχει και μία κοινή σταθερά που δηλώνει τον τηλεφωνικό αριθμό επικοινωνίας με την εταιρία ανάπτυξης της εφαρμογής και μπορεί να χρησιμοποιηθεί χωρίς την δημιουργία κάποιου αντικειμένου της κλάσης. Η τιμή της σταθερής ορίζετε μέσα στον κώδικα της κλάσης.

3.4.5 – Ο κώδικας της κλάσης clsLogFile

```
Option Explicit On
Option Strict On

Public Class clsLogFile

    Private Const STR_TEST_FILE_NAME As System.String = "DebugModule.dll"
    Private booInInDebugMode As System.Boolean
    Private booLogFileIsOpen As System.Boolean
    Private strLogFile As String

    Public Sub New()
        Me.booInInDebugMode = IO.File.Exists(My.Application.Info.DirectoryPath & "\" &
clsLogFile.STR_TEST_FILE_NAME)
        Me.booLogFileIsOpen = False
        Me.strLogFile = My.Application.Info.DirectoryPath & "\" &
My.Application.Info.ProductName.Replace(" ", "") & ".log"
    End Sub 'New

    Protected Overrides Sub Finalize()
        MyBase.Finalize()
    End Sub 'Finalize

#Region " Public "

    Public Const STR_CONTACT_PHONE As System.String = "6949207539"

    Public ReadOnly Property IsInDebugMode() As Boolean
        Get
            Return Me.booInInDebugMode
        End Get
    End Property 'IsInDebugMode

    Public Sub WriteAction(ByVal objOwner As IWin32Window, ByVal strText As String)
        If Not Me.booLogFileIsOpen Then
            Me.OpenLogFile()
        End If
```

```
        Dim strData As String = "#" & CType(objOwner, Form).Name & " [Action] " & strText &
vbNewLine & _
            vbNewLine

        System.IO.File.AppendAllText(Me.strLogFile, strData, System.Text.Encoding.Default)
    End Sub 'WriteAction

    Public Sub WriteGlobalException(ByVal strText As String, ByVal strCaption As String,
ByVal booShowMessagebox As Boolean)
        Call Me.WriteMessageBoxLog(Nothing, strText, My.Application.Info.ProductName & " -
" & strCaption, MessageBoxButtons.OK, MessageBoxIcon.Error, booShowMessagebox)
    End Sub 'WriteGlobalException

    Public Sub WriteDBException(ByVal objOwner As IWin32Window, ByVal strDBException As
String)
        Call Me.WriteMessageBoxLog(objOwner, strDBException,
My.Application.Info.ProductName & " - DB Error...", MessageBoxButtons.OK,
MessageBoxIcon.Error, False)
    End Sub 'WriteDBException

    Public Function WriteMessageBox(ByVal objOwner As IWin32Window, ByVal strText As
String, ByVal strCaption As String, ByVal enmMessageBoxButtons As MessageBoxButtons, ByVal
enmMessageBoxIcon As MessageBoxIcon) As DialogResult
        Return Me.WriteMessageBoxLog(objOwner, strText, My.Application.Info.ProductName & "
- " & strCaption, enmMessageBoxButtons, enmMessageBoxIcon, True)
    End Function 'WriteMessageBox

    Public Sub WriteNotFountMessageBox(ByVal objOwner As IWin32Window, ByVal
intMessageBoxID As Integer)
        Call Me.WriteMessageBoxLog(objOwner, "Το μήνυμα δεν βρέθηκε!!!" & vbNewLine &
vbNewLine & "Παρακαλώ σημειώστε τον κωδικό του μηνύματος (" & intMessageBoxID.ToString() &
") και επικοινωνήστε μαζί μας στο τηλέφωνο " & clsLogFile.STR_CONTACT_PHONE & " (" &
My.Application.Info.CompanyName & ") για να διορθώσουμε το πρόβλημα.",
My.Application.Info.ProductName & " - " & "Κωδικός Μηνύματος: " &
intMessageBoxID.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Warning, True)
    End Sub 'WriteNotFountMessageBox

#End Region

#Region " Private "

    Private Function WriteMessageBoxLog(ByVal objOwner As IWin32Window, ByVal strText As
String, ByVal strCaption As String, ByVal enmMessageBoxButtons As MessageBoxButtons, ByVal
enmMessageBoxIcon As MessageBoxIcon, ByVal booShowMessagebox As Boolean) As DialogResult
        If Not Me.booLogFileIsOpen Then
            Me.OpenLogFile()
        End If

        Dim strData As String = "#"
        If objOwner IsNot Nothing Then
            strData &= CType(objOwner, Form).Name
        Else
            strData &= "GLOBAL"
        End If
        strData &= " - " & enmMessageBoxIcon.ToString() & vbNewLine & _
            strCaption & vbNewLine & _
            strText & vbNewLine & _
            enmMessageBoxButtons.ToString() & vbNewLine

        System.IO.File.AppendAllText(Me.strLogFile, strData, System.Text.Encoding.Default)

        Dim enmResult As DialogResult = DialogResult.None
        If booShowMessagebox Or Me.booInInDebugMode Then
            enmResult = MessageBox.Show(objOwner, strText, strCaption,
enmMessageBoxButtons, enmMessageBoxIcon)
            System.IO.File.AppendAllText(Me.strLogFile, enmResult.ToString() & vbNewLine,
System.Text.Encoding.Default)
        End If

        System.IO.File.AppendAllText(Me.strLogFile, vbNewLine,
System.Text.Encoding.Default)

        Return enmResult
    End Function 'WriteMessageBoxLog

    Private Sub OpenLogFile()
```

```
        Dim strData As String = _
            vbNewLine & _
            vbNewLine & _
            "  LOG STARTED ON " & System.DateTime.Now.ToLongDateString() & " - " &
System.DateTime.Now.ToLongTimeString() & vbNewLine & _

"================================================================================" &
vbNewLine & _
            vbNewLine

        System.IO.File.AppendAllText(Me.strLogFile, strData, System.Text.Encoding.Default)
        Me.booLogFileIsOpen = True
    End Sub 'OpenLogFile

#End Region

End Class 'clsLogFile
```

Το εργαλείο χρήστη ctlListView δημιουργήθηκε εξαιτίας της πολύ συχνής χρήσης λιστών για την προβολή διάφορων στοιχείων στον χρήστη. Το εργαλείο περιέχει ένα αντικείμενο ListView το οποίο αποτελεί την λίστα των εγγραφών που θέλουμε να παρουσιάσουμε στον χρήστη. Το εργαλείο χρήστη ctlListView μας προσφέρει διάφορες ιδιότητες και μεθόδους που μας επιτρέπουν εύκολα να ελέγξουμε τις εγγραφές της λίστας και τον τρόπο εμφάνισής τους. Επίσης προσφέρονται και κάποια γεγονότα με τα οποία μπορούμε να ελέγξουμε τις ενέργειες του χρήστη επάνω στην λίστα.

Μέσα στο ctlListView περιέχετε μία κρυφή κλάση η οποία ελέγχει την ταξινόμηση των εγγραφών αναλόγως με την επικεφαλίδα της λίστας που πάτησε ο χρήστης.

3.5.1 – Οι Constructor του εργαλείου χρήστη ctlListView

```
Public Sub New()
```

Ο Constructor του εργαλείου χρήστη δεν έχει καμία είσοδο και απλά αρχικοποιεί το αντικείμενο.

3.5.2 – Οι Ιδιότητες του εργαλείου χρήστη ctlListView

```
Public WriteOnly Property Title() As String
```

Στην ιδιότητα Title μπορούμε μόνο να γράψουμε και αποτελεί τον τίτλο του αντικειμένου. Ο τίτλος αυτός δεν είναι εμφανής στο περιβάλλον του χρήστη και χρησιμοποιείτε για την τιτλοφόρηση της εκτύπωσης της λίστας του αντικειμένου.

```
Public ReadOnly Property SelectedItemTag() As String
```

Η ιδιότητα SelectedItemTag μας επιστρέφει ένα αλφαριθμητικό που περιέχει την τιμή της ιδιότητας Tag της επιλεγμένης εγγραφής από τον χρήστη εάν αυτός έχει επιλέξει κάποια.

```
Public ReadOnly Property ListViewObject() As ListView
```

Η ιδιότητα ListViewObject μας επιστρέφει το αντικείμενο ListView που περιέχετε στο εργαλείο χρήστη.

```
Public WriteOnly Property ListViewObjectContextMenuStrip() As ContextMenuStrip
```

Στην ιδιότητα ListViewObjectContextMenuStrip μπορούμε να εισάγουμε ένα αντικείμενο τύπου ContextMenuStrip το οποίο περιέχει το αναδυόμενο μενού που εμφανίζετε όταν ο χρήστης κάνει δεξί κλικ σε κάποια εγγραφή της λίστας.

```
Public Property View() As View
```

Η ιδιότητα View μας επιτρέπει να ελέγχουμε τον τρόπο με τον οποίο θα φαίνονται οι εγγραφές στην λίστα.

```
Public Property ShowGroups() As Boolean
```

Η ιδιότητα ShowGroups μας επιτρέπει να ελέγχουμε το εάν οι εγγραφές της λίστας θα φαίνονται σε ομάδες ή όχι.

3.5.3 – Οι Μέθοδοι του εργαλείου χρήστη ctlListView

```
Public Sub ClearItems()
```

Η μέθοδος ClearItems καθαρίζει την λίστα από όλες τις εγγραφές.

```
Public Sub AddItem(ByVal objListViewItem As ListViewItem)
```

Η μέθοδος AddItem έχει είσοδο ένα αντικείμενο ListViewItem το οποίο το προσθέτει στην λίστα ως νέα εγγραφή.

```
Public Function ItemsCount() As Integer
```

Η μέθοδος ItemsCount μας επιστρέφει το πλήθος των εγγραφών που υπάρχουν την δεδομένη στιγμή στην λίστα.

```
Public Sub InitializeColumns(ByVal sctWiSEListViewColumns() As WiSEListViewColumn, Optional
ByVal enmView As View = Windows.Forms.View.Details)
```

Η μέθοδος InitializeColumns μας επιτρέπει να εισάγουμε νέες στήλες στην λίστα μας οι οποίες περιέχονται στον πίνακα sctWiSEListViewColumns που εισάγουμε κατά την κλήση της μεθόδου. Ο πίνακας που εισάγουμε περιέχει αντικείμενα τύπου WiSEListViewColumn. Κάθε τέτοιο αντικείμενο περιέχει μεταβλητές που ορίζουν το όνομα της στήλης, την επικεφαλίδα της, το πλάτος της και την στοίχιση που θα έχει η επικεφαλίδα της. Προαιρετικά μπορούμε να ορίσουμε και τον τρόπο εμφάνισης των εγγραφών στην λίστα.

```
Public Sub InitializeGroups(ByVal sctWiSEListViewGroups() As WiSEListViewGroup, Optional
ByVal booShowGroups As Boolean = True)
```

Η μέθοδος InitializeGroups είναι ανάλογη με την InitializeColumns και μας επιτρέπει να εισάγουμε νέες ομάδες. Ως είσοδο κατά την κλήση της μεθόδου χρησιμοποιούμε έναν πίνακα από αντικείμενα τύπου WiSEListViewGroup. Κάθε τέτοιο αντικείμενο περιέχει μεταβλητές που ορίζουν το όνομα της ομάδας καθώς και την επικεφαλίδα της. Προαιρετικά πάλι μπορούμε να ορίσουμε εάν θα φαίνονται οι ομάδες ή όχι.

3.5.4 – Τα γεγονότα του εργαλείου χρήστη ctlListView

```
Public Event evDoubleClick(ByVal strSelectedItemTag As String)
```

Το γεγονός evDoubleClick συμβαίνει όταν ο χρήστης κάνει διπλό κλικ σε κάποια εγγραφή της λίστας και μας επιστρέφει ένα αλφαριθμητικό με την τιμή της ιδιότητας Tag του αντικειμένου της εγγραφής που έκανε διπλό κλικ ο χρήστης.

```
Public Event evSelectOne(ByVal strSelectedItemTag As String)
```

Το γεγονός evSelectOne συμβαίνει όταν ο χρήστης κάνει κλικ, δηλαδή επιλέξει, σε κάποια εγγραφή της λίστας και μας επιστρέφει ένα αλφαριθμητικό με την τιμή της ιδιότητας Tag του αντικειμένου της εγγραφής που έκανε κλικ ο χρήστης.

```
Public Event evCheckOne(ByVal strSelectedItemTag As String, ByVal booCheckStatus As
Boolean)
```

Το γεγονός evCheckOne συμβαίνει όταν ο χρήστης αλλάξει την κατάσταση που βρίσκετε το κουτί τσεκαρίσματος κάποιας εγγραφής της λίστας εάν αυτό φαίνεται. Το γεγονός συμβαίνει προτού γίνει η αλλαγή στο περιβάλλον του χρήστη. Το γεγονός μας επιστρέφει ένα αλφαριθμητικό με την τιμή της ιδιότητας Tag του αντικειμένου της εγγραφής που άλλαξε ο χρήστης καθώς και την νέα τιμή (True/False) του κουτιού τσεκαρίσματος.

```
Public Event evCheckedOne(ByVal strSelectedItemTag As String, ByVal booCheckStatus As
Boolean)
```

Το γεγονός evCheckedOne συμβαίνει όταν ο χρήστης αλλάξει την κατάσταση που βρίσκετε το κουτί τσεκαρίσματος κάποιας εγγραφής της λίστας εάν αυτό φαίνεται. Το γεγονός συμβαίνει αφού γίνει η αλλαγή στο περιβάλλον του χρήστη. Το γεγονός μας επιστρέφει ένα αλφαριθμητικό με την τιμή της ιδιότητας Tag του αντικειμένου της εγγραφής που άλλαξε ο χρήστης καθώς και την νέα τιμή (True/False) του κουτιού τσεκαρίσματος.

```
Public Event evSelectNone()
```

Το γεγονός evSelectNone συμβαίνει όταν ο χρήστης κάνει κλικ έξω από κάποια εγγραφή της λίστας, δηλαδή αποεπιλέξει αυτήν που είχε επιλέξει πρωτύτερα.

3.5.5 – Ο κώδικας του εργαλείου χρήστη ctlListView

```
Option Explicit On
Option Strict On

Public Class ctlListView
    '################################################
    Public Event evDoubleClick(ByVal strSelectedItemTag As String)
    Public Event evSelectOne(ByVal strSelectedItemTag As String)
    Public Event evCheckOne(ByVal strSelectedItemTag As String, ByVal booCheckStatus As
Boolean)
    Public Event evCheckedOne(ByVal strSelectedItemTag As String, ByVal booCheckStatus As
Boolean)
    Public Event evSelectNone()
    '################################################
    Public WriteOnly Property Title() As String
        Set(ByVal value As String)
            Me.lstWiSEListView.Tag = value
        End Set
    End Property 'Title

    Public ReadOnly Property SelectedItemTag() As String
```

```
        Get
            If Me.lstWiSEListView.SelectedItems.Count = 1 Then
                Return Me.lstWiSEListView.SelectedItems.Item(0).Tag.ToString()
            Else
                Return ""
            End If
        End Get
    End Property 'SelectedItemTag
    '################################################
    Public ReadOnly Property ListViewObject() As ListView
        Get
            Return Me.lstWiSEListView
        End Get
    End Property 'ListViewObject

    Public WriteOnly Property ListViewObjectContextMenuStrip() As ContextMenuStrip
        Set(ByVal value As ContextMenuStrip)
            Me.lstWiSEListView.ContextMenuStrip = value
        End Set
    End Property 'ListViewObjectContextMenuStrip

    Public Sub ClearItems()
        Me.lstWiSEListView.Items.Clear()
    End Sub 'ClearItems

    Public Sub AddItem(ByVal objListViewItem As ListViewItem)
        Me.lstWiSEListView.Items.Add(objListViewItem)
    End Sub 'AddItem

    Public Function ItemsCount() As Integer
        Return Me.lstWiSEListView.Items.Count
    End Function 'ItemsCount
    '################################################
    Public Structure WiSEListViewColumn
        Public strName As String
        Public strText As String
        Public intWidth As Integer
        Public enmTextAlign As HorizontalAlignment
    End Structure

    Public Sub InitializeColumns(ByVal sctWiSEListViewColumns() As WiSEListViewColumn,
Optional ByVal enmView As View = Windows.Forms.View.Details)
        Dim objColumnHeader As ColumnHeader
        For Each sctWiSEListViewColumn As WiSEListViewColumn In sctWiSEListViewColumns
            objColumnHeader = New ColumnHeader()
            objColumnHeader.Name = sctWiSEListViewColumn.strName
            objColumnHeader.Tag = sctWiSEListViewColumn.strName
            objColumnHeader.Text = sctWiSEListViewColumn.strText
            objColumnHeader.Width = sctWiSEListViewColumn.intWidth
            objColumnHeader.TextAlign = sctWiSEListViewColumn.enmTextAlign
            Me.lstWiSEListView.Columns.Add(objColumnHeader)
        Next
        Me.LoadOptions()
        Me.lstWiSEListView.View = enmView
    End Sub 'InitializeColumns

    Public Property View() As View
        Get
            Return Me.lstWiSEListView.View
        End Get
        Set(ByVal value As View)
            Me.lstWiSEListView.View = value
        End Set
    End Property 'View
    '################################################
    Public Structure WiSEListViewGroup
        Public strKey As String
        Public strHeaderText As String
    End Structure

    Public Sub InitializeGroups(ByVal sctWiSEListViewGroups() As WiSEListViewGroup,
Optional ByVal booShowGroups As Boolean = True)
        For Each sctWiSEListViewGroup As WiSEListViewGroup In sctWiSEListViewGroups
            Me.lstWiSEListView.Groups.Add(sctWiSEListViewGroup.strKey,
sctWiSEListViewGroup.strHeaderText)
        Next
        Me.lstWiSEListView.ShowGroups = booShowGroups
```

```
    End Sub 'InitializeGroups

    Public Property ShowGroups() As Boolean
        Get
            Return Me.lstWiSEListView.ShowGroups
        End Get
        Set(ByVal value As Boolean)
            Me.lstWiSEListView.ShowGroups = value
        End Set
    End Property 'ShowGroups
    '################################################
    Private Sub LoadOptions()
        Dim intColumnHeaderWidth As Integer
        For Each objColumnHeader As ColumnHeader In Me.lstWiSEListView.Columns
            If Options.GetOption(objColumnHeader.Tag.ToString() & ".Width").Trim().Length =
0 Then
                intColumnHeaderWidth = objColumnHeader.Width
                Options.SetOption(objColumnHeader.Tag.ToString() & ".Width",
CStr(intColumnHeaderWidth))
            Else
                intColumnHeaderWidth =
CInt(Options.GetOption(objColumnHeader.Tag.ToString() & ".Width"))
            End If
            objColumnHeader.Width = intColumnHeaderWidth
        Next
    End Sub 'LoadOptions

    Private Sub lstWiSEListView_ColumnWidthChanged(ByVal sender As Object, ByVal e As
System.Windows.Forms.ColumnWidthChangedEventArgs) Handles
lstWiSEListView.ColumnWidthChanged
        Dim objColumnHeader As ColumnHeader =
Me.lstWiSEListView.Columns.Item(e.ColumnIndex)
        Options.SetOption(objColumnHeader.Tag.ToString() & ".Width",
CStr(objColumnHeader.Width))
    End Sub 'lstWiSEListView_ColumnWidthChanged

    Private Sub lstWiSEListView_ColumnClick(ByVal sender As Object, ByVal e As
ColumnClickEventArgs) Handles lstWiSEListView.ColumnClick
        Static Dim booASCOrder As Boolean = False
        booASCOrder = Not booASCOrder
        Me.lstWiSEListView.ListViewItemSorter = New ListViewItemComparer(e.Column,
booASCOrder)
    End Sub 'lstWiSEListView_ColumnClick
    '################################################
    Private Sub lstWiSEListView_GotFocus(ByVal sender As Object, ByVal e As
System.EventArgs) Handles lstWiSEListView.GotFocus
        If Me.lstWiSEListView.Items.Count > 0 Then
            If Me.lstWiSEListView.SelectedItems.Count = 0 Then
                Me.lstWiSEListView.Items.Item(0).Selected = True
            End If
        End If
    End Sub 'lstWiSEListView_GotFocus

    Private Sub lstWiSEListView_DoubleClick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles lstWiSEListView.DoubleClick
        If Me.lstWiSEListView.SelectedItems.Count = 1 Then
            RaiseEvent
evDoubleClick(Me.lstWiSEListView.SelectedItems.Item(0).Tag.ToString())
        End If
    End Sub 'lstWiSEListView_DoubleClick

    Private Sub lstWiSEListView_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles lstWiSEListView.SelectedIndexChanged
        If Me.lstWiSEListView.SelectedItems.Count = 1 Then
            RaiseEvent evSelectOne(Me.lstWiSEListView.SelectedItems.Item(0).Tag.ToString())
        Else
            RaiseEvent evSelectNone()
        End If
    End Sub 'lstWiSEListView_SelectedIndexChanged

    Private Sub lstWiSEListView_ItemCheck(ByVal sender As Object, ByVal e As
System.Windows.Forms.ItemCheckEventArgs) Handles lstWiSEListView.ItemCheck
        RaiseEvent evCheckOne(Me.lstWiSEListView.Items.Item(e.Index).Tag.ToString(),
(e.NewValue = CheckState.Checked))
    End Sub 'lstWiSEListView_ItemCheck
```

```
    Private Sub lstWiSEListView_ItemChecked(ByVal sender As Object, ByVal e As
System.Windows.Forms.ItemCheckedEventArgs) Handles lstWiSEListView.ItemChecked
        RaiseEvent evCheckedOne(e.Item.Tag.ToString(), e.Item.Checked)
    End Sub 'lstWiSEListView_ItemChecked
    '################################################
    Private Class ListViewItemComparer
        Implements IComparer

        Private intColumn As Integer
        Private booASCOrder As Boolean

        Public Sub New()
            Me.intColumn = 0
            Me.booASCOrder = True
        End Sub 'New

        Public Sub New(ByVal intColumn As Integer, ByVal booASCOrder As Boolean)
            Me.intColumn = intColumn
            Me.booASCOrder = booASCOrder
        End Sub 'New

        Public Function Compare(ByVal X As Object, ByVal Y As Object) As Integer Implements
IComparer.Compare
            If Me.booASCOrder Then
                If IsDate(CType(X, ListViewItem).SubItems(Me.intColumn).Text) And
IsDate(CType(Y, ListViewItem).SubItems(Me.intColumn).Text) Then
                    Return [DateTime].Compare(CDate(CType(X,
ListViewItem).SubItems(Me.intColumn).Text), CDate(CType(Y,
ListViewItem).SubItems(Me.intColumn).Text))
                Else
                    Return [String].Compare(CType(X,
ListViewItem).SubItems(Me.intColumn).Text, CType(Y,
ListViewItem).SubItems(Me.intColumn).Text)
                End If
            Else
                If IsDate(CType(X, ListViewItem).SubItems(Me.intColumn).Text) And
IsDate(CType(Y, ListViewItem).SubItems(Me.intColumn).Text) Then
                    Return [DateTime].Compare(CDate(CType(Y,
ListViewItem).SubItems(Me.intColumn).Text), CDate(CType(X,
ListViewItem).SubItems(Me.intColumn).Text))
                Else
                    Return [String].Compare(CType(Y,
ListViewItem).SubItems(Me.intColumn).Text, CType(X,
ListViewItem).SubItems(Me.intColumn).Text)
                End If
            End If
        End Function 'Compare

    End Class 'ListViewItemComparer

End Class 'ctlWiSEListView
```

Η κλάση εκτύπωσης λίστας μας δίνει την δυνατότητα να εκτυπώσουμε τα περιεχόμενα ενός αντικειμένου ListView σε έναν πίνακα με αυτοματοποιημένο τρόπο. Η κλάση προσφέρει δύο μεθόδους μία για την εκτύπωση και μία για την προεπισκόπηση της εκτύπωσης. Κατά την δημιουργία του πίνακα που θα εκτυπωθεί η κλάση αυτόματα υπολογίζει το πλήθος των σελίδων που θα χρειαστούν για την εκτύπωσή του αναλόγως με το πλήθος των εγγραφών που περιέχει και αναλόγως με το πλήθος και τις διαστάσεις που έχουν τα πεδία κάθε εγγραφής αριθμώντας κατάλληλα κάθε σελίδα με δύο τρόπους. Πρώτον με τον αριθμό της σελίδας σε σχέση με το πλήθος των εγγραφών και δεύτερων με το τμήμα των πεδίων που φαίνονται στην συγκεκριμένη σελίδα.

3.6.1 – Οι Constructor της κλάσης clsPrintList

```
Public Sub New()
```

Ο Constructor της κλάσης δεν έχει καμία είσοδο και απλά αρχικοποιεί το αντικείμενο της κλάσης.

3.6.2 – Οι Μέθοδοι της κλάσης clsPrintList

```
Public Sub ShowPrintPreview(ByVal objListView As ListView, ByVal
booUseCenteredColumnsHeaders As Boolean, ByVal owner As IWin32Window)
```

Η μέθοδος ShowPrintPreview μας δείχνει μια προεπισκόπηση αυτών που πρόκειται να εκτυπώσουμε. Ως εισόδους η μέθοδος έχει το αντικείμενο ListView που περιέχει τα στοιχεία που θέλουμε να εκτυπώσουμε, μία μεταβλητή τύπου Boolean που ελέγχει εάν οι επικεφαλίδες θα έχουν στοίχιση στο κέντρο ή θα διατηρήσουν την στοίχιση που έχουν στο αντικείμενο ListView και τέλος ένα αντικείμενο που να καλύπτει το Interface IWin32Window, όπως παραδείγματος χάρη μία φόρμα, στο οποίο θα ανήκει το παράθυρο διαλόγου της προεπισκόπησης της εκτύπωσης.

```
Public Sub ShowPrintDialog(ByVal objListView As ListView, ByVal
booUseCenteredColumnsHeaders As Boolean, ByVal owner As IWin32Window)
```

Η μέθοδος ShowPrintDialog είναι ολόιδια με την μέθοδο ShowPrintPreview με την διαφορά ότι το παράθυρο διαλόγου αφορά την εκτύπωση και όχι την προεπισκόπηση αυτής.

3.6.3 – Ο κώδικας της κλάσης clsPrintList

```
Option Explicit On
Option Strict On

Public Class clsPrintList

    Private WithEvents objPrintDocument As New System.Drawing.Printing.PrintDocument()
    Private WithEvents objPrintDialog As New System.Windows.Forms.PrintDialog()
    Private WithEvents objPrintPreview As New System.Windows.Forms.PrintPreviewDialog()
    Private objListView As ListView
    Private booUseCenteredColumnsHeaders As Boolean

    Public Sub New()
        Me.objPrintDialog.Document = Me.objPrintDocument
        Me.objPrintPreview.Document = Me.objPrintDocument
        Me.SetDefaultSettings()
    End Sub 'New

    Private Sub SetDefaultSettings()
        Me.objPrintDocument.DefaultPageSettings.PaperSize.RawKind =
System.Drawing.Printing.PaperKind.A4
        Me.objPrintDocument.DefaultPageSettings.Margins.Left = 0
        Me.objPrintDocument.DefaultPageSettings.Margins.Right = 0
        Me.objPrintDocument.DefaultPageSettings.Margins.Top = 0
        Me.objPrintDocument.DefaultPageSettings.Margins.Bottom = 0
        Me.objPrintDocument.DefaultPageSettings.Landscape = True
    End Sub 'SetDefaultSettings

    Public Sub ShowPrintPreview(ByVal objListView As ListView, ByVal
booUseCenteredColumnsHeaders As Boolean, ByVal owner As IWin32Window)
        Me.objListView = objListView
        Me.booUseCenteredColumnsHeaders = booUseCenteredColumnsHeaders
        Me.objPrintPreview.ShowDialog(owner)
    End Sub 'ShowPrintPreview

    Public Sub ShowPrintDialog(ByVal objListView As ListView, ByVal
booUseCenteredColumnsHeaders As Boolean, ByVal owner As IWin32Window)
        Me.objListView = objListView
```

```
        Me.booUseCenteredColumnsHeaders = booUseCenteredColumnsHeaders
        If objPrintDialog.ShowDialog(owner) = DialogResult.OK Then
            Me.objPrintDocument.Print()
        End If
    End Sub 'ShowPrintDialog

    Private Sub objPrintDocument_PrintPage(ByVal sender As System.Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles objPrintDocument.PrintPage
        Static Dim intNextColumnHeader As Integer = 0
        Static Dim intNextColumnRecord As Integer = 0
        Static Dim intCurrentPage As Integer = 1
        Static Dim intCurrentLine As Integer = 0
        Static Dim intCurrentPart As Integer = 0

        Dim objBoldFont As New Font("Microsoft Sans Serif", 10, FontStyle.Bold)
        Dim objFont As New Font("Microsoft Sans Serif", 10, FontStyle.Regular)
        Dim objBoldPen As New Pen(Color.Black, 2)
        Dim objPen As New Pen(Color.Black, 1)

        'LineAlignment
        '  Near = Top
        '  Center = Middle
        '  Far = Bottom
        'Alignment
        '  Near = Left
        '  Center = Center
        '  Far = Right
        Dim sfLeft As New StringFormat()
        sfLeft.LineAlignment = StringAlignment.Center
        sfLeft.Alignment = StringAlignment.Near
        Dim sfCenter As New StringFormat()
        sfCenter.LineAlignment = StringAlignment.Center
        sfCenter.Alignment = StringAlignment.Center
        Dim sfRight As New StringFormat()
        sfRight.LineAlignment = StringAlignment.Center
        sfRight.Alignment = StringAlignment.Far

        Dim intWidth As Integer =
CInt(Fix(objPrintDocument.DefaultPageSettings.PrintableArea.Width))
        Dim intHeight As Integer =
CInt(Fix(objPrintDocument.DefaultPageSettings.PrintableArea.Height))

        If objPrintDocument.DefaultPageSettings.Landscape Then
            Dim intTemp As Integer
            intTemp = intWidth
            intWidth = intHeight
            intHeight = intTemp
        End If

        e.HasMorePages = False

        Dim intLineHeight As Integer = CInt(objBoldFont.GetHeight(e.Graphics))
        Dim intLinesPerPage As Integer = (intHeight - 5 * intLineHeight) \ intLineHeight
        Dim intTotalPages As Integer = (Me.objListView.Items.Count \ intLinesPerPage) + 1
        Dim intTotalWidth As Integer = 30

        e.Graphics.DrawRectangle(objBoldPen, New Rectangle(0, 0, intWidth, intHeight))

        e.Graphics.DrawString(Date.Now.ToLongDateString() & " - " &
Date.Now.ToLongTimeString().Substring(0, 5), objBoldFont, Brushes.Black, New RectangleF(0,
0 * intLineHeight, intWidth - 200, intLineHeight), sfLeft)

        If intTotalPages > 1 Then
            e.Graphics.DrawString("Σελίδα " & intCurrentPage.ToString() & " από " &
intTotalPages.ToString(), objFont, Brushes.Black, New RectangleF(intWidth - 200, 0 *
intLineHeight, 200, intLineHeight), sfRight)
        End If


e.Graphics.DrawString(Me.objListView.Tag.ToString().Split("|"c).GetValue(0).ToString(),
objBoldFont, Brushes.Black, New RectangleF(0, 1 * intLineHeight, intWidth - 200,
intLineHeight), sfLeft)

e.Graphics.DrawString(Me.objListView.Tag.ToString().Split("|"c).GetValue(1).ToString(),
objFont, Brushes.Black, New RectangleF(0, 2 * intLineHeight, intWidth, intLineHeight),
sfLeft)
        e.Graphics.DrawLine(objBoldPen, 0, 3 * intLineHeight, intWidth, 3 * intLineHeight)
```

```
        e.Graphics.DrawString("A/A", objBoldFont, Brushes.Black, New RectangleF(0, 3 *
intLineHeight + intLineHeight \ 2, intTotalWidth, intLineHeight), sfCenter)
        Dim i As Integer
        For i = intNextColumnHeader To Me.objListView.Columns.Count - 1
            If i = intNextColumnHeader And Me.objListView.Columns.Item(i).Width > intWidth
Then
                e.Graphics.DrawLine(objBoldPen, intTotalWidth, 3 * intLineHeight,
intTotalWidth, intHeight)
                If Me.booUseCenteredColumnsHeaders Then
                    e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont,
Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2,
intWidth, intLineHeight), sfCenter)
                Else
                    Select Case Me.objListView.Columns.Item(i).TextAlign
                        Case HorizontalAlignment.Left
                            e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text,
objBoldFont, Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight
\ 2, intWidth, intLineHeight), sfLeft)
                        Case HorizontalAlignment.Center
                            e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text,
objBoldFont, Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight
\ 2, intWidth, intLineHeight), sfCenter)
                        Case HorizontalAlignment.Right
                            e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text,
objBoldFont, Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight
\ 2, intWidth, intLineHeight), sfRight)
                    End Select
                End If
                If i < Me.objListView.Columns.Count - 1 Then
                    intNextColumnHeader = i + 1
                    e.Graphics.DrawString("Τμήμα " & Chr(Asc("A") + intCurrentPart),
objFont, Brushes.Black, New RectangleF(intWidth - 200, 1 * intLineHeight, 200,
intLineHeight), sfRight)
                    intCurrentPart += 1
                    e.HasMorePages = True
                    Exit For
                Else
                    If intCurrentPart > 0 Then
                        e.Graphics.DrawString("Τμήμα " & Chr(Asc("A") + intCurrentPart),
objFont, Brushes.Black, New RectangleF(intWidth - 200, 1 * intLineHeight, 200,
intLineHeight), sfRight)
                    End If
                    intNextColumnHeader = i + 1
                    Exit For
                End If
            Else
                If intTotalWidth + Me.objListView.Columns.Item(i).Width > intWidth Then
                    intNextColumnHeader = i
                    e.Graphics.DrawString("Τμήμα " & Chr(Asc("A") + intCurrentPart),
objFont, Brushes.Black, New RectangleF(intWidth - 200, 1 * intLineHeight, 200,
intLineHeight), sfRight)
                    intCurrentPart += 1
                    e.HasMorePages = True
                    Exit For
                Else
                    e.Graphics.DrawLine(objBoldPen, intTotalWidth, 3 * intLineHeight,
intTotalWidth, intHeight)
                    If i = Me.objListView.Columns.Count - 1 Then
                        If Me.booUseCenteredColumnsHeaders Then
                            e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text,
objBoldFont, Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight
\ 2, intWidth - intTotalWidth, intLineHeight), sfCenter)
                        Else
                            Select Case Me.objListView.Columns.Item(i).TextAlign
                                Case HorizontalAlignment.Left

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2, intWidth - intTotalWidth,
intLineHeight), sfLeft)
                                Case HorizontalAlignment.Center

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2, intWidth - intTotalWidth,
intLineHeight), sfCenter)
                                Case HorizontalAlignment.Right
```

```
e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2, intWidth - intTotalWidth,
intLineHeight), sfRight)
                            End Select
                        End If
                        If intCurrentPart > 0 Then
                            e.Graphics.DrawString("Τμήμα " & Chr(Asc("A") +
intCurrentPart), objFont, Brushes.Black, New RectangleF(intWidth - 200, 1 * intLineHeight,
200, intLineHeight), sfRight)
                        End If
                    Else
                        If intTotalWidth + Me.objListView.Columns.Item(i).Width +
Me.objListView.Columns.Item(i + 1).Width > intWidth Then
                            If Me.booUseCenteredColumnsHeaders Then
                                e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text,
objBoldFont, Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight
\ 2, intWidth - intTotalWidth, intLineHeight), sfCenter)
                            Else
                                Select Case Me.objListView.Columns.Item(i).TextAlign
                                    Case HorizontalAlignment.Left

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2, intWidth - intTotalWidth,
intLineHeight), sfLeft)
                                    Case HorizontalAlignment.Center

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2, intWidth - intTotalWidth,
intLineHeight), sfCenter)
                                    Case HorizontalAlignment.Right

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2, intWidth - intTotalWidth,
intLineHeight), sfRight)
                                End Select
                            End If
                        Else
                            If Me.booUseCenteredColumnsHeaders Then
                                e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text,
objBoldFont, Brushes.Black, New RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight
\ 2, Me.objListView.Columns.Item(i).Width, intLineHeight), sfCenter)
                            Else
                                Select Case Me.objListView.Columns.Item(i).TextAlign
                                    Case HorizontalAlignment.Left

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2,
Me.objListView.Columns.Item(i).Width, intLineHeight), sfLeft)
                                    Case HorizontalAlignment.Center

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2,
Me.objListView.Columns.Item(i).Width, intLineHeight), sfCenter)
                                    Case HorizontalAlignment.Right

e.Graphics.DrawString(Me.objListView.Columns.Item(i).Text, objBoldFont, Brushes.Black, New
RectangleF(intTotalWidth, 3 * intLineHeight + intLineHeight \ 2,
Me.objListView.Columns.Item(i).Width, intLineHeight), sfRight)
                                End Select
                            End If
                        End If
                    End If
                    intTotalWidth += Me.objListView.Columns.Item(i).Width
                End If
            End If
        Next
        If i = Me.objListView.Columns.Count Then
            intNextColumnHeader = Me.objListView.Columns.Count
        End If
        e.Graphics.DrawLine(objBoldPen, 0, 5 * intLineHeight, intWidth, 5 * intLineHeight)

        Dim r As Integer
        For r = intCurrentLine To Me.objListView.Items.Count - 1
            If r >= intCurrentPage * intLinesPerPage Then
                If intNextColumnHeader = Me.objListView.Columns.Count Then
                    intCurrentLine = r
```

```
                End If
                Exit For
            End If
            intTotalWidth = 30
            e.Graphics.DrawString((r + 1).ToString(), objFont, Brushes.Black, New
RectangleF(0, ((r - (intCurrentPage - 1) * intLinesPerPage) + 5) * intLineHeight,
intTotalWidth, intLineHeight), sfCenter)
            For i = intNextColumnRecord To intNextColumnHeader - 1
                If i = intNextColumnHeader - 1 Then
                    Select Case Me.objListView.Columns.Item(i).TextAlign
                        Case HorizontalAlignment.Left

e.Graphics.DrawString(Me.objListView.Items.Item(r).SubItems.Item(i).Text, objFont,
Brushes.Black, New RectangleF(intTotalWidth, ((r - (intCurrentPage - 1) * intLinesPerPage)
+ 5) * intLineHeight, intWidth - intTotalWidth, intLineHeight), sfLeft)
                        Case HorizontalAlignment.Center

e.Graphics.DrawString(Me.objListView.Items.Item(r).SubItems.Item(i).Text, objFont,
Brushes.Black, New RectangleF(intTotalWidth, ((r - (intCurrentPage - 1) * intLinesPerPage)
+ 5) * intLineHeight, intWidth - intTotalWidth, intLineHeight), sfCenter)
                        Case HorizontalAlignment.Right

e.Graphics.DrawString(Me.objListView.Items.Item(r).SubItems.Item(i).Text, objFont,
Brushes.Black, New RectangleF(intTotalWidth, ((r - (intCurrentPage - 1) * intLinesPerPage)
+ 5) * intLineHeight, intWidth - intTotalWidth, intLineHeight), sfRight)
                    End Select
                Else
                    Select Case Me.objListView.Columns.Item(i).TextAlign
                        Case HorizontalAlignment.Left

e.Graphics.DrawString(Me.objListView.Items.Item(r).SubItems.Item(i).Text, objFont,
Brushes.Black, New RectangleF(intTotalWidth, ((r - (intCurrentPage - 1) * intLinesPerPage)
+ 5) * intLineHeight, CInt(IIf(Me.objListView.Columns.Item(i).Width > intWidth, intWidth,
Me.objListView.Columns.Item(i).Width)), intLineHeight), sfLeft)
                        Case HorizontalAlignment.Center

e.Graphics.DrawString(Me.objListView.Items.Item(r).SubItems.Item(i).Text, objFont,
Brushes.Black, New RectangleF(intTotalWidth, ((r - (intCurrentPage - 1) * intLinesPerPage)
+ 5) * intLineHeight, CInt(IIf(Me.objListView.Columns.Item(i).Width > intWidth, intWidth,
Me.objListView.Columns.Item(i).Width)), intLineHeight), sfCenter)
                        Case HorizontalAlignment.Right

e.Graphics.DrawString(Me.objListView.Items.Item(r).SubItems.Item(i).Text, objFont,
Brushes.Black, New RectangleF(intTotalWidth, ((r - (intCurrentPage - 1) * intLinesPerPage)
+ 5) * intLineHeight, CInt(IIf(Me.objListView.Columns.Item(i).Width > intWidth, intWidth,
Me.objListView.Columns.Item(i).Width)), intLineHeight), sfRight)
                    End Select
                    intTotalWidth += Me.objListView.Columns.Item(i).Width
                End If
            Next
            If r > intCurrentLine Then
                e.Graphics.DrawLine(objPen, 0, ((r - (intCurrentPage - 1) *
intLinesPerPage) + 5) * intLineHeight, intWidth, ((r - (intCurrentPage - 1) *
intLinesPerPage) + 5) * intLineHeight)
            End If
        Next
        If ((r - (intCurrentPage - 1) * intLinesPerPage)) < intLinesPerPage Then
            e.Graphics.DrawLine(objPen, 0, ((r - (intCurrentPage - 1) * intLinesPerPage) +
5) * intLineHeight, intWidth, ((r - (intCurrentPage - 1) * intLinesPerPage) + 5) *
intLineHeight)
        End If
        intNextColumnRecord = intNextColumnHeader

        If intCurrentPage < intTotalPages Then
            If intNextColumnHeader = Me.objListView.Columns.Count Then
                intNextColumnHeader = 0
                intNextColumnRecord = 0
                intCurrentPart = 0
                intCurrentPage += 1
                e.HasMorePages = True
            End If
        End If
    End Sub 'objPrintDocument_PrintPage

End Class 'clsPrintList
```

Η φόρμα με την μπάρα προόδου δεν περιέχει τίποτα άλλο πέρα από την μπάρα προόδου και ένα Label στο οποίο υπάρχει η δυνατότητα να τοποθετηθεί κάποιο κείμενο το οποίο θα περιγράφει τον λόγο αναμονής. Εξαιτίας του ότι θέλουμε η εφαρμογή να συνεχίσει να λειτουργεί πίσω από την φόρμα που εμφανίζει την μπάρα προόδου δημιουργήσαμε την κλάση clsProgressBar η οποία αναλαμβάνει να εμφανίσει την φόρμα σε ένα διαφορετικό Thread από αυτό στο οποίο εκτελείτε η εφαρμογή. Άλλωστε ο λόγος ύπαρξης μιας μπάρας προόδου είναι να δείξει στον χρήστη ότι η εφαρμογή επεξεργάζεται και εκτελείται κάποια εντολή του η οποία απαιτεί περισσότερο χρόνο από τον συνηθισμένο και να τον κρατήσει απασχολημένο για αυτόν τον χρόνο.

3.7.1 – Οι Constructor της κλάσης clsProgressBar

```
Public Sub New(ByVal strText As String)
```

Ο Constructor της κλάσης έχει είσοδο ένα αλφαριθμητικό το οποίο αντιπροσωπεύει τον κείμενο που θα φαίνεται πάνω από την μπάρα προόδου.

3.7.2 – Οι Μέθοδοι της κλάσης clsProgressBar

```
Public Sub StartProgressBar()
```

Η μέθοδος StartProgressBar εμφανίζει την φόρμα με την μπάρα προόδου.

```
Public Sub StopProgressBar()
```

Η μέθοδος StopProgressBar κλείνει την φόρμα με την μπάρα προόδου.

3.7.3 – Ο κώδικας της κλάσης clsProgressBar

```
Option Explicit On
Option Strict On

Public Class clsProgressBar

    Private objThread As Threading.Thread
    Private strText As String

    Public Sub New(ByVal strText As String)
        Me.strText = strText
    End Sub 'New

    Protected Overrides Sub Finalize()
        Me.StopProgressBar()
        MyBase.Finalize()
    End Sub 'Finalize

    Public Sub StartProgressBar()
        objThread = New Threading.Thread(AddressOf Me.Start)
        objThread.Start()
    End Sub 'StartProgressBar

    Public Sub StopProgressBar()
        objThread.Abort()
    End Sub 'StopProgressBar

    Private Sub Start()
        Dim ProgressBarForm As New frmProgressBar()
        ProgressBarForm.lblProgressBar.Text = Me.strText
```

```
        ProgressBarForm.Show()
        Do
            My.Application.DoEvents()
        Loop Until System.Threading.Thread.CurrentThread.ThreadState <>
Threading.ThreadState.Running
        ProgressBarForm.Close()
    End Sub 'Start

End Class 'clsProgressBar
```

Η κλάση clsGlobalEvents χρησιμοποιείται για να δημιουργηθεί ένα αντικείμενο μέσω του οποίου θα μπορούμε να ελέγχουμε και να δημιουργούμε γεγονότα τα οποία θα είναι καθολικά για την εφαρμογή. Η χρησιμότητα της κλάσης βασίζετε στην αναγκαιότητα που υπάρχει κάθε φορά που προστίθεται ή διαγράφεται κάποια εγγραφή από την βάση δεδομένων να ανανεώνονται οι κατάλληλες λίστες στο περιβάλλον της εφαρμογής. Η κλάση περιέχει διάφορα γεγονότα τα οποία μπορούμε να τα ελέγχουμε από οποιοδήποτε σημείο της εφαρμογής.

Η κλάση αυτή είναι σχεδιασμένη να λειτουργεί μόνο για αυτήν την εφαρμογή και δεν καλύπτει τον στόχο του επαναχρησιμοποιήσιμου κώδικα κατά την ανάπτυξη εμπορικών εφαρμογών. Η λογική και ο τρόπος σχεδίασής της όμως μπορούν να χρησιμοποιηθούν για την ανάπτυξη παρόμοιων κλάσεων σε άλλες εφαρμογές όταν αυτό είναι απαραίτητο.

3.8.1 – Οι Constructor της κλάσης clsGlobalEvents

```
Public Sub New()
```

Ο Constructor της κλάσης δεν έχει καμία είσοδο και απλά αρχικοποιεί το αντικείμενο της κλάσης.

3.8.2 – Οι Μέθοδοι της κλάσης clsGlobalEvents

```
Public Sub RaiseGlobalEvent(ByVal strEventName As String)
```

Η μέθοδος RaiseGlobalEvent έχει ως είσοδο ένα αλφαριθμητικό το οποίο περιγράφει το γεγονός που θέλουμε να δημιουργήσουμε. Η μέθοδος αναλαμβάνει να σηκώσει το κατάλληλο γεγονός.

3.8.3 – Ο κώδικας της κλάσης clsGlobalEvents

```
Option Explicit On
Option Strict On

Public Class clsGlobalEvents

    '################################################
    '## Refresh Lists Events
    '################################################
    Public Event RefreshBuildings()
    Public Event RefreshApartments()
    Public Event RefreshExpensesCategories()
```

```
    Public Event RefreshExpenses()
    Public Event RefreshKoinoxrista()

    Public Sub New()
        '
    End Sub 'New

    Protected Overrides Sub Finalize()
        MyBase.Finalize()
    End Sub 'Finalize

    Public Sub RaiseGlobalEvent(ByVal strEventName As String)
        If strEventName IsNot Nothing Then
            If strEventName.Trim().Length > 0 Then
                Select Case strEventName
                    Case "RefreshBuildingsList"
                        RaiseEvent RefreshBuildings()
                    Case "RefreshApartmentsList"
                        RaiseEvent RefreshApartments()
                    Case "RefreshExpensesCategoriesList"
                        RaiseEvent RefreshExpensesCategories()
                    Case "RefreshExpensesList"
                        RaiseEvent RefreshExpenses()
                    Case "RefreshKoinoxristaList"
                        RaiseEvent RefreshKoinoxrista()

                    Case Else
                        'mdlMain.LogFile.WriteGlobalException("Event " & strEventName & "
not found", "Event Not Found", True)
                End Select
            End If
        End If
    End Sub 'RaiseGlobalEvent

End Class 'clsGlobalEvents
```

4 – Η Ανάπτυξη της Εφαρμογής Β` – Τα αντικείμενα της Ολοκληρωμένης Εφαρμογής Διαχείρισης Πολυκατοικιών & Έκδοσης Κοινοχρήστων

Κατά την ανάπτυξη της εφαρμογής αποφασίστηκε να δημιουργηθούν κάποιες κλάσεις οι οποίες θα διαχειρίζονται τις εγγραφές στην βάση δεδομένων. Κάθε μία από αυτές τις κλάσεις αναλαμβάνει να διαχειριστεί και μία διαφορετική οντότητα/αντικείμενο της εφαρμογής. Παραδείγματος χάρη η κλάση clsBuilding διαχειρίζεται τα κτίρια – πολυκατοικίες, η κλάση clsApartment τα διαμερίσματα του κάθε κτιρίου, η clsExpenseCategory τις κατηγορίες δαπανών του κάθε κτιρίου και η clsExpense τις δαπάνες του.

Επίσης για τις περισσότερες οντότητες δημιουργήθηκε και μία φόρμα κατάλληλη για την εισαγωγή και την επεξεργασία των ιδιοτήτων της, όταν αυτό είναι απαραίτητο και επιτρεπτό να γίνετε από τον χρήστη. Παρόμοια δημιουργήθηκαν οι κατάλληλες φόρμες για κάθε οντότητα στις οποίες ο χρήστης μπορεί να βλέπει ένα πλήθος από εγγραφές τις οποίες μπορεί να μεταβάλει.

Πέρα από τις τέσσερεις κλάσεις που προαναφέρθηκαν αναπτύχθηκαν συνολικά ένδεκα κλάσεις/οντότητες για την εφαρμογή. Εδώ παρουσιάζονται με αλφαβητική σειρά και με μια μικρή περιγραφή της οντότητας που αφορούν:

i. clsApartment – Τα διαμερίσματα για κάθε κτίριο

ii. clsApartmentCounter – Οι μετρητές για κάθε διαμέρισμα των κτιρίων και για κάθε κατηγορία δαπανών αυτού

iii. clsApartmentDynamicRate – Τα δυναμικά χιλιοστά του κάθε διαμερίσματος των κτιρίων

iv. clsApartmentRate – Τα χιλιοστά του κάθε διαμερίσματος των κτιρίων

v. clsBuilding – Τα κτίρια

vi. clsExpense – Οι δαπάνες για την έκδοση των κοινοχρήστων για κάθε κτίριο

vii. clsExpenseCategory – Οι κατηγορίες των δαπανών για κάθε κτίριο

viii. clsExpensesCategoryCountDate – Οι ημερομηνίες των μετρήσεων για κάθε κατηγορία δαπανών των κτιρίων

ix. clsExpenseCategoryTotals – Το συνολικό ποσό των δαπανών για κάθε κατηγορία δαπανών των κτιρίων

x. clsIssueKoinoxrista – Τα λεπτομερή στοιχεία των κοινοχρήστων, εκδομένων ή μη, για κάθε κτίριο

xi. clsKoinoxrista – Τα βασικά στοιχεία των κοινοχρήστων, εκδομένων ή μη, για κάθε κτίριο

Κάθε μία από αυτές τις κλάσεις έχει δημιουργηθεί σύμφωνα με έναν βασικό σχεδιασμό ως συλλογή των απαραίτητων ιδιοτήτων και μεθόδων για την προσθήκη, την ανάκτηση, την ενημέρωση και την διαγραφή των εγγραφών που αφορούν την συγκεκριμένη οντότητα.

Οι ιδιότητες της κάθε κλάσης αφορούν κυρίως τα πεδία της βάσης δεδομένων της κάθε οντότητας καθώς επίσης και κάθε άλλη πιθανή ιδιότητα που θα χρειαστεί για τον έλεγχο της εφαρμογής σε σχέση με την οντότητα αυτήν. Όπως η ιδιότητα που ελέγχει εάν η οντότητα που ζητήθηκε από την βάση δεδομένων υπάρχει ως εγγραφή σε αυτήν ή όχι.

Οι μέθοδοι που προσφέρονται από κάθε κλάση είναι συνήθως τέσσερεις, οι τρεις από τις οποίες είναι κοινές και μπορούν να χρησιμοποιηθούν χωρίς να δημιουργηθεί κάποιο αντικείμενο για αυτήν την κλάση. Οι τρεις κοινές μέθοδοι αφορούν πρώτων την ανάκτηση του πλήθους των εγγραφών από την βάση δεδομένων συμφωνά με τα κριτήρια που έχει ορίσει ο χρήστης, δεύτερων την ανάκτηση όλων των εγγραφών από την βάση δεδομένων συμφωνά με τα κριτήρια που έχει ορίσει ο χρήστης και τέλος την διαγραφή των εγγραφών αυτών από την βάση δεδομένων που συμφωνούν με τα κριτήρια που έχει ορίσει ο χρήστης. Η τέταρτη συνηθισμένη μέθοδος που προσφέρεται από κάθε κλάση και οι οποία δεν είναι κοινή, αφορά την ενημέρωση των πεδίων της βάσης δεδομένων σύμφωνα με τις ιδιότητες της κλάσης, είτε πρόκειται για νέα εγγραφή είτε για την ενημέρωση κάποιας παλιάς εγγραφής.

Κάθε κλάση περιέχει και μία εσωτερική κρυφή μέθοδο η οποία ελέγχει για την ορθότητα και την εγκυρότητα των τιμών που έχουν οι ιδιότητες της κλάσης προτού αυτή προβεί σε ενημέρωση της βάσης δεδομένων και ενημερώνει τον χρήστη για τα πιθανά λάθη που έχει κάνει. Επίσης υπάρχει και μία εσωτερική κρυφή ιδιότητα η οποία ελέγχει εάν η συγκεκριμένη οντότητα είναι νέα και πρέπει να προστεθεί στην βάση δεδομένων ή έχει ανακτηθεί από αυτήν οπότε θα πρέπει απλά να ενημερωθεί η βάση δεδομένων για τις αλλαγές που έγιναν στις ιδιότητές της.

Οι κλάσεις αυτές έχουν τρεις Constructor με τους οποίους μπορεί να αρχικοποιηθεί ένα αντικείμενο της συγκεκριμένης κλάσης. Ο πρώτος, ο οποίος παίρνει ως είσοδο ένα αντικείμενο του IWin32Window Interface, χρησιμοποιείται όταν θέλουμε να δημιουργήσουμε ένα νέο αντικείμενο για να το προσθέσουμε αργότερα στην βάση δεδομένων. Ο δεύτερος constructor έχει δύο εισόδους, ένα αλφαριθμητικό με το οποίο εισάγουμε την τιμή από το πρωτεύων κλειδί της εγγραφής που μας ενδιαφέρει και ένα

αντικείμενο του IWin32Window Interface, και χρησιμοποιείται όταν θέλουμε να ανακτήσουμε από την βάση δεδομένων μία εγγραφή βάση του πρωτεύοντος κλειδιού. Ο τρίτος και τελευταίος constructor έχει επίσης δύο εισόδους με την διαφορά ότι αντί για αλφαριθμητικό το οποίο περιέχει το πρωτεύων κλειδί εισάγουμε ένα αντικείμενο DataRow από το System.Data Namespace του .Net Framework το οποίο περιέχει την εγγραφή της βάσης δεδομένων που μας ενδιαφέρει.

Συμπεράσματα

Μετά την ολοκλήρωση αυτής της πτυχιακής εργασίας και ανατρέχοντας στην πορεία της δημιουργίας της μπορούμε να ξεχωρίσουμε κάποια σημαντικά σημεία σε αυτήν την πορεία.

Καταρχάς καθοριστικό παράγοντα στην κατανόηση της έκδοσης των κοινοχρήστων και του καταμερισμού των δαπανών ήταν η επικοινωνία μας με εταιρίες έκδοσης κοινοχρήστων. Η εταιρία Διαχειριστική στο Ηράκλειο και Ερμής στην Θεσσαλονίκη ήταν αυτές οι οποίες μας προσέφεραν πληθώρα πληροφοριών για τον τρόπο με τον οποίο καταμερίζονται οι δαπάνες μιας πολυκατοικίας κατά την έκδοση των κοινοχρήστων.

Μέσα από αυτήν την επικοινωνία με εταιρίες έκδοσης κοινοχρήστων έγινε φανερή και πλήρως κατανοητή η απαίτηση για δημιουργία δυναμικών κατηγοριών δαπανών. Οι δυναμικές κατηγορίες δαπανών είναι ίσως η σημαντικότερη απαίτηση όλης της εφαρμογής. Εξαιτίας της πολυπλοκότητας και της ποικιλίας των τρόπων με τους οποίους καταμερίζονται οι δαπάνες σε κάθε πολυκατοικία ανάλογα με το καταστατικό της και η μεγάλη ανομοιότητα που παρουσιάζεται σε αυτά τα καταστατικά από πολυκατοικία σε πολυκατοικία, χρειάστηκε να δώσουμε στον χρήστη την δυνατότητα να μπορεί να δημιουργήσει κατηγορίες δαπανών διαφορετικές σε κάθε πολυκατοικία οι οποίες να μπορούν να καλύψουν όσων τω δυνατών περισσότερες από τις περιπτώσεις.

Τέλος για να μπορέσει η εφαρμογή μας να κυκλοφορήσει στην αγορά ως ένα ανταγωνιστικό προϊόν θα πρέπει να αναπτυχθεί περισσότερο και κυρίως να ενσωματώσει τα παρακάτω:

- υποστήριξη πολλαπλών χρηστών με διαφορετικά δικαιώματα χρήσης και πρόσβασης στην εφαρμογή που θα καθορίζονται από έναν administrator
- δυνατότητα ταυτόχρονης λειτουργίας σε περισσότερους από έναν υπολογιστές με μία κοινόχρηστη βάση δεδομένων
- διασφάλιση του αυθεντικού αντιτύπου της εφαρμογής μέσω ενεργοποίησης προϊόντος και αδειών χρήσης
- δημιουργία λογιστικού τμήματος της εφαρμογής με έσοδα – έξοδα ή εξόδου των δεδομένων για την ενημέρωση κάποιας άλλης λογιστικής εφαρμογής
- δημιουργία στατιστικού τμήματος της εφαρμογής
- δημιουργία εγχειριδίου χρήσης της εφαρμογής

Βιβλιογραφία

- MSDN Library for Visual Studio 2005
- http://msdn.microsoft.com
- http://www.dotnetzone.gr
- RAM Κοινόχρηστα 1.1 από http://www.in.gr/ram/ram155.htm
- http://www.koino.gr
- http://koinoxrista.atspace.com
- http://www.keme-net.com/faq.htm
- Επικοινωνία με την Εταιρία Διαχειριστική
- Επικοινωνία με την Εταιρία Ερμής (http://www.ermis.com.gr)