Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Εφαρμογής Business Management για τη Χρηματοοικονομική Ανάλυση και Αξιολόγηση των Αντιπροσωπειών Αυτοκινήτου** **Development of Business Management Application for the Financial Analysis and Evaluation of Dealerships in the Car Industry**
Ονοματεπώνυμο Φοιτητή	**Παπάζογλου Ευστράτιος**
Πατρώνυμο	**Χρήστος**
Αριθμός Μητρώου	**ΜΠΠΛ 13064**
Επιβλέπων	**Βίρβου Μαρία, Καθηγήτρια & Πρόεδρος Τμήματος Πληροφορικής**

Ημερομηνία Παράδοσης **Οκτώβριος 2015**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη

Η χρηματοοικονομική ανάλυση και αξιολόγηση των αντιπροσωπειών αυτοκινήτου αποτελεί μια σύνθετη διαδικασία και κυρίως η εκμετάλλευση αυτής της διαδικασίας για τη διεξαγωγή πολύτιμων συμπερασμάτων και για την εκπόνηση των απαραίτητων σχεδίων και ενεργειών. Βασική πρόκληση της όλης διαδικασίας είναι, μετά το πέρας της ανάλυσης, η όσο το δυνατόν πιο συμπυκνωμένη, αλλά περιεκτική την ίδια στιγμή, μετάδοση της προκύπτουσας εικόνας στους κατεξοχήν ενδιαφερόμενους: τους επιχειρηματίες / επενδυτές / ιδιοκτήτες των αντιπροσωπειών αυτοκινήτου αλλά και κυρίως το ανώτατο management αυτών καθώς και το διεθνές ανώτατο management της ξένης άμεσης επένδυσης (foreign direct investment) που εισάγει τα αυτοκίνητα στη χώρα. Κύριο εργαλείο / μεθοδολογία στην παραπάνω προσπάθεια αποτελεί το Business Management όπως αυτό εκφράζεται από τις διαφορετικές και αλληλεπιδρόμενες συνιστώσες του, με βασική την ανάλυση Κερδών εκμετάλλευσης (ROS analysis) και τη σύγκριση του με τη δραστηριότητα του προηγούμενου έτους. Η παρούσα εφαρμογή αποσκοπεί στη δημιουργία ενός ομοιογενούς περιβάλλοντος, φιλικού προς το χρήστη για τη διευκόλυνση της ανάλυσης / αξιολόγησης αυτής προκειμένου οι χρήστες να εισάγουν με ομοιογένεια τα απαιτούμενα δεδομένα, αλλά αφετέρου το top management να έχει μια όσο το δυνατόν πιο σύντομη και ολοκληρωμένη εικόνα της πραγματικότητας.

Abstract

Financial analysis and evaluation of car dealerships is a complicated and very important process for the necessary decision making and required action plans. Basic objective and challenge at the same time is to provide top management with concentrated, short and also meaningful information so as to benchmark and to increase efficiency. ROS Analysis is the most important element of Business Management and constitutes a very useful tool mainly due to the fact that distinguishes the company's profitability by activity and as a total at the same time, especially taking into consideration comparisons between two consecutive years. This software application is designed in a user friendly way so as the users to take advantage of a neutral and safe environment for data input and the top management to be provided with all that is needed: a short, full of information framework enabling accurate and immediate action to be taken and diffused to lower levels of management.

Εισαγωγή

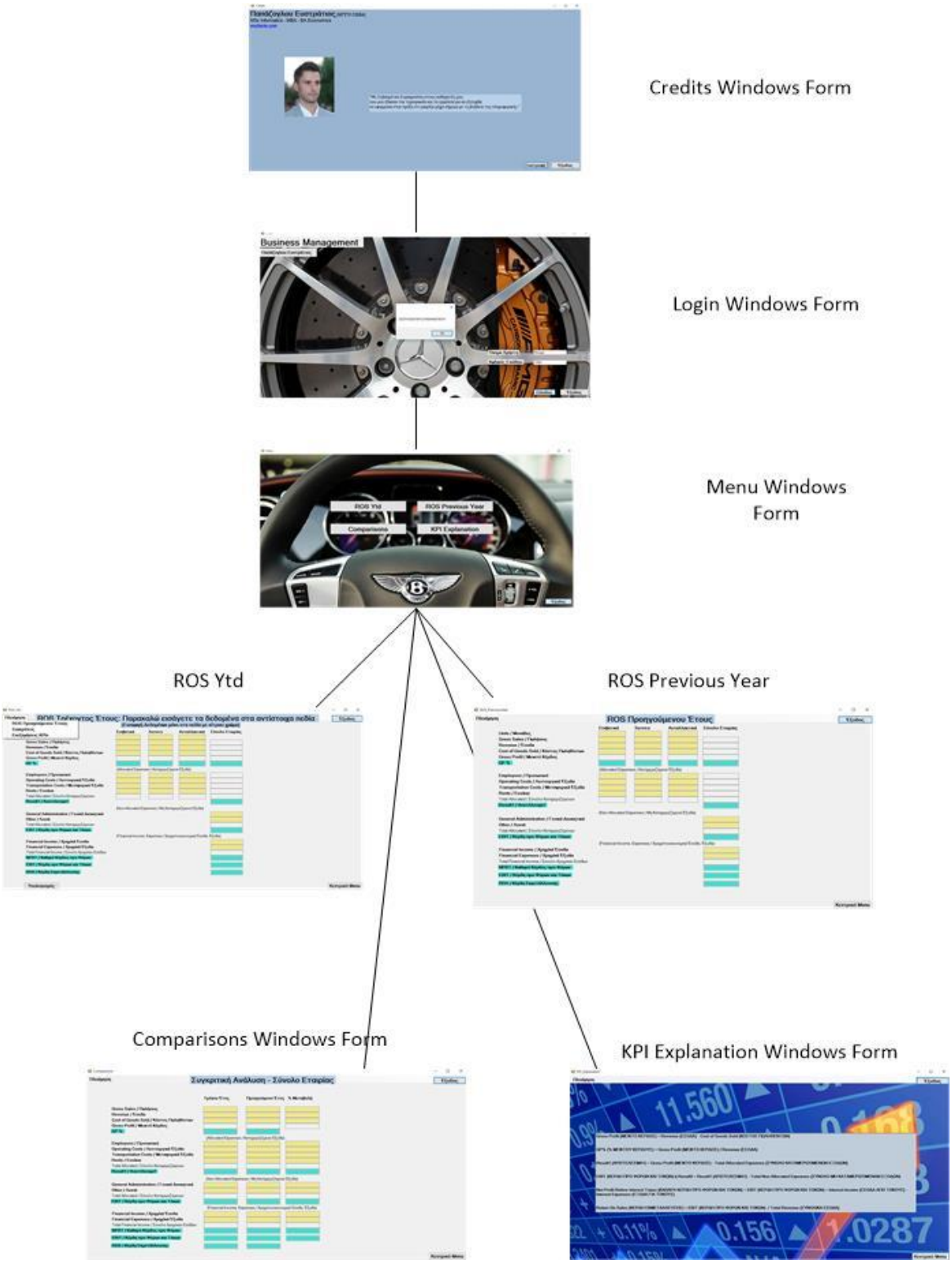
Παραδοσιακά ο χώρος του αυτοκινήτου απαιτούσε, και απαιτεί ακόμα και στις μέρες μας, άμεση και λεπτομερειακή ανάλυση λόγω των μεγάλων χρηματοοικονομικών εισροών και εκροών που λαμβάνουν χώρα. Εισροών με τη μορφή των χρημάτων που καταβάλουν οι πελάτες για αγορά καινούριων αυτοκινήτων και εκροών για την αγορά και εισαγωγή των αυτοκινήτων από πλευράς διανομέων αλλά κυρίως για τον Ελλαδικό χώρο λόγω της ιδιομορφίας των υψηλών δανειακών κεφαλαίων που χρησιμοποιούν οι επιχειρηματίες. Επιπροσθέτως, η πολυπλοκότητα των εργασιών μιας αντιπροσωπείας αυτοκινήτου είναι ιδιάζουσα και διέπεται από τις ενέργειες πώλησης νέων αυτοκινήτων, επισκευής (service) και πώλησης ανταλλακτικών. Αυτές ακριβώς οι λειτουργίες προσπαθούν να διαχωριστούν, να επιμεριστούν σε αυτές τα αντίστοιχα έξοδα τους, να αναλυθούν και να εξαχθούν συμπεράσματα, τα οποία η ανώτατη διοίκηση θα χρησιμοποιήσει για να κατευθύνει τη δραστηριότητα και πιθανώς να μεταβάλλει την ποσόστωση χρησιμοποίησης πόρων ανάλογα με τις ανάγκες, τις επιδιώξεις και κυρίως την παραγωγικότητα κάθε δραστηριότητας. Ειδικά σε περιόδους οικονομικής στενότητας, όπως αυτή που διανύουμε και στην εποχή μας, η ανάγκη αυτή καθίσταται ακόμα μεγαλύτερη προκειμένου να γίνουν οι απαιτούμενες ενέργειες για την αύξηση της δραστηριότητας εκ μέρους των αντιπροσωπειών αυτοκινήτου που θα τους διασφαλίζει την οικονομική βιωσιμότητα και κατ' επέκταση την οικονομική τους ευρωστία. Παραδείγματος χάριν, οι διελεύσεις αυτοκινήτων στα συνεργεία των αντιπροσωπειών, λόγω του όγκου των συναλλαγών τους καθώς επίσης και της συχνότητας που γίνονται μπορούν να εξασφαλίσουν ρευστότητα και βιωσιμότητα ακόμα και σε περιόδους με ελάχιστες πωλήσεις νέων αυτοκινήτων.

Λόγω αυτής της ανάγκης, δημιουργήθηκε το Business Management, με διάφορες συνιστώσες υπολογισμού και ποικίλες εκφάνσεις. Η πιο σημαντική, με άμεσα μετρήσιμα αποτελέσματα που δίνουν τη δυνατότητα λεπτών χειρισμών αλλά και αναζήτησης των αιτίων πίσω από άσχημα οικονομικά αποτελέσματα και από περιορισμένη αποδοτικότητα είναι αυτή των Κερδών Εκμετάλλευσης. Η επιτυχία της ανάλυσης αυτής έγκειται στο γεγονός ότι χωρίζει ολόκληρη την οντότητα μιας αντιπροσωπείας αυτοκινήτου στις λειτουργικές συνιστώσες που προηγήθηκαν παραπάνω. Σε κάθε μια από αυτές καταμερίζονται οι χρηματοοικονομικές εισροές σε επίπεδο καταμεριζόμενων και μη εξόδων και οδηγείται η ανάλυση με τον τρόπο αυτό σε αποτέλεσμα κατώτατου επιπέδου (bottom-line effect). Επιτυγχάνεται έτσι η συγκριτική ανάλυση και γίνεται άμεσο και εμφανές το επίπεδο στο οποίο θα εμφανιστούν και θα ληφθούν τα νέα μέτρα (priority actions). Όλα αυτά σε μια μικρή και συμπυκνωμένη εικόνα, χωρίς περιττή ή μη συναρτώμενη πληροφόρηση που θα μπορούσε να αποπροσανατολίσει ή να στρέψει την προσοχή του ανωτέρου management της εταιρίας μακριά από το ζητούμενο. Για το λόγο αυτό, όπως θα δούμε στην εφαρμογή μας παρακάτω και όπως διακαώς μου είχε ζητηθεί κατά

την επαγγελματική μου σταδιοδρομία στο αντικείμενο αυτό, οι συγκρίσεις και ο πίνακας αποτελεσμάτων περιορίζονται σε μόνο λίγες γραμμές. Στο επίπεδο εκείνο των αποτελεσμάτων που φωτίζει το σωστό δρόμο που πρέπει να ακολουθηθεί και που δείχνει τα αίτια και τις λύσεις. Ένας πίνακας μόνο αρκεί και τις περισσότερες φορές απαιτείται από το management που να δίνει όλη την πληροφόρηση. Η μέθοδος αυτή, λόγω της μεγάλης σημασίας της αλλά και λόγω του εύρους εφαρμογών της προσεγγίζεται προγραμματιστικά για τη δημιουργία ενός ενιαίου πλαισίου που περιγράφεται με σαφήνεια παρακάτω.

Η ανάγκη για τη δημιουργία αυτού του προγράμματος / εφαρμογής προέκυψε μέσα από τις δυσκολίες της καθημερινής πρακτικής. Η χρήση υπολογιστικών φύλλων και άλλων μεθόδων επέτρεπε αποκλίσεις τόσο σχετικά με το είδος των στοιχείων που εισάγονταν, όσο και με τη χρονική συγκυρία που γινόταν αυτό. Βασικό πρόβλημα αποτελούσε ότι οι χρήστες των παραπάνω λειτουργιών δεν αντιλαμβάνονταν τη συνεισφορά τους στη γενικότερη εικόνα της εταιρίας. Ήταν και είναι απαραίτητο οι χρήστες εκ μέρους των αντιπροσωπειών να ακολουθούν ένα ομοιογενές σύστημα, που θα τους καθοδηγεί εντάσσοντας την εισαγωγή δεδομένων σε ένα γενικότερο σύστημα, το οποίο τους επιτρέπει να αντιληφθούν τη σημαντικότητα των ενεργειών τους. Ένα σύστημα με ομοιογένεια που δεν θα μπορεί ο καθένας να εισάγει ότι δεδομένα θέλει και όπως τα αντιλαμβάνεται αυτός, αλλά θα διέπεται από κοινούς κανόνες η εφαρμογή των οποίων θα δίνει άμεσα την απαραίτητη πληροφόρηση στους ίδιους τους επιχειρηματίες και στην ανώτατη διοίκηση αυτών. Με τον τρόπο αυτό επιτυγχάνεται διάχυση της πληροφορία σε τοπικό επίπεδο και όχι μόνο σε κεντρικό και εξοικονομείται χρόνος που θα χρειαζόταν για να συλλεχθούν κεντρικά τα αποτελέσματα και μετά να ενημερωθεί η κάθε αντιπροσωπεία αυτοκινήτων ξεχωριστά. Το σημαντικότερο όμως είναι η εφαρμογή κοινών κανόνων υπολογισμού, δηλαδή δεν υπολογίζει ο καθένας τους αριθμοδείκτες του με τον δικό του τρόπο αντίληψης των πραγμάτων ή όπως τον βολεύει να τα παρουσιάσει. Κοινοί κανόνες διέπουν όλους άρα και κοινά συμπεράσματα, μετρήσιμα και συγκρίσιμα, μπορούν να εξαχθούν.

Σχεδιασμός front-end εφαρμογής



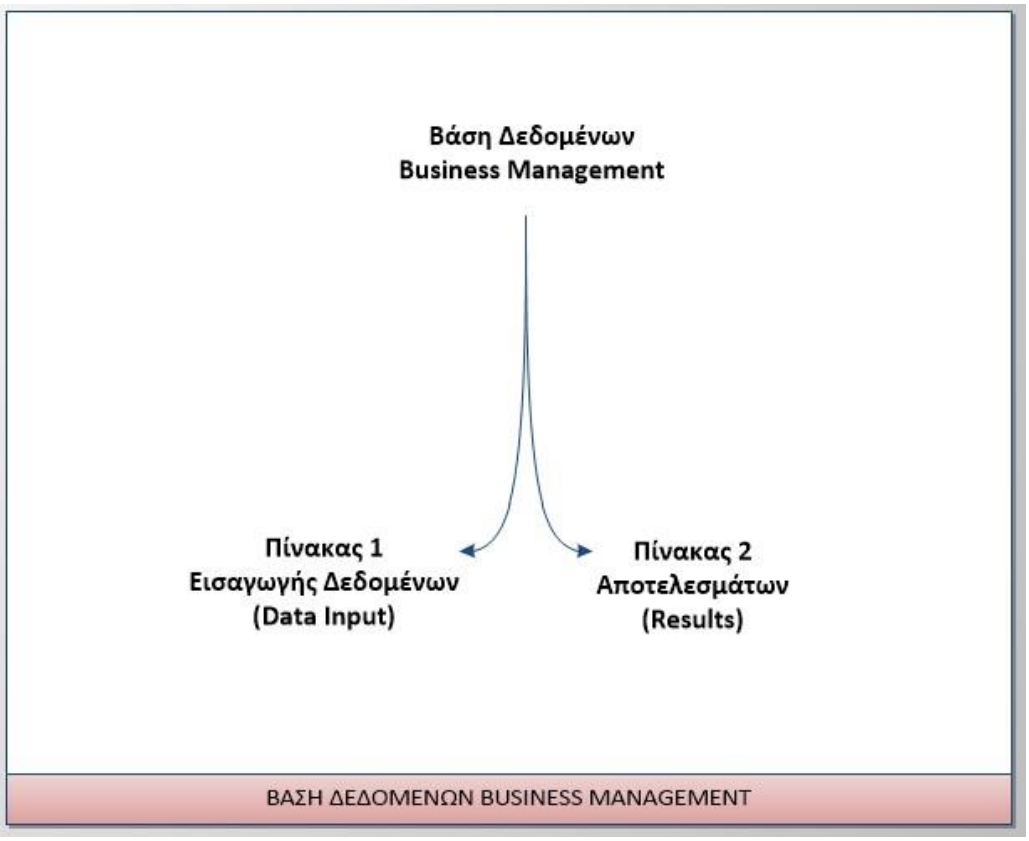
Το front-end τμήμα της εφαρμογής έχει σχεδιαστεί με τέτοιο τρόπο ώστε να υπάρχει ομοιομορφία στο περιβάλλον χρήσης, ο χρήστης να αναγνωρίζει όλες τις επιλογές αμέσως, αλλά και η μορφοποίηση των κουμπιών πλοήγησης να είναι ίδια για όλα τα κουμπιά και σε αρκετές περιπτώσεις ο χρήστης να ξέρει ακριβώς που θα βρει κάθε κουμπί χωρίς καν να έχει δει τη σελίδα /φόρμα που ακολουθεί. Χαρακτηριστικό παράδειγμα είναι το πτυσσόμενο menu αλλά και τα κουμπιά εξόδου στις φόρμες της εφαρμογής μας. Το σχεδιάγραμμα που προηγήθηκε είναι ενδεικτικό της διάταξης των windows forms, που παρατίθενται στην ενότητα αυτή, καθώς η πλοήγηση περιγράφεται εκτενώς στον οδηγό σύντομης εκκίνησης που ακολουθεί στα παρακάτω κεφάλαια. Η λογική που διέπει τόσο το σχεδιασμό όσο και τη λειτουργία της είναι ότι ο χρήστης μπορεί ανά πάσα στιγμή να πλοηγηθεί παντού με ομοιομορφία και ομοιογένεια στην παρουσίαση των λειτουργικών επιλογών που μπορεί να κάνει κατά τη χρήση της εφαρμογής προκειμένου να φτάσει στο επιθυμητο επίπεδο και αποτέλεσμα.

Σχεδιασμός Βάσης Δεδομένων

Πριν προχωρήσουμε στην ανάλυση μας, είναι σημαντικό να δούμε πως έχει σχεδιαστεί η βάση δεδομένων, το δεύτερο σημαντικό τμήμα της εφαρμογής με το οποίο αλληλεπιδρά το front-end τμήμα της καθώς επίσης και τους κοινούς κανόνες που τη διέπουν.

Η βάση δεδομένων, με την οποία επικοινωνεί το front-end τμήμα της εφαρμογής, έχει δημιουργηθεί με βάση τις ανάγκες που ανέκυψαν για την ορθή λειτουργία αυτής, έτσι ώστε να διευκολύνονται οι διάφορες ανακύπτουσες ενέργειες, αλλά και για να είναι εύκολη η προσαρμογή σε μελλοντικές ανάγκες αναβάθμισης. Ο τρόπος με τον οποίο επικοινωνεί η βάση δεδομένων με το front-end τμήμα περιγράφεται αναλυτικά σε ειδικό κεφάλαιο παρακάτω.

Σε συνέπεια της παραπάνω λογικής, βασικές ανάγκες χρησιμοποίησης της βάσης δεδομένων αποτέλεσαν η αποθήκευση των δεδομένων που εισάγει ο χρήστης και η εμφάνιση των αποτελεσμάτων. Πριν βέβαια, το front-end τμήμα της εφαρμογής διαβάζει από τη βάση τα δεδομένα που έχουν εισαχθεί, κάνει τους απαραίτητους υπολογισμούς και αποθηκεύει σε αυτή τα νέα αποτελέσματα για να τα ξαναδιαβάσει προκειμένου να παρουσιαστούν στα windows forms που απαιτείται.



Σχηματικά, βλέπουμε τους δύο πίνακες, από τους οποίους αποτελείται η Βάση Δεδομένων, τον πίνακα inputFields (εισαγωγής δεδομένων / data input) και τον πίνακα outputFields (αποτελεσμάτων / results). Σε κάθε πίνακα, τα έτη (οι οικονομικές χρήσεις / χρονιές / περίοδοι της εταιρίας) αποτελούν τις εγγραφές αυτών. Έτσι παραδείγματος χάρη, τα έτη 2015, 2014 κλπ, αποτελούν διαδοχικές εγγραφές των πινάκων. Το πεδίο anno αποτελεί κοινό πεδίο για τους 2 πίνακες, παίζει το ρόλο του πρωτεύοντος κλειδιού και δένει τους 2 πίνακες μεταξύ τους. Για τον πίνακα inputFields το screenshot που ακολουθεί είναι χαρακτηριστικό:

inputFields

anno	pcUnits	pcGrossSales	pcRevenue	pcCostOfUnitsSold	pcWages	pcOperatingCosts	pcTransportationCosts	pcRents	sUnits
2005	1	1	1	1	1	1	1	1	1
2006	2	2	2	2	2	2	2	2	2
2007	3	3	3	3	3	3	3	3	3
2008	4	4	4	4	4	4	4	4	4
2009	5	5	5	5	5	5	5	5	5
2010	6	6	6	6	6	6	6	6	6
2011	7	7	7	7	7	7	7	7	7
2012	8	8	8	8	8	8	8	8	8
2013	9	9	9	9	9	9	9	9	9
2014	10	10	10	10	10	10	10	10	10

Παραδείγματος χάρη, για το έτος 2015 (για την εγγραφή 2015), υπάρχουν στη βάση μια σειρά από πεδία: pcUnits, pcGrossSales και όλα τα υπόλοιπα που καλείται να συμπληρώσει ο χρήστης στο windows form του ROS Ytd front-end τμήματος της εφαρμογής. Συνολικά ο πίνακας αυτός έχει 29 πεδία (1 το anno και 28 υπόλοιπα που αποθηκεύουν όλα τα δεδομένα που καλείται να εισάγει ο χρήστης).

Αντίστοιχα, στον πίνακα outputFields της βάσης δεδομένων:

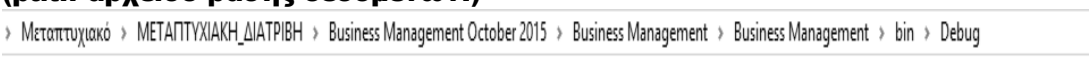
nputFields outputFields

anno	pcInc	pcGp	pcExp	sInc	sGp	sExp	pInc	pGp	pExp	totalGrossS	total
2005	0	0	4	0	0	4	0	0	4	3	
2006	0	0	8	0	0	8	0	0	8	6	
2007	0	0	12	0	0	12	0	0	12	9	
2008	0	0	16	0	0	16	0	0	16	12	
2009	0	0	20	0	0	20	0	0	20	15	
2010	0	0	24	0	0	24	0	0	24	18	
2011	0	0	28	0	0	28	0	0	28	21	
2012	0	0	32	0	0	32	0	0	32	24	
2013	0	0	36	0	0	36	0	0	36	27	
2014	0	0	40	0	0	40	0	0	40	30	

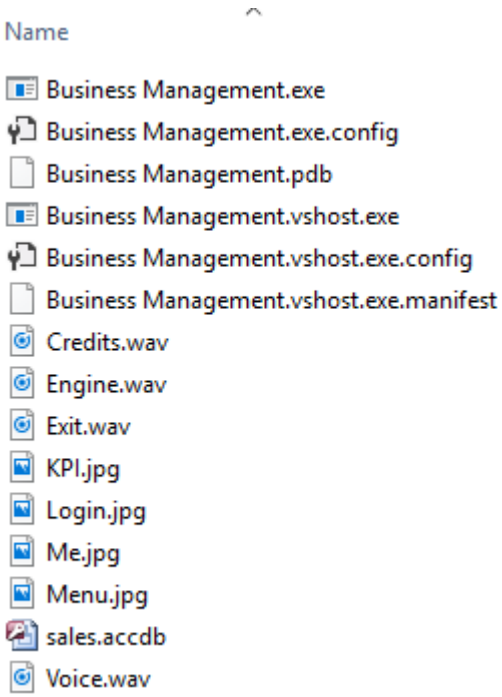
για κάθε έτος (εγγραφή) υπάρχουν τα αντίστοιχα πεδία των αποτελεσμάτων. Συνολικά ο πίνακας αυτός έχει 26 πεδία (1 το anno και 25 υπόλοιπα) που αποθηκεύουν όλα τα αποτελέσματα, τα calculated fields.

Στο σημείο αυτό, για να υπάρχει συνέπεια στην ανάλυση μας αλλά και για να γίνει κατανοητή η ανάγνωση των πινάκων της βάσης δεδομένων (το αρχείο της βρίσκεται μέσα στο debug file στο bin του project):

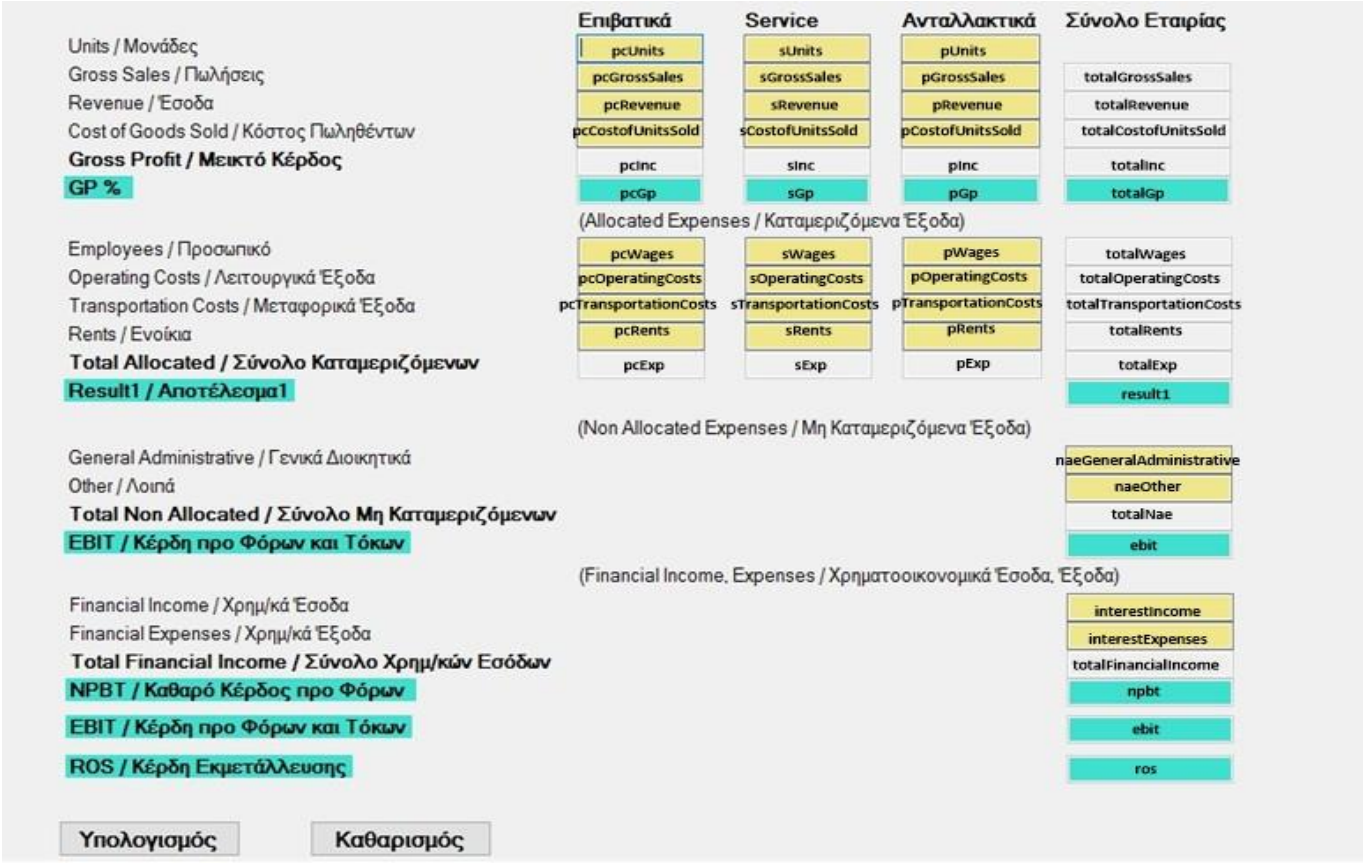
(path αρχείου βάσης δεδομένων:)



(περιεχόμενα debug folder και αρχείο βάσης δεδομένων:)



παραθέτουμε μια αντιστοιχία των κωδικών ονομασιών των πεδίων των πινάκων η οποία ταυτίζεται με τα πεδία που εισάγει ο χρήστης αλλά και αυτά των αποτελεσμάτων. Στη σελίδα που ακολουθεί παρατίθεται αναλυτικός πίνακας (ακριβώς όπως τον βλέπει ο χρήστης της εφαρμογής) πάνω στον οποίο ταυτοποιούνται οι κωδικές ονομασίες των πεδίων για κάθε εγγραφή των πινάκων της βάσης δεδομένων της εφαρμογής μας:

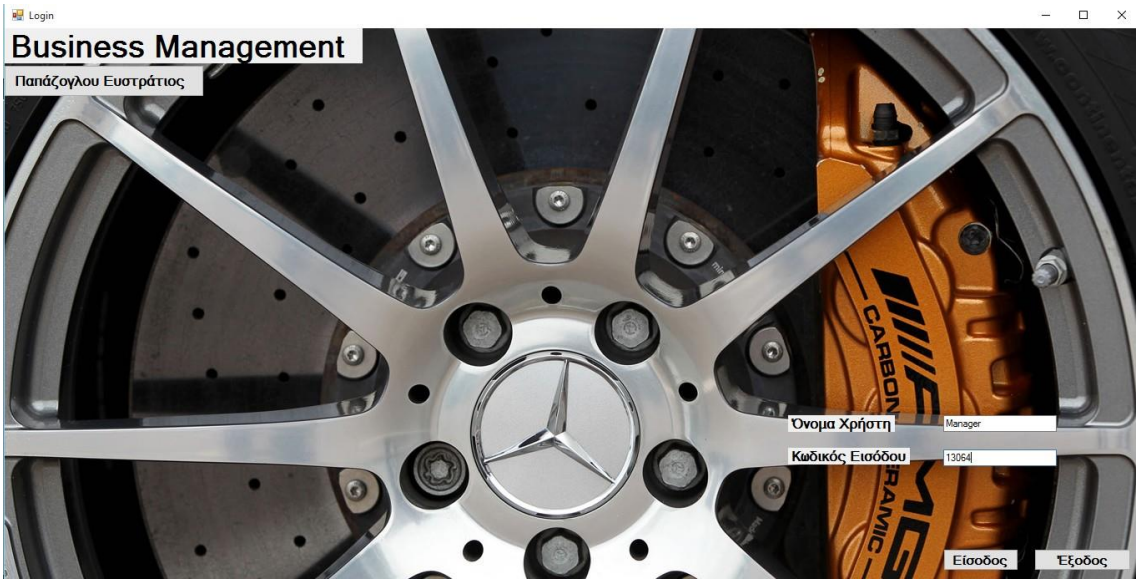


Μετά και από αυτή την ταυτοποίηση και ταυτόχρονα, σχηματική απεικόνιση των κωδικών ονομασιών των πεδίων των εγγραφών των πινάκων της βάσης δεδομένων της εφαρμογής μας και πριν την ανάλυση του τρόπου και του κώδικα με τον οποίο επικοινωνεί η εφαρμογή με τη βάση δεδομένων μας, είναι αναγκαίο και σκόπιμο να δούμε τον τρόπο με τον οποίο ο χρήστης μπορεί να εισέλθει, να πλοηγηθεί και να εξέλθει αυτής παρέχοντας έναν σύντομο οδηγό γρήγορης εκκίνησης.

Οδηγός Γρήγορης Εκκίνησης

Η εφαρμογή του Business Management αποτελείται από διαδοχικά windows forms, συνολικά 7, την πλοήγηση των οποίων θα δούμε σε αυτό το σημείο με τρόπο σύντομο ώστε να διευκολύνεται η γρήγορη εκκίνηση και χρήση της εφαρμογής.

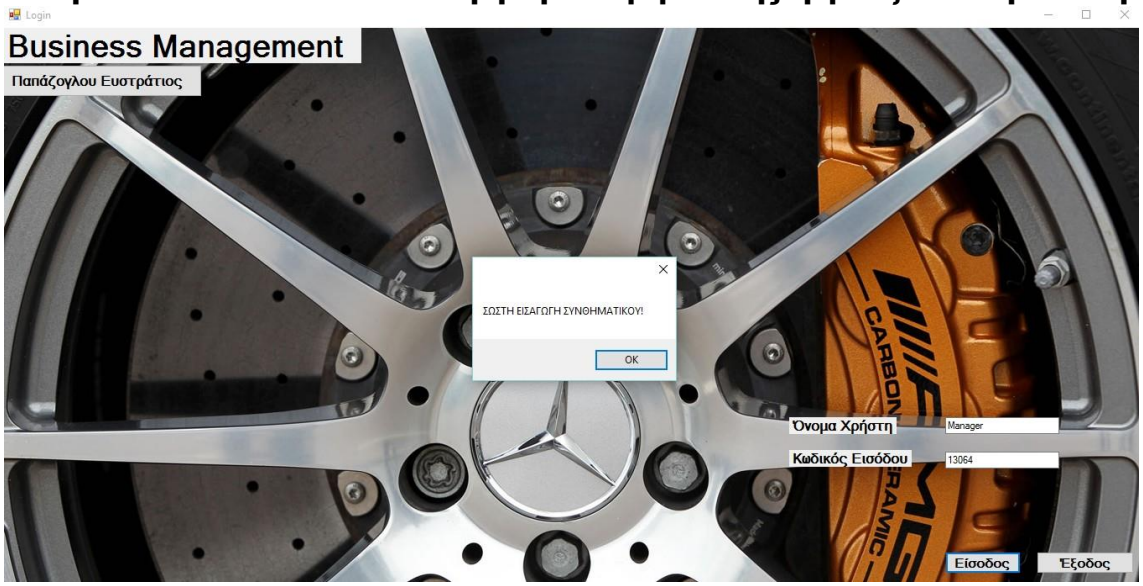
Με την εκκίνηση της εφαρμογής ο χρήστης καλωσορίζεται από ένα ηχητικό μήνυμα, το οποίο τον καλεί να εισάγει τα στοιχεία εισόδου.



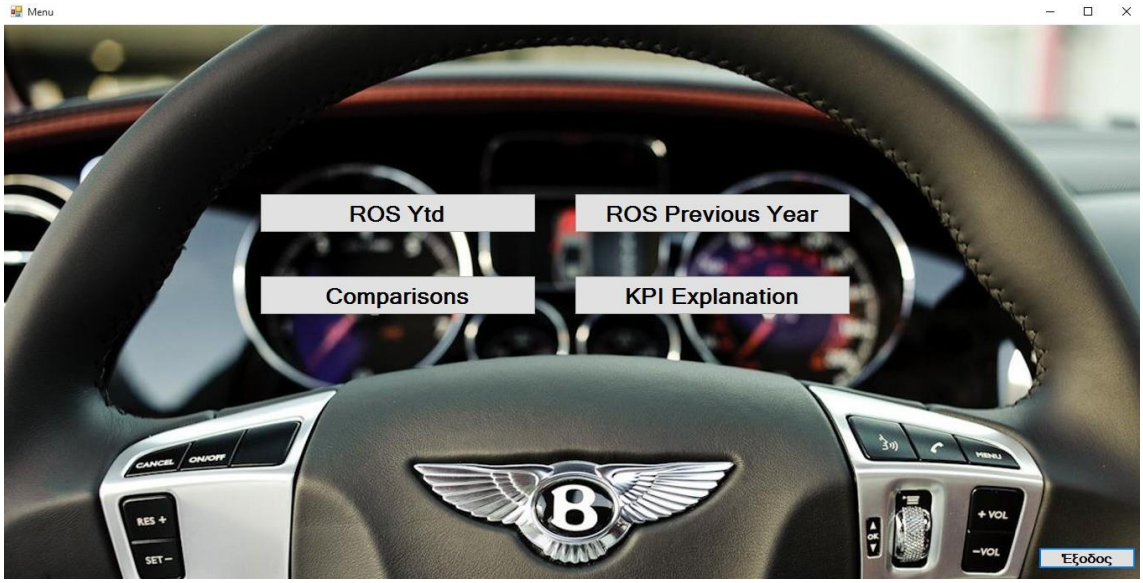
Στο σημείο αυτό και πριν εισάγει τα δεδομένα εισόδου, μπορεί κλικάροντας πάνω στο όνομα ΠΑΠΑΖΟΓΛΟΥ ΕΥΣΤΡΑΤΙΟΣ να μεταβεί στο application form Credits με πληροφορίες για το δημιουργό της εφαρμογής, καθώς επίσης να μεταβεί και στην προσωπική του ιστοσελίδα, πατώντας πάνω στο link που παρατίθεται (you3xcite.com).

Αμέσως με τη μετάβαση σε αυτό το windows form αλλάζει το μουσικό θέμα σηματοδοτώντας την αλλαγή αυτή. Εδώ ο χρήστης μπορεί αν θέλει είτε να τερματίσει την εφαρμογή πατώντας στο κουμπί Έξοδος ή επιστρέφοντας στη φόρμα εισαγωγής δεδομένων.

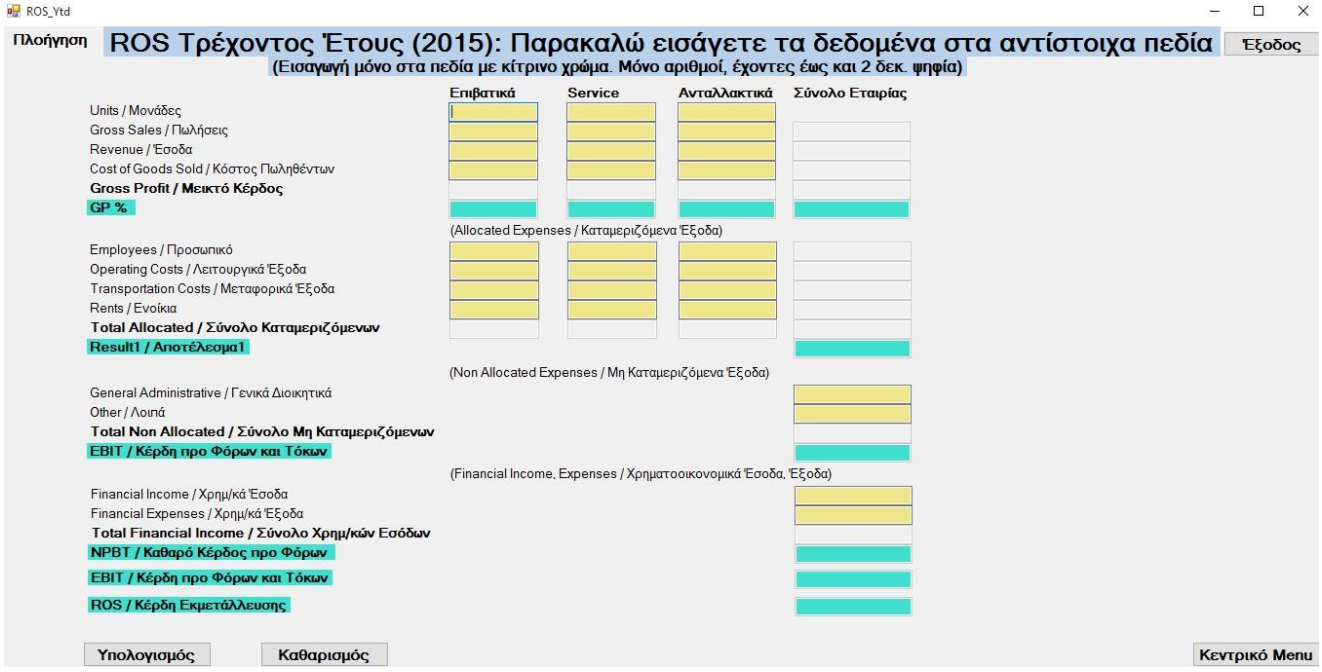
Επιστρέφοντας στη σελίδα εισόδου, τα διακριτικά Εισόδου είναι Όνομα Χρήστη: Manager και Κωδικός Εισόδου: 13064. Με τη σωστή καταχώρηση των δεδομένων που απαιτούνται ένα μήνυμα επιβεβαίωσης εμφανίζεται στην οθόνη



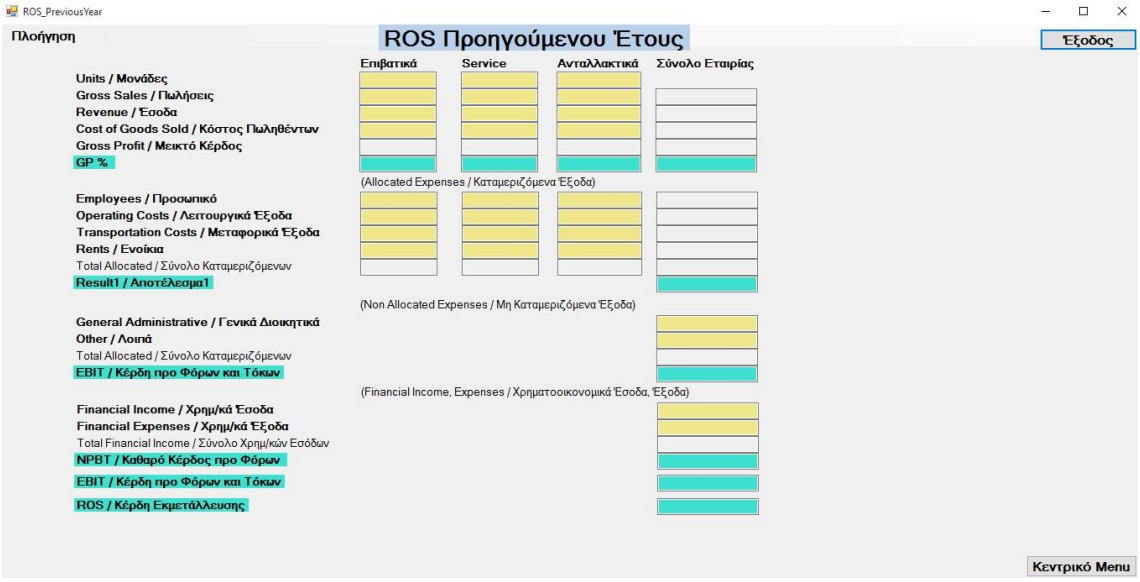
και ο χρήστης περνάει στο κεντρικό menu αφού αλλάξει πάλι το μουσικό θέμα, με ένα ήχο εκκίνησης που σηματοδοτεί την είσοδο στο κεντρικό menu.



Στο σημείο αυτό, ο χρήστης μπορεί να επιλέξει μεταξύ 5 επιλογών: να εξέλθει της εφαρμογής, να μπει στη σελίδα ROS Ytd, να μπει στη σελίδα ROS Previous Year, να μπει στη σελίδα Comparisons ή στην KPI Explanation. Στα screenshots που ακολουθούν αναλύεται με ακρίβεια και σαφήνεια το τι γίνεται και το πως πλοηγείται ο χρήστης σε κάθε σελίδα.



Στη σελίδα ROS Ytd, ο χρήστης μέσω του menu πλοήγησης μπορεί να μεταβεί σε κάθε μια από τις θεματικές ενότητες του κυρίως menu που είδαμε πριν, να εισάγει τα δεδομένα που απαιτούνται στα πεδία που είναι μαρκαρισμένα με κίτρινο χρώμα και μετά να πατήσει το κουμπί υπολογισμού ή να εξέλθει. Ακολούθως και με τον ίδιο τρόπο γίνεται η πλοήγηση στις σελίδες ROS Previous Year και Comparisons, με τη μόνη διαφορά ότι σε αυτές τις σελίδες δεν γίνεται καμία εισαγωγή δεδομένων (για το λόγο αυτό δεν υπάρχει και κουμπί υπολογισμού), τα οποία ανακτώνται κατευθείαν από τη βάση.



Ειδικά στη σελίδα comparisons ή συγκριτική ανάλυση, ο χρήστης βλέπει και τις ποσοστιαίες μεταβολές μεταξύ των 2 ετών. Τέλος με την ίδια λογική πάλι μπορεί να μεταβεί στη σελίδα KPI Explanation

Όπου αναλυτικά παρατίθεται ο τόπος που προκύπτουν οι σημαντικοί αριθμοδείκτες μετά την εισαγωγή των δεδομένων κατά τη φόρμα ROS Ytd. Με συνέπεια στην ανάλυση μας και στη σελίδα αυτή ο χρήστης μπορεί να εξέλθει της εφαρμογής πατώντας στο κουμπί έξοδος στο πάνω αριστερό μέρος της σελίδας. Στο σημείο αυτό, πρέπει να γίνει αναφορά στο γεγονός ότι τόσο τα menu πλοήγησης, όσο και τα κουμπιά είναι φτιαγμένα με τρόπο ώστε να είναι στο ίδιο σημείο, στο ίδιο μέγεθος, στο ίδιο format για τη δημιουργία μιας συνολικής εμπειρίας με ομοιογένεια κατά την πλοήγηση της εφαρμογής.

Κώδικας front end εφαρμογής

Η εφαρμογή του Business Management αποτελείται από διαδοχικά windows forms, συνολικά 7, με βασικές λειτουργίες μετάβασης, πλοήγησης, εισόδου και μουσικής επένδυσης. Τον κώδικα πίσω από αυτές τις λειτουργίες παρουσιάζουμε στην ενότητα αυτή, ενώ το κομμάτι των υπολογισμών και της επικοινωνίας με τη βάση παρατίθενται σε ξεχωριστές ενότητες λόγω της σημαντικότητας τους.

Το πρώτο χαρακτηριστικό που αναλύουμε είναι η μουσική επένδυση 2 ενοτήτων και ο αντίστοιχος κώδικας. Όπως προαναφέρθηκε, κατά την έναρξη της εφαρμογής ο χρήστης έχει τη δυνατότητα να εξέλθει αυτής, να εισάγει τα διακριτικά εισόδου ή πατώντας πάνω στο όνομα ΠΑΠΑΖΟΓΛΟΥ ΕΥΣΤΡΑΤΙΟΣ, να μπει στη σελίδα των credits. Και οι δυο τελευταίες επιλογές διακρίνονται από μουσική επένδυση και κάνουν χρήση της βιβλιοθήκης: using.System.Media.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media; // to use music
```

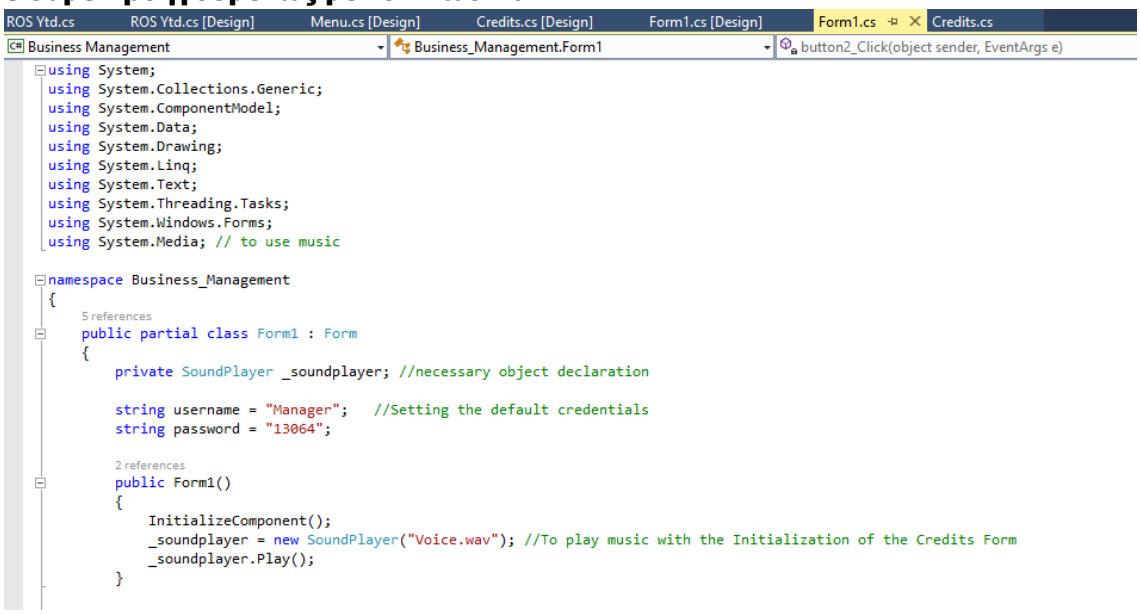
Στην περίπτωση των credits το screenshot που ακολουθεί είναι χαρακτηριστικό:

Business Management | Business_Management.Credits | button2_Click(object sender, EventArgs e)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media; // to use music
using System.Diagnostics; // library for linklabel

namespace Business_Management
{
    5 references
    public partial class Credits : Form
    {
        private SoundPlayer _soundplayer; //necessary object declaration



        1 reference
        public Credits()
        {
            InitializeComponent();

            _soundplayer = new SoundPlayer("Credits.wav"); //To play music with the Initialization of the Credits Form
            _soundplayer.PlayLooping();  // for Looping
        }
```

Γίνεται ένα object declaration, δημιουργείται ένα αντικείμενο _soundplayer, χρησιμοποιεί το wav αρχείο που είναι αποθηκευμένο στο debug bin και ακολούθως γίνεται playlooping, ώστε μόλις ολοκληρωθεί το μουσικό αρχείο να μην τερματιστεί (να μην παίξει μόνο μια φορά αλλά να συνεχίσει να αναπαράγεται) έως ότου ο χρήστης επιλέξει να τερματίσει την εφαρμογή ή να επιστρέψει στο menu εισαγωγής στοιχείων. Σε αυτό το windows form, είτε ο χρήστης μπει για πρώτη φορά είτε μετά από επιστροφή από το credits form, πάλι ένα ηχογραφημένο μήνυμα τον ενημερώνει για την εφαρμογή και για την εισαγωγή διακριτικών στοιχείων.

Πρακτικά αυτό επιτυγχάνεται εκτός από την εισαγωγή της βιβλιοθήκης που είδαμε προηγουμένως με τον κώδικα



Όπως χαρακτηριστικά βλέπουμε στο παράδειγμα μας, εδώ η εντολή δεν είναι playlooping αλλά play, ώστε να μην επαναλαμβάνεται συνεχώς η παρότρυνση για εισαγωγή διακριτικών εισόδου. Στο σημείο αυτό και πάλι σχετικά με την ηχητική επένδυση των windows forms άλλο ένα σημείο πρέπει να τονιστεί.

Αμέσως μόλις ο χρήστης εισάγει τα σωστά συνθηματικά εισόδου και πατήσει πάνω στο κουμπί είσοδος, το πρόγραμμα επαληθεύει το σωστό της εισαγωγής και την ίδια στιγμή που μεταφέρει τον χρήστη στο κεντρικό menu, ακούγεται ο σύντομος 3 δευτερολέπτων ήχος engine.wav, που βρίσκεται στον ίδιο φάκελο που περιγράψαμε πριν μαζί με όλα τα αρχεία, background εικόνες κλπ.

Επιστρέφοντας πάλι στο windows form credits, εκτός από την εντολή εξόδου που πραγματοποιείται με την εντολή Environment.Exit(0).

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    Environment.Exit(0); // to close all the forms and exit

}
```

Η εντολή αυτή είναι κοινή για όλα τα κουμπιά εξόδου, οπότε για λόγους καλύτερης παρουσίασης δεν θα παρατίθεται χωριστά κάθε φορά ο παραπάνω κώδικας για τον τερματισμό της εφαρμογής. Επίσης, ο χρήστης έχει τη δυνατότητα όσο τρέχει η εφαρμογή να μεταβεί στο προσωπικό website you3xcite.com μαθαίνοντας περισσότερες πληροφορίες για το δημιουργό της εφαρμογής. Αυτό πρακτικά επιτυγχάνεται εισάγοντας ένα linkLabel με κώδικα Process.Start και εντός παρενθέσεων ο δικτυακός τόπος που θέλουμε να μεταβούμε.

```
1 reference
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    Process.Start("www.you3xcite.com");  //linklabel to take me to my website
}
```

Τέλος, στο credits windows form, στο κουμπί επιστροφή 3 γραμμές κώδικα παρατίθενται:

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    this.Hide();                // to hide the form
    Form1 ss = new Form1();     // to take me to the login form
    ss.Show();                  // to show the login form


    _soundplayer = new SoundPlayer("Voice.wav"); //To play music with the Initialization of the Credits Form
    _soundplayer.Play();
}
```

βλέπουμε στα σχόλια του κώδικα πως ακριβώς λειτουργεί η μετάβαση στη νέα σελίδα και το soundplayer.play για να ξεκινήσει το μήνυμα εισαγωγής δεδομένων.

Επιστρέφοντας, στη σελίδα εισόδου, ο χρήστης καλείται να εισάγει τα στοιχεία εισόδου και όταν γίνεται η όποια εισαγωγή το σύστημα λειτουργεί ως εξής:

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    if (textBox1.Text == username && textBox2.Text == password)
    {
        MessageBox.Show("ΣΩΣΤΗ ΕΙΣΑΓΩΓΗ ΣΥΝΘΗΜΑΤΙΚΟΥ!");


        this.Hide();
        Menu ss = new Menu(); // constructor to take the app to the Main Menu
        ss.Show();



        _soundplayer = new SoundPlayer("Engine.wav"); //To play different music with the Initialization of the Second Form
        _soundplayer.Play();  // for playing

    }
    else
    {
        MessageBox.Show("ΕΧΕΤΕ ΕΙΣΑΓΕΙ ΛΑΘΟΣ ΣΤΟΙΧΕΙΑ! ΠΑΡΑΚΑΛΩ ΕΠΙΚΟΙΝΩΝΕΙΣΤΕ ΜΕ ΠΑΠΑΖΟΓΛΟΥ ΕΥΣΤΡΑΤΙΟ ΓΙΑ ΕΠΑΛΗΘΕΥΣΗ ΤΩΝ Σ
```

Ο κώδικας ταυτοποιεί τα στοιχεία που εισάγει ο χρήστης και τα συγκρίνει με τα καθορισμένα που έχουν δοθεί στην αρχή του windows form, δηλαδή

ROS Ytd.cs | ROS Ytd.cs [Design] | Menu.cs [Design] | Credits.cs [Design] | Form1.cs [Design] | Form1.cs | Credits.cs

Business Management | Business_Management.Form1 | button2_Click(object sender, Ever

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media; // to use music

namespace Business_Management
{
    5 references
    public partial class Form1 : Form
    {
        private SoundPlayer _soundplayer; //necessary object declaration

        string username = "Manager";   //Setting the default credentials
        string password = "13064";
```

Manager για username και 13064 για password. Στην περίπτωση που ταυτοποιηθούν σωστά τα στοιχεία εισόδου εμφανίζεται το messagebox που ενημερώνει για σωστή εισαγωγή. Σε διαφορετική περίπτωση, πραγματοποιείται το else του κώδικα και εμφανίζεται πάλι ένα messagebox, αλλά με διαφορετικό περιεχόμενο αυτή τη φορά.

Στο κεντρικό menu, ο χρήστης έχει την επιλογή της εξόδου, τον κώδικα τον είδαμε πιο πριν και τις επιλογές να μεταβεί στα 4 διαφορετικά θεματικά windows forms της εφαρμογής (ROS Ytd, ROS Previous Year, Comparisons, KPI Explanation). Προγραμματιστικά, η εντολή πίσω από κάθε κουμπί είναι ίδια με μια ελάχιστη διαφοροποίηση σε σχέση με το όνομα του windows form που θέλει να μεταβεί ο χρήστης. Χαρακτηριστικά παρατίθενται και οι 4 εντολές σε συνέπεια με τη σειρά των επιλογών που μόλις παρουσιάσαμε.

ROS Ytd:

```
1 reference
private void button9_Click(object sender, EventArgs e)
{
    this.Hide();                        // to hide the form
    ROS_Ytd ss = new ROS_Ytd();      // to take me to the ROS_Ytd from the menu
    ss.Show();
}
```

ROS Previous Year:

```
1 reference
private void button8_Click(object sender, EventArgs e)
{
    this.Hide();                                    // to hide the form
    ROS_PreviousYear ss = new ROS_PreviousYear();     // to take me to the ROS_PreviousYear from the menu
    ss.Show();
}
```

Comparisons:

```
1 reference
private void button7_Click(object sender, EventArgs e)
{
    this.Hide();                                // to hide the form
    Comparisons ss = new Comparisons();     // to take me to the Comparisons from the menu
    ss.Show();
}
```

KPI Explanation:

```
1 reference
private void button10_Click(object sender, EventArgs e)
{
    this.Hide();                                        // to hide the form
    KPI_Explanation ss = new KPI_Explanation();     // to take me to the KPI_Explanation from the menu
    ss.Show();
}
```

Όπως χαρακτηριστικά φαίνεται, στην πρώτη γραμμή δίνεται η εντολή να κρυφτεί η παρούσα φόρμα, ενώ στην Τρίτη να εμφανιστεί η νέα. Η δεύτερη γραμμή είναι αυτή που δημιουργεί το αντικείμενο ss της φόρμας που θέλουμε να μεταβούμε, με τη βοήθεια του οποίου στην τρίτη γραμμή όπως είπαμε καλείται η Show().

Στη συνέχεια της ανάλυσης μας, παραθέτουμε τον κώδικα σχετικά με τα υπόλοιπα 4 windows forms, τα οποία όλα έχουν τα ίδια χαρακτηριστικά και διέπονται από τους ίδιους κανόνες. Αριστερά σε κάθε windows form υπάρχει ένα πτυσσόμενο menu που επιτρέπει τη μετάβαση απευθείας (χωρίς να απαιτείται η μετάβαση στο κεντρικό menu).

Το συγκεκριμένο screenshot από το ROS Ytd windows form, είναι ενδεικτικό

Για τη διάρθρωση και των υπολοίπων. Και στην περίπτωση αυτή ο κώδικας μετάβασης που περιγράψαμε πριν χρησιμοποιείται με συνέπεια, μεταφέροντας το χρήστη άμεσα στην επιθυμητή επιλογή.

Τέλος, με συνέπεια προς το σύνολο της εφαρμογή, ο χρήστης έχει τη δυνατότητα της εξόδου, με τον κώδικα στο κουμπί εξόδου που έχουμε ήδη παραθέσει καθώς επίσης και το κουμπί της επιστροφής στο κεντρικό menu, πάλι με την ίδια μεθοδολογία:

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
    Menu ss = new Menu(); // constructor to take the app to the Main Menu
    ss.Show();
}
```

Στο windows form ROS τρέχοντος έτους και μόνο, ο χρήστης καλείται να εισάγει δεδομένα μόνο στα κελιά με κίτρινο χρώμα και πατώντας στο κουμπί υπολογισμός, γίνονται τα calculations που θα αναφέρουμε στη συνέχεια της ανάλυσης μας. Τα κελιά με λευκό και μπλε χρώμα, αποτελούν κελιά αποτελεσμάτων, τα δεδομένα τους προέρχονται από υπολογισμούς μετά την αποθήκευση στη βάση δεδομένων των στοιχείων που ο χρήστης εισάγει και όπως θα δούμε παρακάτω είναι read-only.

Υπολογισμοί

Τελευταίο τμήμα της ανάλυσης μας, και πριν την παράθεση των συμπερασμάτων, αποτελεί αυτό των απαραίτητων υπολογισμών και του κώδικα με τον οποίο πραγματοποιούνται στο front-end τμήμα της εφαρμογής. Όπως είδαμε στο windows form ROS Τρέχοντος έτους (Ytd),

ο χρήστης καλείται να εισάγει τα δεδομένα και κατόπιν να πατήσει το κουμπί του υπολογισμού (το οποίο σε κώδικα αναφέρεται ως button1).

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
```

Ο τρόπος με τον οποίο αυτό συμβαίνει σε γλώσσα προγραμματισμού, δηλαδή το πώς χρησιμοποιούνται τα δεδομένα για να γίνουν οι υπολογισμοί φαίνεται από το screenshot που ακολουθεί και είναι απόσπασμα από το windows form ROS Ytd.cs:

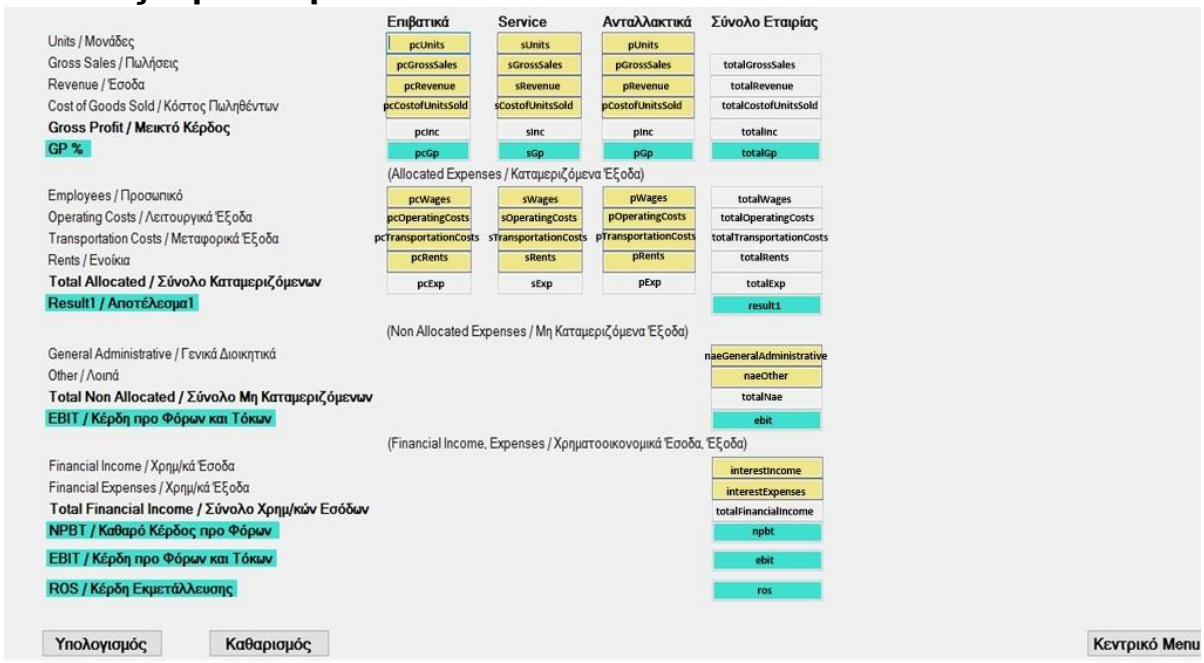
ROS PreviousYear.cs | ROS PreviousYear.cs [Design] | ROS Ytd.cs | ROS Ytd.cs [Design] | Comparisons.cs | Comparisons.cs [Design]

Business Management | Business_Management.ROS_Ytd | ROS_Ytd_Load(object sender, EventArgs e)

```
double _pcInc = _pcRevenue - _pcCostOfUnitsSold;
double _sInc = _sRevenue - _sCostOfUnitsSold;
double _pInc = _pRevenue - _pCostOfUnitsSold;

double _pcGp = _pcInc / _pcRevenue;
double _sGp = _sInc / _sRevenue;
double _pGp = _pInc / _pRevenue;

double _pcExp = _pcWages + _pcOperatingCosts + _pcTransportationCosts + _pcRents;
double _sExp = _sWages + _sOperatingCosts + _sTransportationCosts + _sRents;
double _pExp = _pWages + _pOperatingCosts + _pTransportationCosts + _pRents;

double _totalGrossSales = _pcGrossSales + _sGrossSales + _pGrossSales;
double _totalRevenue = _pcRevenue + _sRevenue + _pRevenue;
double _totalCostOfGoods = _pcCostOfUnitsSold + _sCostOfUnitsSold + _pCostOfUnitsSold;
double _totalGrossProfit = _pcInc + _sInc + _pInc;
double _totalGp = _totalGrossProfit / _totalRevenue;

double _totalWages = _pcWages + _sWages + _pWages;
double _totalOpCosts = _pcOperatingCosts + _sOperatingCosts + _pOperatingCosts;
double _totalTranspCosts = _pcTransportationCosts + _sTransportationCosts + _pTransportationCosts;
double _totalRents = _pcRents + _sRents + _pRents;
double _totalAe = _pcExp + _sExp + _pExp;
double _result1 = _totalGrossProfit - _totalAe;

double _totalNae = _naeGeneralAdministrative + _naeOther;
double _ebit = _result1 - _totalNae;

double _totalFinancialIncome = _interestIncome - _interestExpenses;
double _npbt = _ebit + _totalFinancialIncome;
double _ros = _ebit / _totalRevenue;
```

Βλέπουμε, ότι κάθε αποτέλεσμα αποθηκεύεται σε μια μεταβλητή τύπου double. (πχ _pcInc - η αντιστοιχία του database field αυτού με τα στοιχεία που απορρέουν σαν αποτέλεσμα δίνεται από τον πίνακα που ακολουθεί ενώ το αποτέλεσμα της πράξης που απαιτείται και που περιγράφεται πλήρως για τον χρήστη στο windows form KPI Explanation της εφαρμογής.

Αφού γίνουν λοιπόν οι διάφορες μαθηματικές πράξεις με τον τρόπο που περιγράφεται ανωτέρω, αποθηκεύονται τα αποτελέσματα με ονομασίες ίδιες με τα πεδία των εγγραφών των πινάκων της βάσης δεδομένων, την αντιστοιχία των οποίων έχουμε ήδη περιγράψει αλλά για λόγους παρουσίασης και συνέπειας παραθέτουμε:



Έτσι ολοκληρώνοντας, επεξηγηματικά για τη μεταβλητή pcunits αναφερόμαστε στις μονάδες των επιβατικών οχημάτων, sunits στις μονάδες του service κλπ.

Πηγαίος Κώδικας Windows Forms & Επικοινωνία με βάση

Όπως είδαμε, το front-end User Interface δημιουργήθηκε σε περιβάλλον Visual Studio με γλώσσα προγραμματισμού c#. Επίσης, σε ξεχωριστό κεφάλαιο είδαμε τη βάση δεδομένων, στην οποία υποθηκεύονται τα δεδομένα που εισάγει ο χρήστης και ανασύρονται από τη front-end εφαρμογή, τόσο για να παρουσιαστούν στο windows form ROS Previous Year και Comparisons, αλλά όσο και για να γίνουν τα απαραίτητα calculations (από το front-end τμήμα της εφαρμογής) και για να εξαχθούν τα απαραίτητα αποτελέσματα, τα οποία και αυτά με τη σειρά τους εμφανίζονται στα ROS Ytd και Comparisons εν συνεχεία. Με άλλα λόγια, για να το πούμε πιο σχηματικά, η βάση δεδομένων θα μπορούσε να προσομοιωθεί με ένα λευκό χαρτί, στο οποίο ο χρήστης εισάγοντας δεδομένα από το front-end της εφαρμογής τα εγγράφει και στη συνέχεια αποθηκεύονται. Όταν έλθει η ώρα και ζητήσει ο χρήστης από άλλα windows forms πληροφορίες, αυτές ανακτώνται από τη βάση δεδομένων μέσω του κώδικα της εφαρμογής είτε για να παρουσιαστούν, είτε για να χρησιμοποιηθούν για να γίνουν τα calculations, οι υπολογιστικές πράξεις. Όπως θα δούμε και στον κώδικα παρακάτω με exception handling που υπάρχει στον κώδικα της εφαρμογής, σε περίπτωση που ο χρήστης εισάγει δεδομένα μη αντίστοιχα με τη μορφή αυτών που θα έπρεπε να εισάγει κανονικά, η βάση δεν επιτρέπει να γίνει η ενέργεια και ο κώδικας της c# χειρίζεται το exception με τον τρόπο που θα αναλύσουμε ακολούθως.

Αναλυτικά, το αρχείο BM.cs εισάγει μια στατική κλάση BM (υπάρχει μια φορά και δεν μπορώ να φτιάξω αντικείμενο) στην οποία υπάρχουν 2 public static μεταβλητές για να είναι ορατές από παντού στην εφαρμογή μας.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business_Management
{
    18 references
    public static class Bm
    {
        public static bool emptyDb;
        public static int latestYear;
    }
}
```

Η emptyDb είναι τύπου bool, δείχνει αν η βάση δεδομένων είναι άδεια και αποθηκέυει μια τιμή. Αν είναι άδεια αποθηκέυει true, αν όχι αποθηκεύει false.

Η latestYear δείχνει ποιο είναι το τελευταίο έτος που έχει καταχωρηθεί στη βάση δεδομένων (με άλλα λόγια δείχνει ποιο είναι το μεγαλύτερο καταχωρημένο έτος και όχι το νέο που πρόκειται να καταχωρηθεί). Έτσι, οι 2 παραπάνω static μεταβλητές της κλάσης BM είναι global για όλες τις φόρμες, σε όλη την εφαρμογή.

Το αρχείο Credits.cs, αποτελεί μια σύντομη αναφορά στον δημιουργό αυτής της εφαρμογής και περιλαμβάνει μια επιπρόσθετη βιβλιοθήκη, τόσο για τη μουσική επένδυση της φόρμας αυτής όσο και για το linkLabel που οδηγεί στο website: you3xcite.com.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media; // to use music
using System.Diagnostics; // library for linklabel

namespace Business_Management
{
    5 references
    public partial class Credits : Form
    {
        private SoundPlayer _soundplayer; //necessary object declaration

        1 reference
        public Credits()
        {
            InitializeComponent();
            _soundplayer = new SoundPlayer("Credits.wav");   //To play music with the Initialization of the Credits Form
            _soundplayer.PlayLooping();                      // for Looping
        }

        1 reference
        private void button3_Click(object sender, EventArgs e)
        {
            Environment.Exit(0); // to close all the forms and exit
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();                // to hide the form
            Form1 ss = new Form1();     // to take me to the login form
            ss.Show();                  // to show the login form
        }

        1 reference
        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            Process.Start("www.you3xcite.com");  //linklabel to take me to my website
        }
    }
}
```

Στο σημείο αυτό πρέπει να αναφερθεί ότι καθώς τις λειτουργίες πλοήγησης και τον κώδικά τους τα είδαμε σε ξεχωριστό τμήμα της ανάλυσης μας (κώδικας front-end εφαρμογής), στην παρούσα θα αναλύσουμε ότι έχει να κάνει σε σχέση με την επικοινωνία με την βάση δεδομένων. Το παρόν αρχείο δεν επικοινωνεί με την βάση οπότε στην παρούσα ενότητα δεν χρήζει περεταίρω ανάλυσης.

Το αρχείο Form1.cs και ο κώδικας του σε ότι έχει να κάνει με την επικοινωνία με τη βάση δεδομένων έχει μια χαρακτηριστική λειτουργία:

```
2 references
public Form1()
{
    InitializeComponent();
    _soundplayer = new SoundPlayer("Voice.wav"); //To play music with the Initialization of the Credits Form
    _soundplayer.Play();

    OleDbConnection conn = new OleDbConnection();
    OleDbCommand sqlCmd = new OleDbCommand();
    OleDbDataReader dbReader;
    DataTable dTable = new DataTable();
    int max = 0;
    int currentYear = 0;
    try
    {
        conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=sales.accdb;Persist Security Info=F
        conn.Open();
        sqlCmd.Connection = conn;
        sqlCmd.CommandText = "SELECT anno FROM inputFields";
        dbReader = sqlCmd.ExecuteReader();
        dTable.Load(dbReader);
        conn.Close();
    }
    catch (Exception ex)
    {
        if (conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
        MessageBox.Show(ex.Message);
        MessageBox.Show(ex.ToString());
    }
```

```
//find max (latest) year in db and store it to an app-wide global int "latestYear"
if (dTable.Rows.Count > 0)
{
    Bm.emptyDb = false;
    max = Convert.ToInt32(dTable.Rows[0][0]);
    for (int j = 1; j < dTable.Rows.Count; j++)
    {
        currentYear = Convert.ToInt32(dTable.Rows[j][0]);
        if (currentYear > max)
        {
            max = currentYear;
        }
    }
    Bm.latestYear = max;
    //MessageBox.Show("Latest year is: " + Bm.latestYear.ToString());
}
else
{
    Bm.emptyDb = true;
    //MessageBox.Show("No data found in DB");
}
```

Αμέσως μόλις φορτώσει το windows form αυτό, γίνεται άμεσα σύνδεση με τη βάση δεδομένων. Σκοπός είναι να ενημερωθούν οι 2 global BM μεταβλητές (που είδαμε στο BM.cs αρχείο) πριν καν βάλει ο χρήστης τα στοιχεία του ή επιλέξει να επισκεφτεί το credits windows form, πριν αρχίσει η εφαρμογή την κύρια λειτουργία της. Όλος ο κώδικας λειτουργεί μέσα στο try{ } section, ενώ στο catch (exception ex){ } διευθετείται το ενδεχόμενο εμφάνισης σφάλματος ενημερώνοντας με ένα Message Box τον χρήστη.
Ακολούθως, τρέχει ένας βρόγχος

```
//find max (latest) year in db and store it to an app-wide global int "latestYear"
if (dTable.Rows.Count > 0)
{
    Bm.emptyDb = false;
    max = Convert.ToInt32(dTable.Rows[0][0]);
    for (int j = 1; j < dTable.Rows.Count; j++)
    {
        currentYear = Convert.ToInt32(dTable.Rows[j][0]);
        if (currentYear > max)
        {
            max = currentYear;
        }
    }
    Bm.latestYear = max;
```

που αποθηκεύει / ενημερώνει την άλλη global μεταβλητή που προαναφέραμε (στο BM.cs), την latestYear, και θέτει ως μέγιστη την τιμή που βρίσκει από τον βρόγχο.

Το αρχείο Menu.cs δεν έρχεται σε επικοινωνία με τη βάση δεδομένων, όπως και το αρχείο KPI Explanation. Για το λόγο αυτό, παρατίθεται απλά ο κώδικας τους, η λειτουργίες των οποίων έχουν αναλυθεί σε προηγούμενο τμήμα της ανάλυσης μας (κώδικας front-end εφαρμογής). Ακολουθεί στις επόμενες σελίδες διαδοχικά screenshot για το κάθε windows form:

Menu.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Business_Management
{
    13 references
    public partial class Menu : Form
    {
        5 references
        public Menu()
        {
            InitializeComponent();
        }

        1 reference
        private void button3_Click(object sender, EventArgs e)
        {
            Environment.Exit(0); // to close all the forms and exit
        }

        1 reference
        private void button4_Click(object sender, EventArgs e)
        {
            this.Hide();                    // to hide the form
            ROS_Ytd ss = new ROS_Ytd();     // to take me to the ROS_Ytd from the menu
            ss.Show();
        }

        1 reference
        private void button5_Click(object sender, EventArgs e)
        {
            this.Hide();                                    // to hide the form
            ROS_PreviousYear ss = new ROS_PreviousYear();   // to take me to the ROS_PreviousYear from the menu
            ss.Show();
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();                            // to hide the form
            Comparisons ss = new Comparisons();     // to take me to the Comparisons from the menu
            ss.Show();
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();                                    // to hide the form
            KPI_Explanation ss = new KPI_Explanation();     // to take me to the KPI_Explanation from the menu
            ss.Show();
        }
    }
}
```

KPI Explanation.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Business_Management
{
    11 references
    public partial class KPI_Explanation : Form
    {
        4 references
        public KPI_Explanation()
        {
            InitializeComponent();
        }

        1 reference
        private void button3_Click(object sender, EventArgs e)
        {
            Environment.Exit(0); // to close all the forms and exit
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Menu ss = new Menu(); // constructor to take the app to the Main Menu
            ss.Show();
        }

        1 reference
        private void rOSΤρέχοντοςΈτουςToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Hide();                    // to hide the form
            ROS_Ytd ss = new ROS_Ytd();     // to take me to the ROS_Ytd from the menu
            ss.Show();

        }

        1 reference
        private void rOSΠροηγούμενουΈτουςToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Hide();                                    // to hide the form
            ROS_PreviousYear ss = new ROS_PreviousYear();     // to take me to the ROS_PreviousYear from the menu
            ss.Show();

        }

        1 reference
        private void συγκρίσειςToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Hide();                            // to hide the form
            Comparisons ss = new Comparisons();     // to take me to the Comparisons from the menu
            ss.Show();

        }
    }
}
```

Όπως βλέπουμε, τα 2 windows forms αυτά, απλά περιέχουν κουμπιά και menu πλοήγησης, ο κώδικας των οποίων όπως έχει ήδη αναλυθεί πραγματοποιείται on-click, δηλαδή με την πραγματοποίηση κάποιου γεγονότος από τον χρήστη.

Στα τελευταία 3 αρχεία (ROS Ytd.cs, ROS PreviousYear.cs και Comparisons.cs) όπως προκύπτουν από τα αντίστοιχα design windows forms γίνεται η μεγαλύτερη επικοινωνία με τη βάση δεδομένων και ως εκ τούτου αναλύονται διεξοδικά παρακάτω.

Στο ROS Ytd.cs, γίνεται declaration και ομαδοποιεί όλα τα textboxes που περιέχει χαρακτηριστικά σε 2 μεταβλητές με λίστες από textboxes (ορατές σε όλες τις συναρτήσεις της εφαρμογής), σε αυτά των input (όσα χρειάζονται για να βάλει τα δεδομένα ο χρήστης που θα αποθηκευτούν στον πρώτο πίνακα της βάσης), και σε αυτά των output (όσα χρειάζονται για να εμφανίσουν τα αποτελέσματα των υπολογισμών).

```
namespace Business_Management
{
    10 references
    public partial class ROS_Ytd : Form
    {
        int yearToBeWritten;
        List<TextBox> inputFields;
        List<TextBox> outputFields;
```

Ακολούθως, γίνεται initialization για κάθε μια από αυτές τις 2 μεταβλητές, δηλαδή:

```
inputFields = new List<TextBox>();
```

και

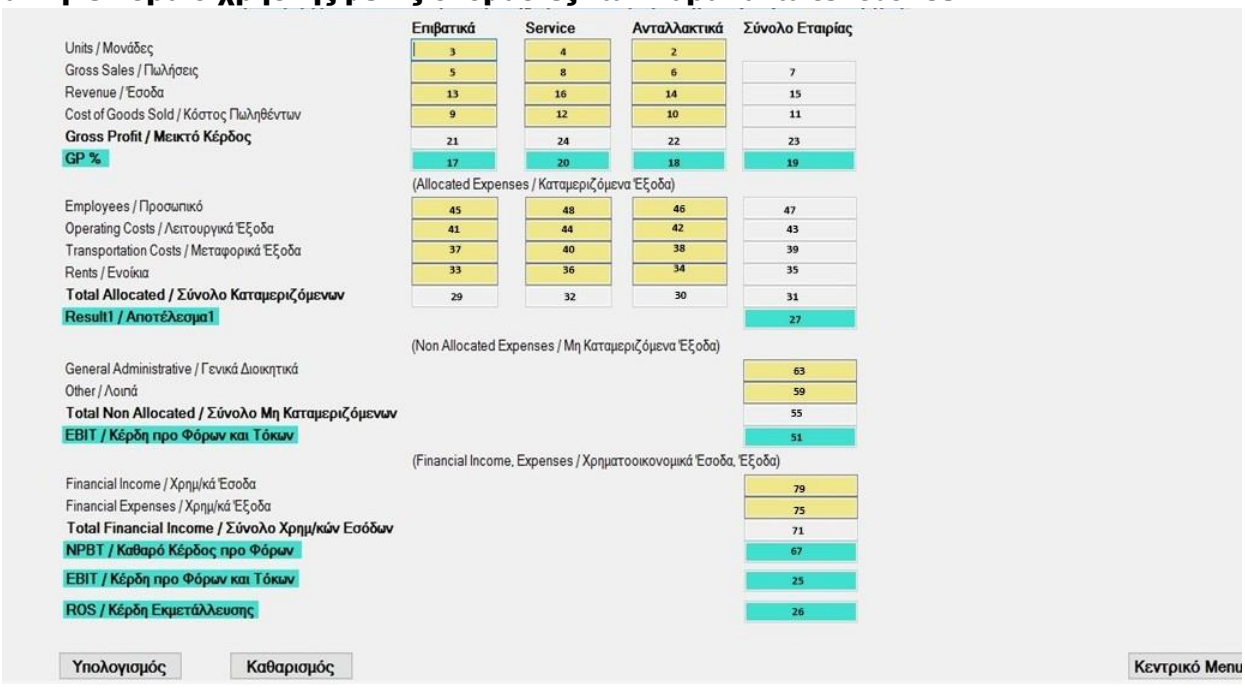
```
outputFields = new List<TextBox>();
```

Ακολούθως, με την ίδια λογική μπαίνουν τα πεδία από τα οποία θα αποτελείται η κάθε λίστα (inputFields και outputFields):

```
inputFields.Add(textBox3);
inputFields.Add(textBox4);
inputFields.Add(textBox2);
inputFields.Add(textBox5);
inputFields.Add(textBox8);
inputFields.Add(textBox6);
inputFields.Add(textBox13);
inputFields.Add(textBox16);
inputFields.Add(textBox14);
inputFields.Add(textBox9);
inputFields.Add(textBox12);
inputFields.Add(textBox10);
inputFields.Add(textBox45);
inputFields.Add(textBox48);
inputFields.Add(textBox46);
inputFields.Add(textBox41);
inputFields.Add(textBox44);
inputFields.Add(textBox42);
inputFields.Add(textBox37);
inputFields.Add(textBox40);
inputFields.Add(textBox38);
inputFields.Add(textBox33);
inputFields.Add(textBox36);
inputFields.Add(textBox34);
inputFields.Add(textBox63);
inputFields.Add(textBox59);
inputFields.Add(textBox79);
inputFields.Add(textBox75);
```

```
outputFields.Add(textBox7);
outputFields.Add(textBox15);
outputFields.Add(textBox11);
outputFields.Add(textBox23);
outputFields.Add(textBox19);
outputFields.Add(textBox47);
outputFields.Add(textBox43);
outputFields.Add(textBox39);
outputFields.Add(textBox35);
outputFields.Add(textBox31);
outputFields.Add(textBox27);
outputFields.Add(textBox55);
outputFields.Add(textBox51);
outputFields.Add(textBox71);
outputFields.Add(textBox67);
outputFields.Add(textBox25);
outputFields.Add(textBox26);
outputFields.Add(textBox21);
outputFields.Add(textBox17);
outputFields.Add(textBox24);
outputFields.Add(textBox20);
outputFields.Add(textBox22);
outputFields.Add(textBox18);
outputFields.Add(textBox29);
outputFields.Add(textBox32);
outputFields.Add(textBox30);
```

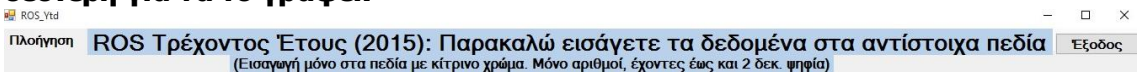
Με τον τρόπο αυτό γεμίζουν τα πεδία των 2 λιστών. Για λόγους παρουσίασης, ακολουθεί screenshot την αντιστοιχία των textboxes της εφαρμογής που αλληλεπιδρά ο χρήστης με τις ονομασίες των παραπάνω textboxes:



Στη συνέχεια καλούνται 2 συναρτήσεις:

```
determineYearToBeWritten();
refreshFormLabel();
```

Η πρώτη καθορίζει ποιο έτος θα γραφτεί στο Label του windows form και η δεύτερη για να το γράψει:



Ο κώδικας με τον οποίο γίνεται αυτό φαίνεται παρακάτω:

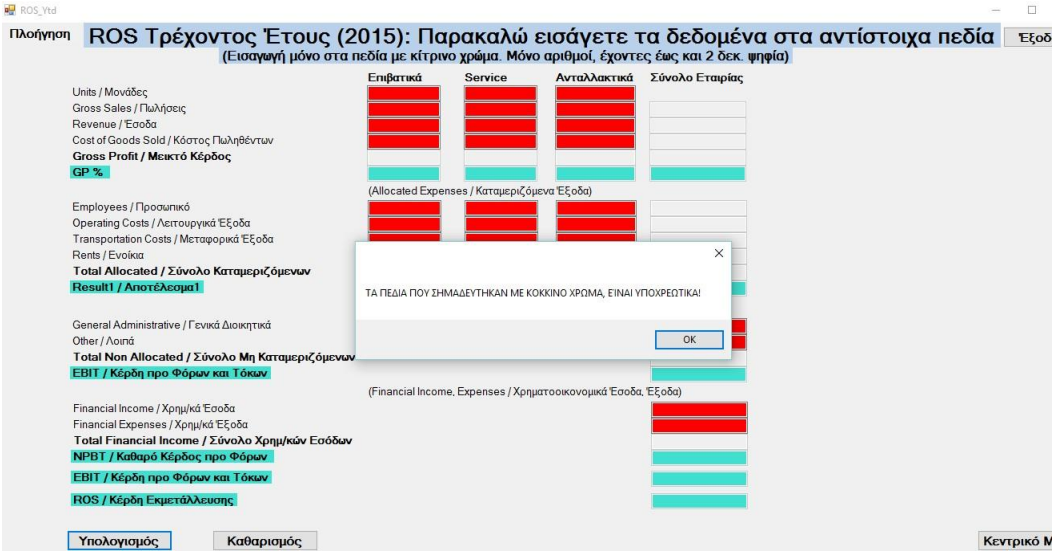
```
2 references
private void determineYearToBeWritten()
{
    if (Bm.emptyDb)
    {
        yearToBeWritten = Convert.ToInt32(DateTime.Now.Year);
    }
    else
    {
        yearToBeWritten = Bm.latestYear + 1;
    }
}
```

Αν η βάση είναι άδεια (δηλαδή το Boolean γίνεται true), τότε μετατρέπει σε ακέραιο (με το Convert.ToInt32) το έτος που δίνει η DateTime. Αλλιώς, αποθηκεύει στη μεταβλητή yearToBeWritten, το τελευταίο που υπάρχει συν 1, δηλαδή το επόμενο.
Αντίστοιχα, για την άλλη συνάρτηση:

```
2 references
private void refreshFormLabel()
{
    label1.Text = "ROS Τρέχοντος Έτους (" + yearToBeWritten.ToString() +"): Παρακαλώ εισάγετε τα δεδομένα στα αντίστοιχα π
}
```

μετατρέπει σε αλφαριθμητικό το έτος που έκανε determine πριν, δηλαδή το yearToBeWritten (με το: yearToBeWritten.ToString).

Στο σημείο αυτό, θα δούμε τι γίνεται όταν ο χρήστης πατάει το κουμπί υπολογισμού, το button1 δηλαδή. Σε περίπτωση που ο χρήστης προσπαθήσει να πατήσει το κουμπί χωρίς να έχει συμπληρώσει έστω ένα απαραίτητο πεδίο τότε αυτό ή αυτά γίνονται κόκκινα



Και εμφανίζεται μήνυμα στον χρήστη ότι πρέπει να τα συμπληρώσει γιατί είναι υποχρεωτικά. Μέσα στον κώδικα του button1 (κουμπί υπολογισμού) αρχικά με την resetColor συνάρτηση επαναφέρεται το χακί-κιτρινωπό χρώμα των textboxes (την πρώτη φορά που τρέχει επαναφέρεται χωρίς να έχει κοκκινήσει αλλά αυτό δεν επηρεάζει την εφαρμογή).

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    resetColor();
```

Ο κώδικας της resetColor() είναι απλός και με ένα foreach (αντί για τον κλασσικό τρόπο παρουσίασης του βρόγχου) στα inputFields, στα πεδία που αναφέρονται στα δεδομένα που εισάγει ο χρήστης κάνει χακί όλα τα textboxes που έχουν διαφορετικό χρώμα:

```
private void resetColor()
{
    foreach (TextBox t in inputFields)
    {
        if (t.BackColor != Color.Khaki)
        {
            t.BackColor = Color.Khaki;
        }
    }
}
```

Αμέσως μετά όλος ο κώδικας μπαίνει σε ένα μεγάλο if else concept, μέσα στο οποίο γίνονται όσα θέλουμε αλλά και handling κάποια exceptions. Πριν δούμε

τον κώδικα στο σύνολο του αλλά και αποσπασματικά η συνάρτηση allFields() που θα δούμε παρακάτω λειτουργεί ως εξής:

```
1 reference
private bool allFields()
{
    int empty = 0;
    foreach (TextBox t in inputFields)
    {
        if (string.IsNullOrWhiteSpace(t.Text))
        {
            t.BackColor = Color.Red;
            empty++;
        }
    }
    if (empty > 0)
    {
        if (empty == 1)
        {
            MessageBox.Show("ΤΟ ΠΕΔΙΟ ΠΟΥ ΣΗΜΑΔΕΥΤΗΚΕ ΜΕ ΚΟΚΚΙΝΟ ΧΡΩΜΑ, ΕΊΝΑΙ ΥΠΟΧΡΕΩΤΙΚΟ!");
        }
        else if (empty > 1)
        {
            MessageBox.Show("ΤΑ ΠΕΔΙΑ ΠΟΥ ΣΗΜΑΔΕΥΤΗΚΑΝ ΜΕ ΚΟΚΚΙΝΟ ΧΡΩΜΑ, ΕΊΝΑΙ ΥΠΟΧΡΕΩΤΙΚΑ!");
        }
        return false;
    }
    else
    {
        return true;
    }
}
```

Τρέχει και επιστρέφει true ή false. Για κάθε κίτρινο κελί, αν είναι κενό κάνει το χρώμα του κόκκινο και μετράει τα πεδία. Η περίπτωση να υπάρχει έστω και ένα κενό πεδίο με τη σειρά της χωρίζεται σε 2 περιπτώσεις, να υπάρχει μόνο ένα πεδίο ή να υπάρχουν περισσότερα από ένα πεδία. Έτσι, διαφοροποιείται το μήνυμα λάθους και αν το πεδίο είναι μονό ένα εμφανίζει «Το πεδίο», ενώ στην άλλη περίπτωση εμφανίζει «Τα πεδία». Ακολούθως επιστρέφει false και στις 2 περιπτώσεις ενώ αν δεν υπάρχει σφάλμα επιστρέφει true και σταματά (με το return). Λόγω πολυπλοκότητας αυτού του form, θα παραθέσουμε τον κώδικα και συνολικά αλλά και θα εξηγηθεί σε κάθε περίπτωση μεμονωμένα για να διασφαλιστεί ότι η ανάλυση έγινε με τον καλύτερο δυνατό τρόπο.

Συγκεκριμένα ο κώδικας στο σύνολο του:

```
if (allFields())
{
    bool input2DbError = false;
    bool output2DbError = false;
    OleDbConnection conn = new OleDbConnection();
    OleDbCommand sqlCmdInputFields = new OleDbCommand();
    OleDbCommand sqlCmdOutputFields = new OleDbCommand();
    conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=sales.accdb;Persist Security Info=False;";
    sqlCmdInputFields.Connection = conn;
    sqlCmdOutputFields.Connection = conn;

    int _pcUnits = Convert.ToInt32(textBox3.Text);
    int _sUnits = Convert.ToInt32(textBox4.Text);
    int _pUnits = Convert.ToInt32(textBox2.Text);

    double _pcGrossSales = Convert.ToDouble(textBox5.Text);
    double _sGrossSales = Convert.ToDouble(textBox8.Text);
    double _pGrossSales = Convert.ToDouble(textBox6.Text);

    double _pcRevenue = Convert.ToDouble(textBox13.Text);
    double _sRevenue = Convert.ToDouble(textBox16.Text);
    double _pRevenue = Convert.ToDouble(textBox14.Text);

    double _pcCostOfUnitsSold = Convert.ToDouble(textBox9.Text);
    double _sCostOfUnitsSold = Convert.ToDouble(textBox12.Text);
    double _pCostOfUnitsSold = Convert.ToDouble(textBox10.Text);

    double _pcWages = Convert.ToDouble(textBox45.Text);
    double _sWages = Convert.ToDouble(textBox48.Text);
    double _pWages = Convert.ToDouble(textBox46.Text);

   double _pcOperatingCosts = Convert.ToDouble(textBox41.Text);
   double _sOperatingCosts = Convert.ToDouble(textBox44.Text);
   double _pOperatingCosts = Convert.ToDouble(textBox42.Text);

   double _pcTransportationCosts = Convert.ToDouble(textBox37.Text);
   double _sTransportationCosts = Convert.ToDouble(textBox40.Text);
   double _pTransportationCosts = Convert.ToDouble(textBox38.Text);

   double _pcRents = Convert.ToDouble(textBox33.Text);
   double _sRents = Convert.ToDouble(textBox36.Text);
   double _pRents = Convert.ToDouble(textBox34.Text);

   double _naeGeneralAdministrative = Convert.ToDouble(textBox63.Text);
   double _naeOther = Convert.ToDouble(textBox59.Text);

   double _interestIncome = Convert.ToDouble(textBox79.Text);
   double _interestExpenses = Convert.ToDouble(textBox75.Text);


   double _pcInc = _pcRevenue - _pcCostOfUnitsSold;
   double _sInc = _sRevenue - _sCostOfUnitsSold;
   double _pInc = _pRevenue - _pCostOfUnitsSold;

   double _pcGp = _pcInc / _pcRevenue;
   double _sGp = _sInc / _sRevenue;
   double _pGp = _pInc / _pRevenue;

   double _pcExp = _pcWages + _pcOperatingCosts + _pcTransportationCosts + _pcRents;
   double _sExp = _sWages + _sOperatingCosts + _sTransportationCosts + _sRents;
   double _pExp = _pWages + _pOperatingCosts + _pTransportationCosts + _pRents;
```

```
    double _totalGrossSales = _pcGrossSales + _sGrossSales + _pGrossSales;
    double _totalRevenue = _pcRevenue + _sRevenue + _pRevenue;
    double _totalCostOfGoods = _pcCostOfUnitsSold + _sCostOfUnitsSold + _pCostOfUnitsSold;
    double _totalGrossProfit = _pcInc + _sInc + _pInc;
    double _totalGp = _totalGrossProfit / _totalRevenue;

    double _totalWages = _pcWages + _sWages + _pWages;
    double _totalOpCosts = _pcOperatingCosts + _sOperatingCosts + _pOperatingCosts;
    double _totalTranspCosts = _pcTransportationCosts + _sTransportationCosts + _pTransportationCosts;
    double _totalRents = _pcRents + _sRents + _pRents;
    double _totalAe = _pcExp + _sExp + _pExp;
    double _result1 = _totalGrossProfit - _totalAe;

    double _totalNae = _naeGeneralAdministrative + _naeOther;
    double _ebit = _result1 - _totalNae;

    double _totalFinancialIncome = _interestIncome - _interestExpenses;
    double _npbt = _ebit + _totalFinancialIncome;
    double _ros = _ebit / _totalRevenue;

  sqlCmdInputFields.CommandText = @"INSERT INTO inputFields ([anno],
      [pcUnits], [pcGrossSales], [pcRevenue], [pcCostOfUnitsSold],
      [sUnits], [sGrossSales], [sRevenue], [sCostOfUnitsSold],
      [pUnits], [pGrossSales], [pRevenue], [pCostOfUnitsSold],
      [pcWages], [pcOperatingCosts], [pcTransportationCosts], [pcRents],
      [sWages], [sOperatingCosts], [sTransportationCosts], [sRents],
      [pWages], [pOperatingCosts], [pTransportationCosts], [pRents],
      [naeGeneralAdministrative], [naeOther],
      [interestIncome], [interestExpenses])
      VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
  sqlCmdInputFields.Parameters.AddRange(new OleDbParameter[]{
      new OleDbParameter("@anno", yearToBeWritten),
      new OleDbParameter("@pcUnits", _pcUnits),
      new OleDbParameter("@pcGrossSales", _pcGrossSales),
      new OleDbParameter("@pcRevenue", _pcRevenue),
      new OleDbParameter("@pcCostOfUnitsSold", _pcCostOfUnitsSold),
      new OleDbParameter("@sUnits", _sUnits),
      new OleDbParameter("@sGrossSales", _sGrossSales),
      new OleDbParameter("@sRevenue", _sRevenue),
      new OleDbParameter("@sCostOfUnitsSold", _sCostOfUnitsSold),
      new OleDbParameter("@pUnits", _pUnits),
      new OleDbParameter("@pGrossSales", _pGrossSales),
      new OleDbParameter("@pRevenue", _pRevenue),
      new OleDbParameter("@pCostOfUnitsSold", _pCostOfUnitsSold),
      new OleDbParameter("@pcWages", _pcWages),
      new OleDbParameter("@pcOperatingCosts", _pcOperatingCosts),
      new OleDbParameter("@pcTransportationCosts", _pcTransportationCosts),
      new OleDbParameter("@pcRents", _pcRents),
      new OleDbParameter("@sWages", _sWages),
      new OleDbParameter("@sOperatingCosts", _sOperatingCosts),
      new OleDbParameter("@sTransportationCosts", _sTransportationCosts),
      new OleDbParameter("@sRents", _sRents),
      new OleDbParameter("@pWages", _pWages),
      new OleDbParameter("@pOperatingCosts", _pOperatingCosts),
```

```
        new OleDbParameter("@pTransportationCosts", _pTransportationCosts),
        new OleDbParameter("@pRents", _pRents),
        new OleDbParameter("@naeGeneralAdministrative", _naeGeneralAdministrative),
        new OleDbParameter("@naeOther", _naeOther),
        new OleDbParameter("@interestIncome", _interestIncome),
        new OleDbParameter("@interestExpenses", _interestExpenses)
    });
    try
    {
        conn.Open();
        sqlCmdInputFields.ExecuteNonQuery();
        conn.Close();
    }
    catch (Exception ex)
    {
        if (conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
        MessageBox.Show("ΣΦΑΛΜΑ ΚΑΤΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΧΡΗΣΤΗ ΣΤΗ ΒΔ.\n\n" + ex.Message);
        MessageBox.Show(ex.ToString());
        input2DbError = true;
    }

    sqlCmdOutputFields.CommandText = @"INSERT INTO outputFields ([anno],
        [pcInc], [pcGp], [pcExp], [sInc], [sGp], [sExp], [pInc], [pGp], [pExp],
        [totalGrossSales], [totalRevenue], [totalCostOfGoods], [totalGrossProfit], [totalGp],
        [totalWages], [totalOpCosts], [totalTranspCosts], [totalRents], [totalAe], [result1],
        [totalNae], [ebit], [totalFinancialIncome], [npbt], [ros])
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    sqlCmdOutputFields.Parameters.AddRange(new OleDbParameter[]{
        new OleDbParameter("@anno", yearToBeWritten),
        new OleDbParameter("@pcInc", _pcInc),
        new OleDbParameter("@pcGp", _pcGp),
        new OleDbParameter("@pcExp", _pcExp),
        new OleDbParameter("@sInc", _sInc),
        new OleDbParameter("@sGp", _sGp),
        new OleDbParameter("@sExp", _sExp),
        new OleDbParameter("@pInc", _pInc),
        new OleDbParameter("@pGp", _pGp),
        new OleDbParameter("@pExp", _pExp),
        new OleDbParameter("@totalGrossSales", _totalGrossSales),
        new OleDbParameter("@totalRevenue", _totalRevenue),
        new OleDbParameter("@totalCostOfGoods", _totalCostOfGoods),
        new OleDbParameter("@totalGrossProfit", _totalGrossProfit),
        new OleDbParameter("@totalGp", _totalGp),
        new OleDbParameter("@totalWages", _totalWages),
        new OleDbParameter("@totalOpCosts", _totalOpCosts),
        new OleDbParameter("@totalTranspCosts", _totalTranspCosts),
        new OleDbParameter("@totalRents", _totalRents),
        new OleDbParameter("@totalAe", _totalAe),
        new OleDbParameter("@result1", _result1),
        new OleDbParameter("@totalNae", _totalNae),
        new OleDbParameter("@ebit", _ebit),
        new OleDbParameter("@totalFinancialIncome", _totalFinancialIncome),
```

```
        new OleDbParameter("@npbt", _npbt),
        new OleDbParameter("@ros", _ros)
    });
    try
    {
        conn.Open();
        sqlCmdOutputFields.ExecuteNonQuery();
        conn.Close();
    }
    catch (Exception ex)
    {
        if (conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
        MessageBox.Show("ΣΦΑΛΜΑ ΚΑΤΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΣΤΗ ΒΔ.\n\n" + ex.Message);
        MessageBox.Show(ex.ToString());
        output2DbError = true;
    }
    if (!(input2DbError || output2DbError))
    {
        button1.Enabled = false;
        Bm.emptyDb = false;
        Bm.latestYear = yearToBeWritten;
        determineYearToBeWritten();
        MessageBox.Show("ΕΠΙΤΥΧΗΣ ΚΑΤΑΧΩΡΗΣΗ ΤΟΥ ΕΤΟΥΣ " + Bm.latestYear + "\n\nΕΑΝ ΕΠΙΘΥΜΕΙΤΕ ΝΕΑ ΚΑΤΑΧΩΡΗΣΗ,
        textBox7.Text = _totalGrossSales.ToString();
        textBox15.Text = _totalRevenue.ToString();
        textBox11.Text = _totalCostOfGoods.ToString();
        textBox23.Text = _totalGrossProfit.ToString();
        textBox19.Text = _totalGp.ToString();
        textBox21.Text = _pcInc.ToString();
        textBox17.Text = _pcGp.ToString();
        textBox24.Text = _sInc.ToString();
        textBox20.Text = _sGp.ToString();
        textBox22.Text = _pInc.ToString();
        textBox18.Text = _pGp.ToString();
        textBox29.Text = _pcExp.ToString();
        textBox32.Text = _sExp.ToString();
        textBox30.Text = _pExp.ToString();
        textBox47.Text = _totalWages.ToString();
        textBox43.Text = _totalOpCosts.ToString();
        textBox39.Text = _totalTranspCosts.ToString();
        textBox35.Text = _totalRents.ToString();
        textBox31.Text = _totalAe.ToString();
        textBox27.Text = _result1.ToString();
        textBox55.Text = _totalNae.ToString();
        textBox51.Text = _ebit.ToString();
        textBox71.Text = _totalFinancialIncome.ToString();
        textBox67.Text = _npbt.ToString();
        textBox25.Text = _ebit.ToString();
        textBox26.Text = _ros.ToString();
```

```
    }
    else //make the entire operation atomic
    {
        if (input2DbError && !(output2DbError))
        {
            OleDbCommand deleteFromOutputFields = new OleDbCommand(@"DELETE * FROM outputFields WHERE anno = @anno", 
            deleteFromOutputFields.Parameters.Add(new OleDbParameter(@"anno", yearToBeWritten));
            try
            {
                conn.Open();
                deleteFromOutputFields.ExecuteNonQuery();
                conn.Close();
            }
            catch (Exception ex)
            {
                if (conn.State == ConnectionState.Open)
                {
                    conn.Close();
                }
                MessageBox.Show(ex.Message);
                MessageBox.Show("ΣΦΑΛΜΑ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ.\nΤΕΡΜΑΤΙΣΤΕ ΑΜΕΣΩΣ ΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΔΙΑΓΡΑΨΤΕ ΤΗΝ ΕΓΓΡΑΦΗ
            }
        }
             }
             else if (!(input2DbError) && output2DbError)
             {
                 OleDbCommand deleteFromInputFields = new OleDbCommand(@"DELETE * FROM inputFields WHERE anno = @anno", conn
                 deleteFromInputFields.Parameters.Add(new OleDbParameter(@"anno", yearToBeWritten));
                 try
                 {
                     conn.Open();
                     deleteFromInputFields.ExecuteNonQuery();
                     conn.Close();
                 }
                 catch (Exception ex)
                 {
                     if (conn.State == ConnectionState.Open)
                     {
                         conn.Close();
                     }
                     MessageBox.Show(ex.Message);
                     MessageBox.Show("ΣΦΑΛΜΑ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ.\nΤΕΡΜΑΤΙΣΤΕ ΑΜΕΣΩΣ ΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΔΙΑΓΡΑΨΤΕ ΤΗΝ ΕΓΓΡΑΦΗ "
                 }
             }
         }
     }
}
```

Μετά την παράθεση ολόκληρου του κώδικα που είναι αναγκαίος για την κατανόηση του σκεπτικού βλέπουμε τα επιμέρους χαρακτηριστικά. Στο πρώτο if λοιπόν, η allFields() είναι bool όπως είδαμε (επιστρέφει true ή false), αν επιστρέψει true συνεχίζει, ενώ αν επιστρέψει false τερματίζει. Θέτουμε 2 μεταβλητές (input2DbError, output2DbError) ίσες με false και είναι φυσικά τύπου bool. Με άλλα λόγια δεν υπάρχει σφάλμα στη βάση δεδομένων, ούτε στον πίνακα εισαγωγής δεδομένων που ελέγχεται από την πρώτη μεταβλητή, ούτε στον πίνακα αποτελεσμάτων. Στην περίπτωση αυτήγίνεται η σύνδεση με τη βάση δεδομένων

```
OleDbConnection conn = new OleDbConnection();
OleDbCommand sqlCmdInputFields = new OleDbCommand();
OleDbCommand sqlCmdOutputFields = new OleDbCommand();
conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=sales.accdb;Persist Security Info=False;";
sqlCmdInputFields.Connection = conn;
sqlCmdOutputFields.Connection = conn;
```

και δίνονται ταυτόχρονα 2 εντολές μια για κάθε πίνακα: για τον InputFields και για τον OutputFields.

Ακολούθως, μια αλληλουχία ενεργειών πραγματοποιείται και τα περιεχόμενα των textboxes αφού μετατραπούν σε αριθμό double (με την εξαίρεση των τριών πρώτων που είναι ακέραιοι) αποθηκεύονται σε μεταβλητές τύπου double.

```
    int _pcUnits = Convert.ToInt32(textBox3.Text);
    int _sUnits = Convert.ToInt32(textBox4.Text);
    int _pUnits = Convert.ToInt32(textBox2.Text);

    double _pcGrossSales = Convert.ToDouble(textBox5.Text);
    double _sGrossSales = Convert.ToDouble(textBox8.Text);
    double _pGrossSales = Convert.ToDouble(textBox6.Text);

double _pcRevenue = Convert.ToDouble(textBox13.Text);
double _sRevenue = Convert.ToDouble(textBox16.Text);
double _pRevenue = Convert.ToDouble(textBox14.Text);

double _pcCostOfUnitsSold = Convert.ToDouble(textBox9.Text);
double _sCostOfUnitsSold = Convert.ToDouble(textBox12.Text);
double _pCostOfUnitsSold = Convert.ToDouble(textBox10.Text);

double _pcWages = Convert.ToDouble(textBox45.Text);
double _sWages = Convert.ToDouble(textBox48.Text);
double _pWages = Convert.ToDouble(textBox46.Text);

double _pcOperatingCosts = Convert.ToDouble(textBox41.Text);
double _sOperatingCosts = Convert.ToDouble(textBox44.Text);
double _pOperatingCosts = Convert.ToDouble(textBox42.Text);

double _pcTransportationCosts = Convert.ToDouble(textBox37.Text);
double _sTransportationCosts = Convert.ToDouble(textBox40.Text);
double _pTransportationCosts = Convert.ToDouble(textBox38.Text);

double _pcRents = Convert.ToDouble(textBox33.Text);
double _sRents = Convert.ToDouble(textBox36.Text);
double _pRents = Convert.ToDouble(textBox34.Text);

double _naeGeneralAdministrative = Convert.ToDouble(textBox63.Text);
double _naeOther = Convert.ToDouble(textBox59.Text);

double _interestIncome = Convert.ToDouble(textBox79.Text);
double _interestExpenses = Convert.ToDouble(textBox75.Text);
```

Οι μεταβλητές αυτές παρακάτω θα χρησιμοποιηθούν για να γεμίσουν τα στοιχεία του πίνακα InputFields.

Στη συνέχεια, ακολουθούν τα calculations που έχουν ήδη περιγραφεί σε προηγούμενη ενότητα:

```
double _pcInc = _pcRevenue - _pcCostOfUnitsSold;
double _sInc = _sRevenue - _sCostOfUnitsSold;
double _pInc = _pRevenue - _pCostOfUnitsSold;

double _pcGp = _pcInc / _pcRevenue;
double _sGp = _sInc / _sRevenue;
double _pGp = _pInc / _pRevenue;

double _pcExp = _pcWages + _pcOperatingCosts + _pcTransportationCosts + _pcRents;
double _sExp = _sWages + _sOperatingCosts + _sTransportationCosts + _sRents;
double _pExp = _pWages + _pOperatingCosts + _pTransportationCosts + _pRents;

double _totalGrossSales = _pcGrossSales + _sGrossSales + _pGrossSales;
double _totalRevenue = _pcRevenue + _sRevenue + _pRevenue;
double _totalCostOfGoods = _pcCostOfUnitsSold + _sCostOfUnitsSold + _pCostOfUnitsSold;
double _totalGrossProfit = _pcInc + _sInc + _pInc;
double _totalGp = _totalGrossProfit / _totalRevenue;

double _totalWages = _pcWages + _sWages + _pWages;
double _totalOpCosts = _pcOperatingCosts + _sOperatingCosts + _pOperatingCosts;
double _totalTranspCosts = _pcTransportationCosts + _sTransportationCosts + _pTransportationCosts;
double _totalRents = _pcRents + _sRents + _pRents;
double _totalAe = _pcExp + _sExp + _pExp;
double _result1 = _totalGrossProfit - _totalAe;

double _totalNae = _naeGeneralAdministrative + _naeOther;
double _ebit = _result1 - _totalNae;

double _totalFinancialIncome = _interestIncome - _interestExpenses;
double _npbt = _ebit + _totalFinancialIncome;
double _ros = _ebit / _totalRevenue;
```

Και γράφονται τα δεδομένα του χρήστη στη βάση δεδομένων:

```
sqlCmdInputFields.CommandText = @"INSERT INTO inputFields ([anno],
    [pcUnits], [pcGrossSales], [pcRevenue], [pcCostOfUnitsSold],
    [sUnits], [sGrossSales], [sRevenue], [sCostOfUnitsSold],
    [pUnits], [pGrossSales], [pRevenue], [pCostOfUnitsSold],
    [pcWages], [pcOperatingCosts], [pcTransportationCosts], [pcRents],
    [sWages], [sOperatingCosts], [sTransportationCosts], [sRents],
    [pWages], [pOperatingCosts], [pTransportationCosts], [pRents],
    [naeGeneralAdministrative], [naeOther],
    [interestIncome], [interestExpenses])
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
sqlCmdInputFields.Parameters.AddRange(new OleDbParameter[]{
    new OleDbParameter("@anno", yearToBeWritten),
    new OleDbParameter("@pcUnits", _pcUnits),
    new OleDbParameter("@pcGrossSales", _pcGrossSales),
    new OleDbParameter("@pcRevenue", _pcRevenue),
    new OleDbParameter("@pcCostOfUnitsSold", _pcCostOfUnitsSold),
    new OleDbParameter("@sUnits", _sUnits),
    new OleDbParameter("@sGrossSales", _sGrossSales),
    new OleDbParameter("@sRevenue", _sRevenue),
    new OleDbParameter("@sCostOfUnitsSold", _sCostOfUnitsSold),
    new OleDbParameter("@pUnits", _pUnits),
    new OleDbParameter("@pGrossSales", _pGrossSales),
    new OleDbParameter("@pRevenue", _pRevenue),
    new OleDbParameter("@pCostOfUnitsSold", _pCostOfUnitsSold),
    new OleDbParameter("@pcWages", _pcWages),
    new OleDbParameter("@pcOperatingCosts", _pcOperatingCosts),
    new OleDbParameter("@pcTransportationCosts", _pcTransportationCosts),
    new OleDbParameter("@pcRents", _pcRents),
    new OleDbParameter("@sWages", _sWages),
    new OleDbParameter("@sOperatingCosts", _sOperatingCosts),
    new OleDbParameter("@sTransportationCosts", _sTransportationCosts),
    new OleDbParameter("@sRents", _sRents),
    new OleDbParameter("@pWages", _pWages),
    new OleDbParameter("@pOperatingCosts", _pOperatingCosts),
    new OleDbParameter("@pTransportationCosts", _pTransportationCosts),
    new OleDbParameter("@pRents", _pRents),
    new OleDbParameter("@naeGeneralAdministrative", _naeGeneralAdministrative),
    new OleDbParameter("@naeOther", _naeOther),
    new OleDbParameter("@interestIncome", _interestIncome),
    new OleDbParameter("@interestExpenses", _interestExpenses)
```

Με ένα try { } / catch { } γίνεται handling το exception και εμφανίζει το μήνυμα λάθους και ποιο είναι αυτό αλλά κυρίως ενημερώνει τη μεταβλητή input2DbError που γίνεται true (ενημερώνει τη μεταβλητή ότι κάτι έχει γίνει λάθος).
Ακριβώς η ίδια διαδικασία ακολουθείται και για τον πίνακα OutputFields με τη χρήση της μεταβλητής output2DbError.

Αν είναι όλα σωστά (δεν έχει προκύψει πουθενά σφάλμα)

```
if (!(input2DbError || output2DbError))
{
    button1.Enabled = false;
    Bm.emptyDb = false;
    Bm.latestYear = yearToBeWritten;
    determineYearToBeWritten();
    MessageBox.Show("ΕΠΙΤΥΧΗΣ ΚΑΤΑΧΩΡΗΣΗ ΤΟΥ ΕΤΟΥΣ " + Bm.latestYear + "\n\nΕΑΝ ΕΠΙΘΥΜΕΙΤΕ ΝΕΑ ΚΑΤΑΧΩΡΗΣΗ, ΕΠΙΛΕΞΤ
    textBox7.Text = _totalGrossSales.ToString();
    textBox15.Text = _totalRevenue.ToString();
    textBox11.Text = _totalCostOfGoods.ToString();
    textBox23.Text = _totalGrossProfit.ToString();
    textBox19.Text = _totalGp.ToString();
    textBox21.Text = _pcInc.ToString();
    textBox17.Text = _pcGp.ToString();
    textBox24.Text = _sInc.ToString();
    textBox20.Text = _sGp.ToString();
    textBox22.Text = _pInc.ToString();
    textBox18.Text = _pGp.ToString();
    textBox29.Text = _pcExp.ToString();
    textBox32.Text = _sExp.ToString();
    textBox30.Text = _pExp.ToString();
    textBox47.Text = _totalWages.ToString();
    textBox43.Text = _totalOpCosts.ToString();
    textBox39.Text = _totalTranspCosts.ToString();
    textBox35.Text = _totalRents.ToString();
    textBox31.Text = _totalAe.ToString();
    textBox27.Text = _result1.ToString();
    textBox55.Text = _totalNae.ToString();
    textBox51.Text = _ebit.ToString();
    textBox71.Text = _totalFinancialIncome.ToString();
    textBox67.Text = _npbt.ToString();
    textBox25.Text = _ebit.ToString();
    textBox26.Text = _ros.ToString();
}
```

Γίνεται το κουμπί υπολογισμού (button1) enabled και γράφεται μήνυμα που ενημερώνει το χρήστη για μια πολύ σημαντική λειτουργία: Αφού εισάγει τα δεδομένα και πατήσει υπολογισμό μια φορά, μετά το σύστημα δεν τον αφήνει να πατήσει και δεύτερη το κουμπί χωρίς πρώτα να κάνει καθαρισμό για να αποφευχθεί η διπλοκαταχώρηση δεδομένων εξαιτίας λάθους πατήματος του κουμπιού υπολογισμού για δεύτερη φορά! Ακολούθως, αναθέτονται στα textboxes με τον αντίστοιχο αριθμό οι μεταβλητές στις οποίες έχουν υπολογιστεί τα αποτελέσματα (μπαίνει και η κατάληξη .ToString για να εγγραφεί ως αλφαριθμητικό το νούμερο που προκύπτει).
Αλλιώς, μπαίνει το else είναι 2 τα ενδεχόμενα που μας απασχολούν: Να υπάρχει λάθος μόνο στην μεταβλητή input2DbError και όχι στην output2DbError και το αντίστροφο. Και στις 2 περιπτώσει υπάρχει πρόβλημα γιατί οι πίνακες από σχεδιασμού είναι φτιαγμένοι με συσχετισμό 1-1.

```
else //make the entire operation atomic
{
    if (input2DbError && !(output2DbError))
    {
        OleDbCommand deleteFromOutputFields = new OleDbCommand(@"DELETE * FROM outputFields WHERE anno = @anno", c
        deleteFromOutputFields.Parameters.Add(new OleDbParameter(@"anno", yearToBeWritten));
        try
        {
            conn.Open();
            deleteFromOutputFields.ExecuteNonQuery();
            conn.Close();
        }
        catch (Exception ex)
        {
            if (conn.State == ConnectionState.Open)
            {
                conn.Close();
            }
            MessageBox.Show(ex.Message);
            MessageBox.Show("ΣΦΑΛΜΑ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ.\nΤΕΡΜΑΤΙΣΤΕ ΑΜΕΣΩΣ ΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΔΙΑΓΡΑΨΤΕ ΤΗΝ ΕΓΓΡΑΦΗ
        }
    }
    else if (!(input2DbError) && output2DbError)
    {
        OleDbCommand deleteFromInputFields = new OleDbCommand(@"DELETE * FROM inputFields WHERE anno = @anno", con
        deleteFromInputFields.Parameters.Add(new OleDbParameter(@"anno", yearToBeWritten));
        try
        {
            conn.Open();
            deleteFromInputFields.ExecuteNonQuery();
            conn.Close();
        }
                catch (Exception ex)
                {
                    if (conn.State == ConnectionState.Open)
                    {
                        conn.Close();
                    }
                    MessageBox.Show(ex.Message);
                    MessageBox.Show("ΣΦΑΛΜΑ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ.\nΤΕΡΜΑΤΙΣΤΕ ΑΜΕΣΩΣ ΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΔΙΑΓΡΑΨΤΕ ΤΗΝ ΕΓΓΡΑΦΗ
                }
            }
        }
    }
}
```

Στην περίπτωση που το λάθος είναι μόνο στην input2DbError, σβήνω στον πίνακα output, ενώ το αντίστροφο συμβαίνει στην άλλη περίπτωση. Κάθε φορά αναφέρεται στο χρήστη το τι πρέπει να κάνει.

Στο ROS PreviousYear.cs, γίνεται έλεγχος αν η βάση δεδομένων είναι άδεια, εμφανίζεται το μήνυμα προειδοποίησης

```
1 reference
private void ROS_PreviousYear_Load(object sender, EventArgs e)
{
    OleDbConnection conn;
    OleDbCommand sqlCmd;
    OleDbDataReader dbReader;
    DataTable dTable;
    if(Bm.emptyDb)
    {
        MessageBox.Show("ΔΕΝ ΒΡΕΘΗΚΑΝ ΣΤΟΙΧΕΙΑ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ");
    }
```

και όλος ο κώδικας εκτελείται μέσα στο else:

```
else
{
    conn = new OleDbConnection();
    sqlCmd = new OleDbCommand();
    dTable = new DataTable();
    conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=sales.accdb;Persist Security Info=False;";
    sqlCmd.Connection = conn;
    sqlCmd.CommandText = "SELECT * FROM inputFields INNER JOIN outputFields ON inputFields.anno = outputFields.anno WH
    sqlCmd.Parameters.Add(new OleDbParameter("@yr", (Bm.latestYear - 1)));
    try
    {
        conn.Open();
        dbReader = sqlCmd.ExecuteReader();
        dTable.Load(dbReader);
        conn.Close();
        if (dTable.Rows.Count == 0)
        {
            MessageBox.Show("ΔΕΝ ΒΡΕΘΗΚΑΝ ΟΙΚΟΝΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΓΙΑ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΕΤΟΣ,\n" + (Bm.latestYear - 1).ToStri
        }
        else
        {
            populateFields(dTable);
            updateTitle(dTable);
        }
    }
    catch (Exception ex)
    {
        if (conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
        MessageBox.Show("ΣΦΑΛΜΑ ΚΑΤΑ ΤΗΝ ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ.\n" + ex.Message);
        MessageBox.Show(ex.ToString());
    }
}
```

Για το previous year, δηλαδή το latest-1, φορτώνει τα αποτελέσματα από το reader στο Table, αν δεν βρει κάτι να επιστρλεψει εμφανίζει μήνυμα λάθους, ενώ αν βρει στοιχεία καλεί 2 συναρτήσεις: την populateFields() και την updateTitle(). Παράλληλα, ελέγχει και κάνει handling τα exceptions και εμφανίζει το μήνυμα σφάλματος. Η populateFields() είναι αυτή που εμφανίζει όλα τα αποτελέσματα με μορφή πίνακα:

```
1 reference
private void populateFields(DataTable dt)
{
    textBox3.Text = dt.Rows[0][dt.Columns.IndexOf("pcUnits")].ToString();
    textBox5.Text = dt.Rows[0][dt.Columns.IndexOf("pcGrossSales")].ToString();
    textBox13.Text = dt.Rows[0][dt.Columns.IndexOf("pcRevenue")].ToString();
    textBox9.Text = dt.Rows[0][dt.Columns.IndexOf("pcCostOfUnitsSold")].ToString();
    textBox21.Text = dt.Rows[0][dt.Columns.IndexOf("pcInc")].ToString();
    textBox17.Text = dt.Rows[0][dt.Columns.IndexOf("pcGp")].ToString();
    textBox45.Text = dt.Rows[0][dt.Columns.IndexOf("pcWages")].ToString();
    textBox41.Text = dt.Rows[0][dt.Columns.IndexOf("pcOperatingCosts")].ToString();
    textBox37.Text = dt.Rows[0][dt.Columns.IndexOf("pcTransportationCosts")].ToString();
    textBox33.Text = dt.Rows[0][dt.Columns.IndexOf("pcRents")].ToString();
    textBox29.Text = dt.Rows[0][dt.Columns.IndexOf("pcExp")].ToString();
    textBox4.Text = dt.Rows[0][dt.Columns.IndexOf("sUnits")].ToString();
    textBox8.Text = dt.Rows[0][dt.Columns.IndexOf("sGrossSales")].ToString();
    textBox16.Text = dt.Rows[0][dt.Columns.IndexOf("sRevenue")].ToString();
    textBox12.Text = dt.Rows[0][dt.Columns.IndexOf("sCostOfUnitsSold")].ToString();
    textBox24.Text = dt.Rows[0][dt.Columns.IndexOf("sInc")].ToString();
    textBox20.Text = dt.Rows[0][dt.Columns.IndexOf("sGp")].ToString();
    textBox48.Text = dt.Rows[0][dt.Columns.IndexOf("sWages")].ToString();
    textBox44.Text = dt.Rows[0][dt.Columns.IndexOf("sOperatingCosts")].ToString();
    textBox40.Text = dt.Rows[0][dt.Columns.IndexOf("sTransportationCosts")].ToString();
    textBox36.Text = dt.Rows[0][dt.Columns.IndexOf("sRents")].ToString();
    textBox32.Text = dt.Rows[0][dt.Columns.IndexOf("sExp")].ToString();
    textBox2.Text = dt.Rows[0][dt.Columns.IndexOf("pUnits")].ToString();
    textBox6.Text = dt.Rows[0][dt.Columns.IndexOf("pGrossSales")].ToString();
    textBox14.Text = dt.Rows[0][dt.Columns.IndexOf("pRevenue")].ToString();
    textBox10.Text = dt.Rows[0][dt.Columns.IndexOf("pCostOfUnitsSold")].ToString();
    textBox22.Text = dt.Rows[0][dt.Columns.IndexOf("pInc")].ToString();
    textBox18.Text = dt.Rows[0][dt.Columns.IndexOf("pGp")].ToString();
    textBox46.Text = dt.Rows[0][dt.Columns.IndexOf("pWages")].ToString();
    textBox42.Text = dt.Rows[0][dt.Columns.IndexOf("pOperatingCosts")].ToString();
    textBox38.Text = dt.Rows[0][dt.Columns.IndexOf("pTransportationCosts")].ToString();
    textBox34.Text = dt.Rows[0][dt.Columns.IndexOf("pRents")].ToString();
    textBox30.Text = dt.Rows[0][dt.Columns.IndexOf("pExp")].ToString();
    textBox7.Text = dt.Rows[0][dt.Columns.IndexOf("totalGrossSales")].ToString();
    textBox15.Text = dt.Rows[0][dt.Columns.IndexOf("totalRevenue")].ToString();
    textBox11.Text = dt.Rows[0][dt.Columns.IndexOf("totalCostOfGoods")].ToString();
    textBox23.Text = dt.Rows[0][dt.Columns.IndexOf("totalGrossProfit")].ToString();
    textBox19.Text = dt.Rows[0][dt.Columns.IndexOf("totalGp")].ToString();
    textBox47.Text = dt.Rows[0][dt.Columns.IndexOf("totalWages")].ToString();
    textBox43.Text = dt.Rows[0][dt.Columns.IndexOf("totalOpCosts")].ToString();
    textBox39.Text = dt.Rows[0][dt.Columns.IndexOf("totalTranspCosts")].ToString();
    textBox35.Text = dt.Rows[0][dt.Columns.IndexOf("totalRents")].ToString();
    textBox31.Text = dt.Rows[0][dt.Columns.IndexOf("totalAe")].ToString();
    textBox27.Text = dt.Rows[0][dt.Columns.IndexOf("result1")].ToString();
    textBox63.Text = dt.Rows[0][dt.Columns.IndexOf("naeGeneralAdministrative")].ToString();
    textBox59.Text = dt.Rows[0][dt.Columns.IndexOf("naeOther")].ToString();
    textBox55.Text = dt.Rows[0][dt.Columns.IndexOf("totalNae")].ToString();
    textBox51.Text = dt.Rows[0][dt.Columns.IndexOf("ebit")].ToString();
    textBox79.Text = dt.Rows[0][dt.Columns.IndexOf("interestIncome")].ToString();
    textBox75.Text = dt.Rows[0][dt.Columns.IndexOf("interestExpenses")].ToString();
    textBox71.Text = dt.Rows[0][dt.Columns.IndexOf("totalFinancialIncome")].ToString();
    textBox67.Text = dt.Rows[0][dt.Columns.IndexOf("npbt")].ToString();
    textBox25.Text = dt.Rows[0][dt.Columns.IndexOf("ebit")].ToString();
    textBox26.Text = dt.Rows[0][dt.Columns.IndexOf("ros")].ToString();
```

ενώ η updateTitle(DataTable dt) παίρνει σαν όρισμα τον παραπάνω πίνακα

```
1 reference
private void updateTitle(DataTable dt)
{
    label1.Text = "ROS Προηγούμενου Έτους" + " (" + dt.Rows[0][dt.Columns.IndexOf("inputFields.anno")].ToString() + ")";
}
```

Και βάζει στον τίτλο του label1 το χρόνο που βρήκε σαν αριθμό, γι' αυτό χρησιμοποιείται ".ToString()".

Τέλος, στο Comparisons.cs, με το που τρέχει το from load, δίνονται οι πληροφορίες για τη σύνδεση με τη βάση με χρήση 2 μεταβλητών για σφάλματα ανάγνωσης: την previousYearError και την latestYearError και τις θέτω ίσε με false.

```
private void Comparisons_Load(object sender, EventArgs e)
{
    OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=sales.accdb;Persist Securit
    OleDbCommand sqlCmdPrevious, sqlCmdLatest;
    OleDbDataReader dbReaderPrevious, dbReaderLatest;
    DataTable dTablePrevious, dTableLatest;
    bool previousYearError = false;
    bool latestYearError = false;
```

Αν η βάση είναι άδεια, βγαίνει μήνυμα λάθους

```
if (Bm.emptyDb)
{
    MessageBox.Show("ΔΕΝ ΒΡΕΘΗΚΑΝ ΣΤΟΙΧΕΙΑ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ");
}
```

ενώ στο else (δηλαδή αν η βάση δεν είναι άδεια), τρέχει όλος ο κώδικας, πρώτα για PreviousYear

```
else
{
    sqlCmdPrevious = new OleDbCommand();
    sqlCmdPrevious.Connection = conn;
    sqlCmdPrevious.CommandText = "SELECT [totalGrossSales], [totalRevenue], [totalCostOfGoods], [totalGrossProfit], [to
        "[totalWages], [totalOpCosts], [totalTranspCosts], [totalRents], [totalAe], [result1], [naeGeneralAdministrativ
        "[interestIncome], [interestExpenses], [totalFinancialIncome], [npbt], [ebit], [ros] " +
        "FROM inputFields INNER JOIN outputFields ON inputFields.anno = outputFields.anno WHERE inputFields.anno = @yr"
    sqlCmdPrevious.Parameters.Add(new OleDbParameter("@yr", (Bm.latestYear - 1)));
    dTablePrevious = new DataTable();
    try
    {
        conn.Open();
        dbReaderPrevious = sqlCmdPrevious.ExecuteReader();
        dTablePrevious.Load(dbReaderPrevious);
        conn.Close();
        if (dTablePrevious.Rows.Count == 0)
        {
            MessageBox.Show("ΔΕΝ ΒΡΕΘΗΚΑΝ ΟΙΚΟΝΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΓΙΑ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΕΤΟΣ,\n" + (Bm.latestYear - 1).ToStrin
            previousYearError = true;
        }
    }
    catch (Exception ex)
    {
        if (conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
        MessageBox.Show("ΣΦΑΛΜΑ ΚΑΤΑ ΤΗΝ ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΠΡΟΗΓΟΥΜΕΝΟΥ ΕΤΟΥΣ.\n" + ex.Message);
        MessageBox.Show(ex.ToString());
        previousYearError = true;
    }
```

και μετά για το LatestYear:

```
sqlCmdLatest = new OleDbCommand();
sqlCmdLatest.Connection = conn;
sqlCmdLatest.CommandText = "SELECT [totalGrossSales], [totalRevenue], [totalCostOfGoods], [totalGrossProfit], [tota
    "[totalWages], [totalOpCosts], [totalTranspCosts], [totalRents], [totalAe], [result1], [naeGeneralAdministrativ
    "[interestIncome], [interestExpenses], [totalFinancialIncome], [npbt], [ebit], [ros] " +
    "FROM inputFields INNER JOIN outputFields ON inputFields.anno = outputFields.anno WHERE inputFields.anno = @yr"
sqlCmdLatest.Parameters.Add(new OleDbParameter("@yr", Bm.latestYear));
dTableLatest = new DataTable();
try
{
    conn.Open();
    dbReaderLatest = sqlCmdLatest.ExecuteReader();
    dTableLatest.Load(dbReaderLatest);
    conn.Close();
    if (dTableLatest.Rows.Count == 0)
    {
        MessageBox.Show("ΔΕΝ ΒΡΕΘΗΚΑΝ ΟΙΚΟΝΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΓΙΑ ΤΟ ΠΙΟ ΠΡΟΣΦΑΤΟ ΕΤΟΣ,\n" + Bm.latestYear.ToString() +
        latestYearError = true;
    }
}
catch (Exception ex)
{
    if (conn.State == ConnectionState.Open)
    {
        conn.Close();
    }
    MessageBox.Show("ΣΦΑΛΜΑ ΚΑΤΑ ΤΗΝ ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΤΡΕΧΟΝΤΟΣ ΕΤΟΥΣ.\n" + ex.Message);
    MessageBox.Show(ex.ToString());
    latestYearError = true;
}
```

Ακολουθεί έλεγχος, για PreviousYear αν και τα 2 είναι false, δηλαδή αν όλα έχουν πάει καλά

```
if (!(previousYearError || latestYearError))
{
    double previousGrossSales = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalGross
    double previousRevenue = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalRevenue"
    double previousCostOfGoods = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalCost
    double previousGrossProfit = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalGros
    double previousGp = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalGp")]);
    double previousWages = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalWages")]);
    double previousOpCosts = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalOpCosts"
    double previousTranspCosts = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalTran
    double previousRents = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalRents")]);
    double previousAe = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalAe")]);
    double previousResult1 = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("result1")]);
    double previousGenAdmin = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("naeGeneralAd
    double previousOtherNae = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("naeOther")])
    double previousNae = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalNae")]);
    double previousFinIncome = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("interestInc
    double previousFinExpenses = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("interestE
    double previousNetFinIncome = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("totalFin
    double previousNpbt = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("npbt")]);
    double previousEbit = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("ebit")]);
    double previousRos = Convert.ToDouble(dTablePrevious.Rows[0][dTablePrevious.Columns.IndexOf("ros")]);

    textBox30.Text = previousGrossSales.ToString();
    textBox29.Text = previousRevenue.ToString();
    textBox28.Text = previousCostOfGoods.ToString();
    textBox24.Text = previousGrossProfit.ToString();
    textBox22.Text = previousGp.ToString();
    textBox21.Text = previousWages.ToString();
    textBox20.Text = previousOpCosts.ToString();
    textBox18.Text = previousTranspCosts.ToString();
    textBox17.Text = previousRents.ToString();
    textBox16.Text = previousAe.ToString();
textBox14.Text = previousResult1.ToString();
textBox13.Text = previousGenAdmin.ToString();
textBox12.Text = previousOtherNae.ToString();
textBox10.Text = previousNae.ToString();
textBox9.Text = previousEbit.ToString();
textBox8.Text = previousFinIncome.ToString();
textBox6.Text = previousFinExpenses.ToString();
textBox5.Text = previousNetFinIncome.ToString();
textBox4.Text = previousNpbt.ToString();
textBox3.Text = previousEbit.ToString();
textBox2.Text = previousRos.ToString();
```

Και ενημερώνονται τα textboxes από τις μεταβλητές βάση των αντιστοιχιών που δίνονται από το παρακάτω screenshot για την αρίθμηση τους:

Αξίζει να αναφερθεί, ότι μέχρι το σημείο αυτό, δεν γίνεται κανένα calculation ακόμα, απλά ενημερώνονται τα textboxes. Με datatable ήταν αδύνατον να γίνουν calculations που είναι πλέον εφικτά με τη χρήση double μεταβλητών. Ακολούθως, η ίδια λογική χρησιμοποιείται και για το latest year (Ytd):

```
double latestGrossSales = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalGrossSales"
double latestRevenue = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalRevenue")]);
double latestCostOfGoods = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalCostOfGood
double latestGrossProfit = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalGrossProfi
double latestGp = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalGp")]);
double latestWages = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalWages")]);
double latestOpCosts = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalOpCosts")]);
double latestTranspCosts = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalTranspCost
double latestRents = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalRents")]);
double latestAe = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalAe")]);
double latestResult1 = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("result1")]);
double latestGenAdmin = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("naeGeneralAdminist
double latestOtherNae = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("naeOther")]);
double latestNae = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalNae")]);
double latestFinIncome = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("interestIncome")]
double latestFinExpenses = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("interestExpense
double latestNetFinIncome = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("totalFinancial
double latestNpbt = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("npbt")]);
double latestEbit = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("ebit")]);
double latestRos = Convert.ToDouble(dTableLatest.Rows[0][dTableLatest.Columns.IndexOf("ros")]);

textBox7.Text = latestGrossSales.ToString();
textBox15.Text = latestRevenue.ToString();
textBox11.Text = latestCostOfGoods.ToString();
textBox23.Text = latestGrossProfit.ToString();
textBox19.Text = latestGp.ToString();
textBox47.Text = latestWages.ToString();
textBox43.Text = latestOpCosts.ToString();
textBox39.Text = latestTranspCosts.ToString();
textBox35.Text = latestRents.ToString();
textBox31.Text = latestAe.ToString();
textBox27.Text = latestResult1.ToString();
textBox63.Text = latestGenAdmin.ToString();
textBox59.Text = latestOtherNae.ToString();
textBox55.Text = latestNae.ToString();
textBox51.Text = latestEbit.ToString();
textBox79.Text = latestFinIncome.ToString();
textBox75.Text = latestFinExpenses.ToString();
textBox71.Text = latestNetFinIncome.ToString();
textBox67.Text = latestNpbt.ToString();
textBox25.Text = latestEbit.ToString();
textBox26.Text = latestRos.ToString();
```

Με την ίδια λογική ενημερώνω πάλι τα textboxes από τις μεταβλητές, απλά γίνεται η διαδικασία για το τρέχον έτος.

Στη συνέχεια ακολουθούν οι υπολογισμοί:

```
textBox60.Text = ((latestGrossSales - previousGrossSales) / previousGrossSales).ToString();
textBox58.Text = ((latestRevenue - previousRevenue) / previousRevenue).ToString();
textBox57.Text = ((latestCostOfGoods - previousCostOfGoods) / previousCostOfGoods).ToString();
textBox56.Text = ((latestGrossProfit - previousGrossProfit) / previousGrossProfit).ToString();
textBox53.Text = ((latestWages - previousWages) / previousWages).ToString();
textBox52.Text = ((latestOpCosts - previousOpCosts) / previousOpCosts).ToString();
textBox50.Text = ((latestTranspCosts - previousTranspCosts) / previousTranspCosts).ToString();
textBox49.Text = ((latestRents - previousRents) / previousRents).ToString();
textBox48.Text = ((latestAe - previousAe) / previousAe).ToString();
textBox33.Text = ((latestResult1 - previousResult1) / previousResult1).ToString();
textBox45.Text = ((latestGenAdmin - previousGenAdmin) / previousGenAdmin).ToString();
textBox44.Text = ((latestOtherNae - previousOtherNae) / previousOtherNae).ToString();
textBox42.Text = ((latestNae - previousNae) / previousNae).ToString();
textBox41.Text = ((latestEbit - previousEbit) / previousEbit).ToString();
textBox40.Text = ((latestFinIncome - previousFinIncome) / previousFinIncome).ToString();
textBox38.Text = ((latestFinExpenses - previousFinExpenses) / previousFinExpenses).ToString();
textBox37.Text = ((latestNetFinIncome - previousNetFinIncome) / previousNetFinIncome).ToString();
textBox36.Text = ((latestNpbt - previousNpbt) / previousNpbt).ToString();
textBox34.Text = ((latestEbit - previousEbit) / previousEbit).ToString();
```

Οι υπολογισμοί γίνονται κατευθείαν και αναθέτονται στα αντίστοιχα textboxes, με τη γνωστή αντιστοιχία από το screenshot του περιβάλλοντος αλληλεπίδρασης του χρήστη.

```
label10.Text = (Bm.latestYear - 1).ToString();
label2.Text = Bm.latestYear.ToString();
```

Τέλος, ενημερώνονται οι επικεφαλίδες των 2 στηλών, με label10 για το προηγούμενο και label12 για το τρέχον.

Συμπεράσματα

Το πολύπλοκο οικονομικό περιβάλλον, οι συνεχιζόμενες μεταβολές του καθώς επίσης και οι συνέπειες αυτών στα οικονομικά μέλη αλλά και στις οικονομικές οντότητες που επηρεάζονται με τη σειρά τους, καθιστούν την ανάγκη ανάλυσης, προσαρμογής αλλά και εξαγωγής χρήσιμων και υλοποιήσιμων συμπερασμάτων εξαιρετικά άμεση και ιδιάζουσα. Κυρίως, αν αναλογιστεί κανείς την ανάγκη για δημιουργία διαδικασιών κοινές για όλους με ομοιογένεια στη χρήση, αλλά και στην δημιουργία αποτελεσμάτων προκειμένου να εξοικονομείται πολύτιμος χρόνος και να επιτυγχάνεται το πολυπόθητο ανταγωνιστικό πλεονέκτημα, σε όρους ποιότητας, ανθρώπινου κεφαλαίου, παραγωγικότητας και οικονομικής βιωσιμότητας / ευμάρειας. Η χρήση της πληροφορικής παρέχοντας ένα περιβάλλον ασφαλές, ομοιογενές και φιλικό επισπεύδει τα στάδια χρηματοοικονομικής ανάλυσης και αξιολόγησης και συντελεί στην αύξηση της παραγωγικότητας αλλά και στην γρήγορη αναγνώριση (early warning) προβληματικών καταστάσεων που καλείται να αντιμετωπίσει ο εκάστοτε επιχειρηματίας και το top management που καλείται να ασκήσει ενεργητική διοίκηση (hands on approach). Στα πλαίσια αυτά κινείται η παρούσα εφαρμογή, που αναλύθηκε εκτενώς στη μελέτη αυτή, και που ως στόχο εκτός όλων των υπολοίπων έχει τη δημιουργία ενός περιβάλλοντος χρήσης που να προκαλεί το ενδιαφέρον των χρηστών, δημιουργώντας μια συνολική εμπειρία μέσω ηχητικών τμημάτων και θεματικών εικόνων, αλλά παράλληλα δίνει μέσα σε ένα μόνο μικρό πλαίσιο όλη την πληροφορία που χρειάζεται η ανώτατη διοίκηση για να προχωρήσει στις απαιτούμενες ενέργειες (corrective-preventive actions).

Βιβλιογραφία

1. Key Performance Indicators: The 75 Measures Every Manager Needs to Know, Bernard Marr, Pearson 2013, Financial Times Publishing

2. Key Performance Indicators, Developing Implementing and Using Winning KPIs, 2nd edition, David Parmenter, 2010, Wiley Publishing

3. Transforming Performance Measurement, Dean R. Spitzer, 2007, Amacom Publishing

4. Designing Metrics: Crafting balanced Measures for Managing Performance, Bob Frost, 2007, Measurement International Publishing

5. Conducting Assessments: Evaluating Programs, Facilities, Organizations, Bob Frost, 2007, Measurement International Publishing

Παράρτημα, παράθεση αρχείων