Τ.Ε.Ι. ΗΠΕΙΡΟΥ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΤΗΛΕΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΓΙΑ ΤΙΣ ΑΝΑΓΚΕΣ ΤΩΝ ΛΟΓΙΣΤΗΡΙΩΝ ΤΩΝ ΤΡΑΠΕΖΩΝ

Του σπουδαστή
ΓΡΙΒΑ ΑΡΓΥΡΗ

Επιβλέπων καθηγητής
ΚΟΝΕΤΑΣ ΔΗΜΗΤΡΙΟΣ

ΑΡΤΑ,25 ΙΟΥΝΙΟΥ 2003

Πτυχιακή εργασία μέρος των απαιτήσεων του Τμήματος Τηλεπληροφορικής και Διοίκησης

ΕΥΧΑΡΙΣΤΙΕΣ

Στην εκπόνηση της παρούσας Πτυχιακής Εργασίας συνέβαλε τα μέγιστα ο επιβλέπων καθηγητής μου κ. Δημήτριος Κονετάς. Νιώθω βαθύτατη την ανάγκη να τον ευχαριστήσω τόσο για την υπόδειξη του θέματος όσο και για την καθοδήγηση σε όλα τα στάδια εκπόνησης της εργασίας μου.

Παράλληλα αισθάνομαι βαθιά υποχρεωμένος να ευχαριστήσω τον κ. Αβραάμ Ναυρόζογλου που συνέβαλε τα μέγιστα παρέχοντας την απαιτούμενη γνώση για το χώρο των τραπεζών. Τον ευχαριστώ επίσης για τις σημαντικότατες παρατηρήσεις, τα εύστοχα σχόλια και την καθοριστική συμμετοχή του στην επικύρωση του λογισμικού αλλά και την συνολική εκτίμηση του αποτελέσματος.

Abstract

Αντικείμενο της παρούσας μελέτης, που γίνεται στα πλαίσια πτυχιακής εργασίας, είναι η ανάπτυξη ενός μέρους από το πληροφοριακό σύστημα μιας τράπεζας και συγκεκριμένα του υποσυστήματος του λογιστηρίου. Ο στόχος της συγκεκριμένης προσπάθειας δεν επικεντρώνεται μόνο στη παράδοση ενός πλήρους λειτουργικού συστήματος λογισμικού, αλλά και στη παρακολούθηση ολόκληρης της διαδικασίας ανάπτυξης. Με αφορμή την ανάπτυξη του συστήματος θα παρουσιαστούν ορισμένες από τις πλέον σύγχρονες τεχνολογίες και προσεγγίσεις για όλα τα στάδια της ανάπτυξης λογισμικού. Σημαντική βοήθεια στη προσπάθεια ανάπτυξης παρέχει ένα ήδη υπάρχον σύστημα, που αν και είναι τεχνολογικά ξεπερασμένο, αποδεδειγμένα καλύπτει τις ανάγκες των τραπεζών σήμερα. Στόχος είναι να χρησιμοποιηθεί το αποτέλεσμα της προσπάθειας αυτής για την κάλυψη των αναγκών του λογιστηρίου στα πλαίσια ολόκληρου του τραπεζικού συστήματος. Το γεγονός αυτό σημαίνει, ότι αυτό που θα δημιουργηθεί θα πρέπει σε κάθε περίπτωση να πληρεί το επίπεδο ποιότητας που θεωρείται εμπορικά αποδεκτό για τέτοιου είδους προϊόντα λογισμικού. Επίσης, σημαντικό κριτήριο για την επιτυχία του εγχειρήματος είναι η δυνατότητα ενσωμάτωσης του στο ευρύτερο πλαίσιο του τραπεζικού πληροφοριακού συστήματος, κάνοντας έτσι δυνατή και την εμπορική εκμετάλλευσή του.

Λέξεις κλειδιά:
Τραπεζικά πληροφοριακά συστήματα
Διαδικασία ανάπτυξης λογισμικού

ΚΕΦΑΛΑΙΟ 1ο
ΕΙΣΑΓΩΓΗ

Ένας από τους χώρους που η επιστήμη της πληροφορικής έχει τη μεγαλύτερη συνεισφορά είναι αναμφισβήτητα οι τραπεζικές συναλλαγές. Ο όγκος των πληροφοριών που θα πρέπει να αποθηκευθούν αλλά και να επεξεργασθούν ήταν ο βασικός λόγος που ο χώρος αυτός εκδήλωσε από πολύ νωρίς το ενδιαφέρον του για τη πληροφορική και τις υπηρεσίες που μπορεί να προσφέρει. Ξεκινώντας από απλά συστήματα που σαν στόχο είχαν απλά την αυτοματοποίηση κάποιων διαδικασιών και διευρύνοντας συνεχώς τις διαδικασίες αυτές, φτάσαμε σήμερα σε συστήματα πλήρους μηχανογράφησης σε όλα τα επίπεδα της οργάνωσης και λειτουργίας μιας τράπεζας. Στις μέρες μας, δεν είναι πια υπερβολή, ότι το σύνολο των τραπεζικών συναλλαγών στηρίζεται σε μηχανογραφημένα πληροφοριακά συστήματα.

Λαμβάνοντας λοιπόν υπόψη τη σημασία των πληροφοριακών συστημάτων για τις τραπεζικές συναλλαγές γίνεται αντιληπτό και το ενδιαφέρον που παρουσιάζει η εξέλιξη και η ανάπτυξη τέτοιων συστημάτων για το χώρο της πληροφορικής. Τα μεγάλα ποσά πληροφορίας, οι απαιτήσεις για επεξεργασία σε πραγματικό χρόνο και οι αυξημένες ανάγκες σε αξιοπιστία και ασφάλεια κάνουν την ανάπτυξη τέτοιων συστημάτων ιδιαίτερα δύσκολη και απαιτητική διαδικασία. Δεν είναι τυχαίο ότι πολλές από τις πλέον σύγχρονες τεχνολογίες για την ανάπτυξη συστημάτων λογισμικού χρησιμοποιούνται αρχικά σε τέτοιου είδους συστήματα προκειμένου να καλυφθούν οι συνεχώς αυξανόμενες ανάγκες του χώρου αυτού. Η ανάπτυξη ενός τέτοιου συστήματος σε συνδυασμό με όλη τη διαδικασία της ανάπτυξης θα είναι σε γενικές γραμμές και το κεντρικό αντικείμενο της μελέτης.

Είναι μάλλον προφανές ότι ένα σύγχρονο σύστημα μηχανογράφησης τραπεζών καλείται να καλύψει πολλές διαφορετικές λειτουργίες εντελώς διαφορετικών ομάδων χρηστών. Τα γεγονός αυτό επιβάλλει και το χωρισμό ενός τέτοιου συστήματος σε υποσυστήματα, που συνήθως προορίζονται για να καλύψουν τις ανάγκες ενός συγκεκριμένου τμήματος ή ομάδας χρηστών. Επιπλέον, ο χωρισμός αυτός παρουσιάζει σαφώς μικρότερη δυσκολία τόσο στην ανάπτυξη όσο και στη συντήρηση, συγκρινόμενος με τη προσπάθεια ανάπτυξης του συστήματος σαν σύνολο όλων των απαιτήσεων. Στην ανάπτυξη ενός υποσυστήματος και συγκεκριμένα του υποσυστήματος του λογιστηρίου θα επικεντρώσουμε τη προσπάθεια με στόχο τη δημιουργία ενός μικρού αλλά πραγματικά λειτουργικού

κομματιού του συστήματος. Το μέρος του συστήματος που θα αναπτυχθεί θα πρέπει να ανταποκρίνεται τόσο στις πραγματικές απαιτήσεις του τραπεζικού χώρου όσο και στα επίπεδα ποιότητας που θέτει η αγορά λογισμικού για συστήματα τέτοιας κλίμακας και σημασίας.

Η βάση αλλά και η κύρια πηγή άντλησης πληροφοριών στην προσπάθεια αυτή είναι ένα ήδη υπάρχον σύστημα που καλύπτει αυτή τη στιγμή τις συγκεκριμένες ανάγκες σε αρκετές συνεταιριστικές τράπεζες στην Ελλάδα. Η αξία της ύπαρξης του συστήματος αυτού για την επιτυχία του επιτεύγματος γίνεται εύκολα αντιληπτή αν αναλογιστεί κανείς ότι περιλαμβάνει την εμπειρία και τη γνώση των πραγματικών αναγκών του περιβάλλοντος εργασίας, όπως διαμορφώθηκε μετά από αρκετά χρόνια εξέλιξης και χρήσης. Βέβαια είναι γεγονός, ότι το συγκεκριμένο πακέτο λογισμικού δεν μπορεί σε καμία περίπτωση να παρέχει καθοδήγηση στην επιλογή τεχνολογιών και γενικότερα στο τρόπο εξέλιξης του νέου συστήματος καθώς οι τεχνολογίες που χρησιμοποιεί είναι ξεπερασμένες. Το γεγονός αυτό, αποτελεί και το βασικό λόγο αντικατάστασής του, χωρίς όμως να έχει επιπτώσεις στη σημαντική συνεισφορά του όσον αφορά στη σωστή κατανόηση του προβλήματος και την δημιουργία ενός συστήματος που θα απαντά στα πραγματικά προβλήματα των χρηστών στους οποίους απευθύνεται.

Αυτό που αξίζει τέλος να τονιστεί είναι ότι εκτός από τη δημιουργία ενός τελικού προϊόντος μεγάλο ενδιαφέρον παρουσιάζει και η όλη διαδικασία ανάπτυξης πακέτων λογισμικού. Πολλές από τις πλέον σύγχρονες προσεγγίσεις για κάθε στάδιο της ανάπτυξης λογισμικού είναι σε θέση να συνεισφέρουν στη προσπάθεια ανάπτυξης. Για το λόγο αυτό, η μελέτη και σύγκριση όλων των τάσεων που υπάρχουν σήμερα κρίνεται ως ένα ιδιαίτερα σημαντικό κομμάτι της προσπάθειας. Μετά από σύγκριση των πλεονεκτημάτων και μειονεκτημάτων της κάθε μίας από αυτές, λαμβάνοντας πάντα υπόψη τους στόχους και τις ιδιαίτερες απαιτήσεις του συγκεκριμένου εγχειρήματος, θα επιλέγεται η προσέγγιση ή η τεχνολογία που εξυπηρετεί καλύτερα τους στόχους σε κάθε περίπτωση. Με τον τρόπο αυτό και προχωρώντας με μεγάλη προσοχή σε κάθε βήμα είναι δυνατή και η αποδοτική χρησιμοποίηση των σύγχρονων μεθόδων ανάπτυξης, αλλά και η επίτευξη όλων των στόχων μέσα στα δεδομένα χρονικά πλαίσια.

ΚΕΦΑΛΑΙΟ 2ο
ΕΠΙΛΟΓΗ ΜΟΝΤΕΛΟΥ ΑΝΑΠΤΥΞΗΣ

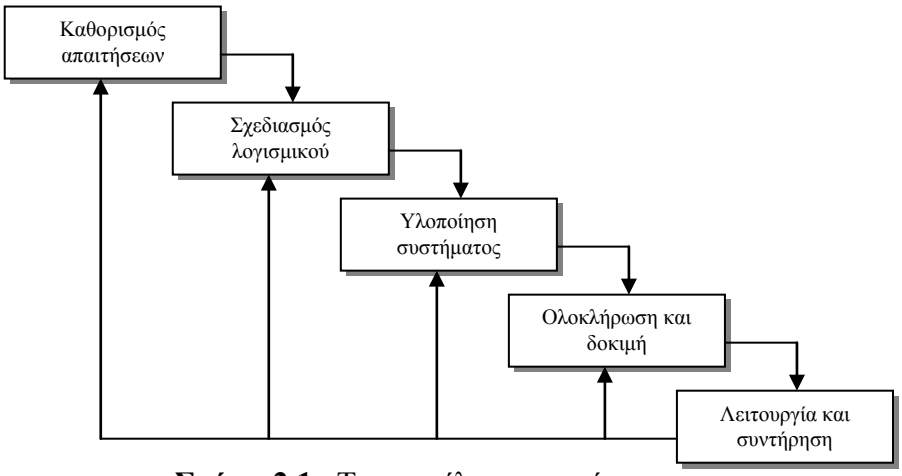
2.1. Διαδικασία ανάπτυξης λογισμικού

Η ανάπτυξη λογισμικού είναι μια συστηματοποιημένη εργασία που μπορεί να περιγραφεί από ένα μοντέλο. Το μοντέλο είναι μια γενική αναπαράσταση της διαδικασίας ανάπτυξης λογισμικού και εξετάζει την ανάπτυξη από μια συγκεκριμένη οπτική γωνία παρέχοντας μερικές μόνο πληροφορίες για αυτή. Τα μοντέλα αυτά είναι περισσότερο αφηρημένες περιγραφές των προσεγγίσεων για την ανάπτυξη λογισμικού παρά αυστηροί ορισμοί για τις διαδικασίες ανάπτυξης. Στη πραγματικότητα και σε εφαρμογές μεγάλου μεγέθους δεν χρησιμοποιείται ποτέ ένα μόνο μοντέλο ανάπτυξης αλλά διαφορετικά μοντέλα χρησιμοποιούνται για διαφορετικά κομμάτια της εφαρμογής. Στη συνέχεια παρουσιάζονται τα τέσσερα βασικά μοντέλα για τη διαδικασία ανάπτυξης λογισμικού [1].

Το πρώτο από τα μοντέλα είναι το μοντέλο «καταρράκτη» (*waterfall model*) που φαίνεται στο σχήμα 2.1. Σύμφωνα με το μοντέλο αυτό οι βασικές δραστηριότητες της ανάπτυξης λογισμικού όπως ο καθορισμός των απαιτήσεων, η υλοποίηση και η δοκιμή και επικύρωση αποτελούν ξεχωριστές φάσεις της διαδικασίας. Τα στάδια αυτά της διαδικασίας σύμφωνα με το μοντέλο είναι [1]:

- η ανάλυση και ο ορισμός των απαιτήσεων,
- η σχεδίαση του συστήματος,
- η υλοποίηση της σχεδίασης και η δοκιμή των επιμέρους μονάδων,
- η εγκατάσταση και δοκιμή του συστήματος συνολικά και
- η λειτουργία και συντήρηση του

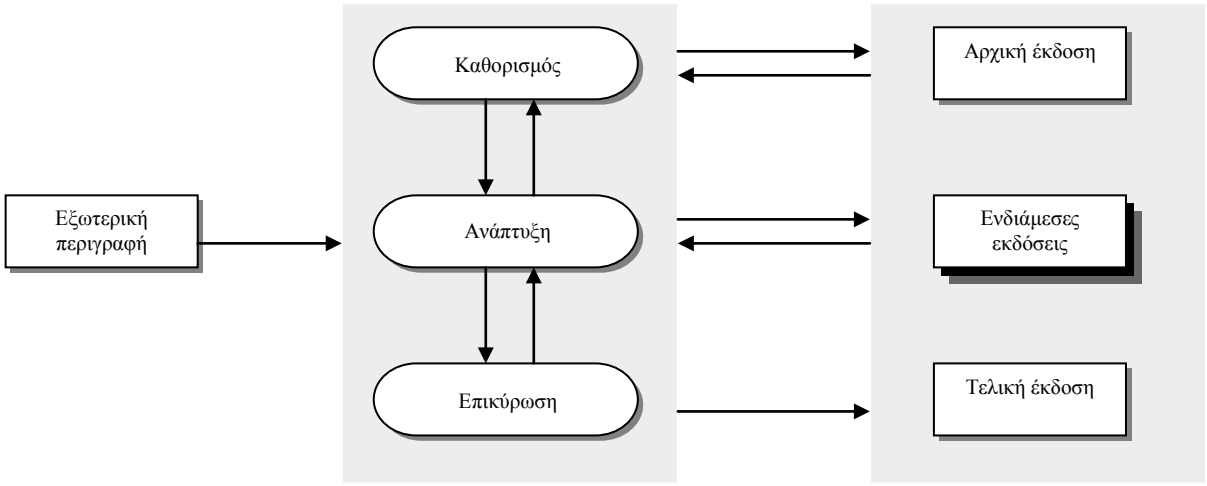
Τα αποτελέσματα κάθε σταδίου που αναπαρίστανται από κάποια τεκμηρίωση (ένα ή περισσότερα έγγραφα) αποτελούν το υλικό που θα τροφοδοτήσει την επόμενη φάση της ανάπτυξης. Η επόμενη φάση δεν ξεκινάει κατά κανόνα πριν ολοκληρωθεί η προηγούμενη και τεκμηριωθούν τα αποτελέσματά της. Στο μοντέλο αυτό οι επαναλήψεις που απαιτούνται για να καλυφθούν απαιτήσεις που ανακαλύφθηκαν ή τροποποιήθηκαν στη πορεία της ανάπτυξης είναι αρκετά δαπανηρή λόγω του αυξημένου αριθμού των εγγράφων που πρέπει να δημιουργηθούν και να εγκριθούν σε κάθε στάδιο.



Σχήμα 2.1 Το μοντέλο καταρράκτη

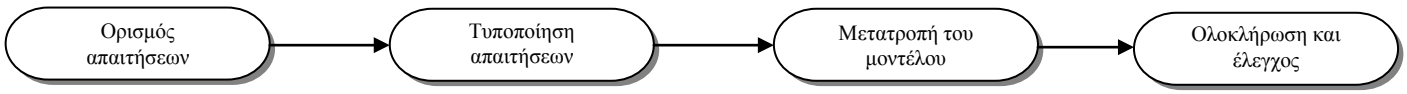
Η εξελικτική ανάπτυξη (*evolutionary development*) είναι μια προσέγγιση όπου οι δραστηριότητες του καθορισμού των απαιτήσεων, της ανάπτυξης και της επικύρωσης επικαλύπτονται. Μια αρχική έκδοση του συστήματος υλοποιείται άμεσα με βάση κάποιες αρχικές απαιτήσεις. Στη συνέχεια με τη βοήθεια του πελάτη το σύστημα συνεχώς βελτιώνεται ώστε να φτάσει να ικανοποιεί τις πραγματικές απαιτήσεις. Υπάρχουν δύο βασικοί τύποι εξελικτικής ανάπτυξης [1]:

- η εξερευνητική ανάπτυξη (*exploratory development*) όπου η διαδικασία γίνεται σε συνεργασία με το πελάτη σε μια προσπάθεια εντοπισμού των πραγματικών απαιτήσεων προσθέτοντας τη λειτουργικότητα που προτείνει ο χρήστης σε κάποια αρχική έκδοση και
- η προτυποποίηση απόρριψης (*throw-away prototyping*) όπου γίνεται προσέγγιση των πραγματικών απαιτήσεων του πελάτη μέσω πειραματισμών στα κομμάτια εκείνα που χρειάζονται βελτίωση.



Σχήμα 2.2 Το εξελικτικό μοντέλο

Η ανάπτυξη τυποποιημένων συστημάτων (*formal systems development*) βασίζεται στον ορισμό του συστήματος με μαθηματικούς τύπους και τη μετατροπή στη συνέχεια του μαθηματικού αυτού μοντέλου έτσι ώστε να είναι άμεσα υλοποιήσιμο. Η πιστοποίηση των μερών του συστήματος στο μοντέλο αυτό γίνεται εξασφαλίζοντας ότι κάθε κομμάτι ανταποκρίνεται στο μαθηματικό μοντέλο βάσει του οποίου σχεδιάστηκε. Αν και το μοντέλο αυτό έχει αρκετά κοινά με το μοντέλο «καταρράκτη» υπάρχουν δύο βασικές διαφορές: πρώτον ο καθορισμός των απαιτήσεων εκφράζεται εδώ με μαθηματικό τρόπο και δεύτερον οι διαδικασίες του σχεδιασμού, της υλοποίησης και της δοκιμής έχουν αντικατασταθεί από τη μετατροπή του μαθηματικού μοντέλου των απαιτήσεων σε κάτι άμεσα υλοποιήσιμο.



Σχήμα 2.3 Ανάπτυξη τυποποιημένων συστημάτων

Η τέταρτη και τελευταία προσέγγιση για την ανάπτυξη λογισμικού είναι η ανάπτυξη μέσω συναρμολόγησης (*reuse-based development*). Η προσέγγιση αυτή βασίζεται στην ύπαρξη έτοιμων συστατικών μερών που μπορούν να συναρμολογηθούν μεταξύ τους και να δώσουν έτσι το ζητούμενο σύστημα. Στο μοντέλο αυτό δίνεται περισσότερο βάση στην εύρεση και συναρμολόγηση έτοιμων συστατικών μερών για την ανάπτυξη του λογισμικού παρά στην ανάπτυξη ενός συστήματος από την αρχή. Τα στάδια της διαδικασίας ανάπτυξης στη προσέγγιση αυτή, διαφοροποιούνται σε σχέση με τις άλλες μεθόδους μόνο στα ενδιάμεσα στάδια

μια και ο καθορισμός των απαιτήσεων και η πιστοποίηση του λογισμικού παραμένουν ίδιες. Οι ενδιάμεσες φάσεις στο μοντέλο αυτό είναι [1]:

- η ανάλυση των υπαρχόντων συστατικών μερών που μπορεί να χρησιμεύσουν στην ανάπτυξη
- η μετατροπή των απαιτήσεων που έχουν καθοριστεί αρχικά με βάση τις δυνατότητες των διαθέσιμων συστατικών μερών
- η σχεδίαση του συστήματος που έχει ουσιαστικά να κάνει με τη συναρμολόγηση των μερών με τέτοιο τρόπο ώστε να ικανοποιούνται οι απαιτήσεις και
- η ανάπτυξη και ολοκλήρωση του συστήματος.

Το βασικό πλεονέκτημα της προσέγγισης αυτής είναι η ελαχιστοποίηση του λογισμικού που θα πρέπει να υλοποιηθεί από την αρχή και επομένως και την ελαχιστοποίηση του κόστους και του ρίσκου ανάπτυξης του συστήματος.

Σχήμα 2.4 Το μοντέλο ανάπτυξης μέσω συναρμολόγησης

Εκτός από τις παραπάνω προσεγγίσεις, που η καθεμιά έχει τα πλεονεκτήματά της, υπάρχει η ανάγκη για υποστήριξη της επανάληψης κάποιων σταδίων των παραπάνω μοντέλων κατά τη διαδικασία της ανάπτυξης. Ένα παράδειγμα είναι η περίπτωση όπου θα πρέπει να επανεξεταστούν ο σχεδιασμός και η υλοποίηση όταν υπάρχει αλλαγή στις απαιτήσεις. Για την επανάληψη κάποιων σταδίων της διαδικασίας της ανάπτυξης υπάρχουν δύο προσεγγίσεις [1]:

- η επαυξητική ανάπτυξη (*incremental development*) όπου η ανάλυση των απαιτήσεων, ο σχεδιασμός και η υλοποίηση χωρίζονται σε μια σειρά επαναλήψεων σε κάθε μια από τις οποίες προστίθεται στην εφαρμογή και κάποιο κομμάτι λειτουργικότητας και
- η σπειροειδής ανάπτυξη όπου αντί να θεωρεί τη διαδικασία ανάπτυξης λογισμικού σαν σειριακή με οπισθοδρομήσεις, θεωρεί πως ακολουθεί

σπειροειδή πορεία από το εσωτερικό της σπείρας προς τα έξω φτάνοντας από ένα αρχικό σχέδιο στην τελική εφαρμογή.

2.2. Επιλογή μοντέλου

Στη περίπτωση της μηχανογράφησης του λογιστηρίου της τράπεζας η επιλογή του μοντέλου ανάπτυξης απαιτεί μια μικρή ανάλυση τόσο των παραμέτρων του συστήματος όσο και των πλεονεκτημάτων των παραπάνω μοντέλων. Είναι σαφές λοιπόν ότι οι δύο σημαντικότερες παράμετροι του συστήματος που είναι σε θέση να επηρεάσουν μια τέτοια απόφαση είναι η ανάγκη για πλήρη κάλυψη των απαιτήσεων που θα καθοριστούν και τα δεδομένα χρονικά πλαίσια για την ανάπτυξη του συστήματος. Από τη άλλη μεριά, αναλύοντας τις δυνατότητες των μοντέλων στους τομείς αυτούς, εύκολα διακρίνεται η υπεροχή των μοντέλων βασισμένων στην επανάληψη των δραστηριοτήτων της ανάπτυξης όσον αφορά την κάλυψη των απαιτήσεων. Αντίστοιχα, η προσέγγιση μέσω συναρμολόγησης δείχνει να υπερέχει σημαντικά, όσον την αφορά το χρόνο παράδοσης του συστήματος. Επιπλέον κάτι που πρέπει να ληφθεί υπόψη είναι το γεγονός ότι το μοντέλο μέσω συναρμολόγησης είναι μια από τις πιο σύγχρονες και πολλά υποσχόμενες για το μέλλον προσεγγίσεις με σημαντική υποστήριξη από προϊόντα λογισμικού.

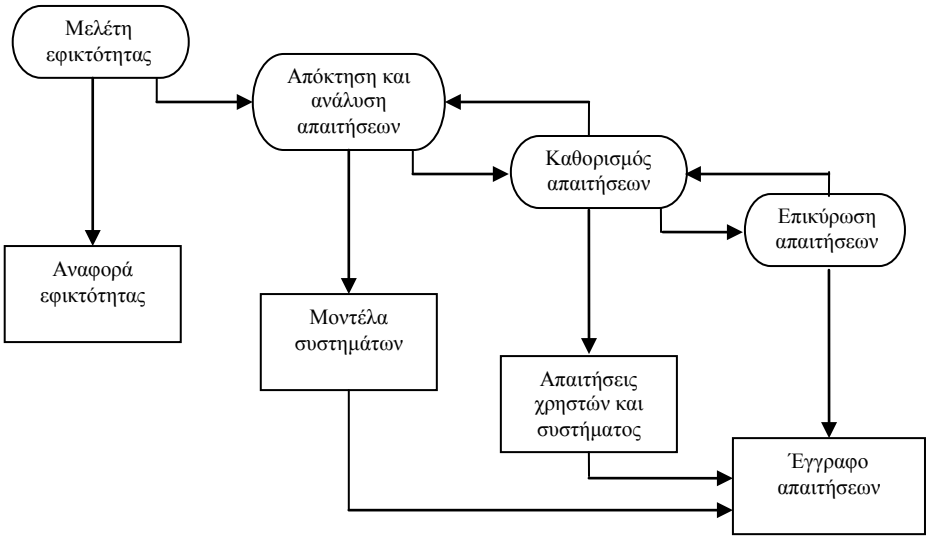
Σταθμίζοντας τους παραπάνω παράγοντες και γνωρίζοντας πως καμία από τις δυο βασικές απαιτήσεις δεν μπορεί να παραμεληθεί, η καταλληλότερη λύση θα ήταν η υιοθέτηση του μοντέλου ανάπτυξης με τη συναρμολόγηση έτοιμων μερών λογισμικού επαναλαμβάνοντας όμως τις δραστηριότητες μέχρι να είναι πλέον βέβαιο ότι το προϊόν της ανάπτυξης είναι σε θέση να καλύψει τις απαιτήσεις που έχουν καθοριστεί. Είναι σαφές ότι η επιλογή προσφέρει τη μέγιστη δυνατή ευελιξία, αφού εκμεταλλεύεται από τη μια μεριά τη ταχύτητα ανάπτυξης της συναρμολόγησης, ενώ παράλληλα δίνει τη δυνατότητα επανεξέτασης και αναθεώρησης μερών του συστήματος που δεν βρίσκονται στο απαιτούμενο επίπεδο. Στη συνέχεια θα εξεταστούν αναλυτικά τα στάδια της ανάπτυξης που όπως έχει προαναφερθεί είναι ο καθορισμός των απαιτήσεων, η ανάλυση των υπαρχόντων συστατικών μερών, η ενδεχόμενη μεταβολή των απαιτήσεων με βάση τις δυνατότητες των μερών που είναι διαθέσιμα, το σχεδιασμό, την υλοποίηση και ολοκλήρωση της εφαρμογής και την πιστοποίηση του συστήματος.

ΚΕΦΑΛΑΙΟ 3ο
ΔΙΑΔΙΚΑΣΙΑ ΑΝΑΠΤΥΞΗΣ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

3.1 Καθορισμός του έργου (*software specification*)

3.1.1 Μελέτη μεθόδων και τεχνικών

Η πρώτη αυτή από τις δραστηριότητες της ανάπτυξης λογισμικού καλείται ουσιαστικά να καθορίσει τις υπηρεσίες που θα πρέπει να παρέχει το υπό ανάπτυξη σύστημα καθώς και τους περιορισμούς λειτουργίας και ανάπτυξής του. Το προϊόν της δραστηριότητας αυτής είναι το έγγραφο των απαιτήσεων που αποτελεί ουσιαστικά και τον ορισμό του συστήματος. Στο έγγραφο αυτό, οι απαιτήσεις παρουσιάζονται σε δύο επίπεδα λεπτομέρειας εκ των οποίων το ένα (μικρότερη λεπτομέρεια) απευθύνεται στους τελικούς χρήστες και τους πελάτες, ενώ το άλλο (λεπτομερές) αφορά τους εμπλεκόμενους στην ανάπτυξη του συστήματος. Αξίζει να σημειωθεί ότι η συγκεκριμένη φάση της ανάπτυξης είναι ίσως η πλέον κρίσιμη αφού ενδεχόμενα λάθη θα δημιουργήσουν σημαντικά προβλήματα στη σχεδίαση και την υλοποίηση του συστήματος. (σχήμα 3.1)



Σχήμα 3.1 Η διαδικασία καθορισμού του έργου

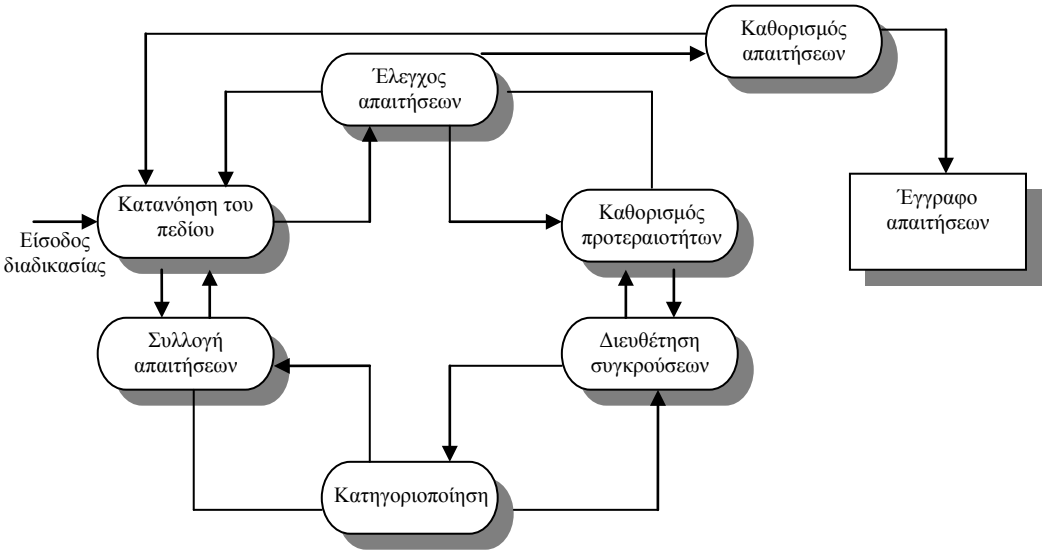
Αν και στη φάση αυτή της ανάπτυξης μπορούν να εφαρμοστούν συγκεκριμένες μέθοδοι για το καθορισμό των απαιτήσεων (π.χ. αντικειμενοστραφής ανάλυση) εδώ θα εξετάσουμε πιο γενικές προσεγγίσεις που καλύπτουν πληρέστερα δραστηριότητες όπως η απόκτηση των απαιτήσεων των χρηστών. Τα τέσσερα βασικά στάδια του σταδίου αυτού είναι [1]:

- η μελέτη εφικτότητας, που είναι μια εκτίμηση για το κατά πόσο είναι εφικτή η ικανοποίηση των απαιτήσεων του πελάτη ώστε το αποτέλεσμα να είναι συμφέρον από εμπορικής πλευράς,
- η απόκτηση απαιτήσεων και ανάλυσή τους,
- ο καθορισμός των απαιτήσεων, που είναι ουσιαστικά η μετατροπή των πληροφοριών που συλλέχθηκαν σε ένα σύνολο απαιτήσεων και
- η επιβεβαίωση των απαιτήσεων όπου οι απαιτήσεις ελέγχονται ως προς την εφικτότητα, τη συνέπεια και τη πληρότητά τους.

Η πρώτη φάση για την ανάπτυξη του λογισμικού είναι η μελέτη εφικτότητας. Το υλικό που απαιτείται για τη μελέτη αυτή είναι μια γενική περιγραφή του έργου καθώς και του τρόπου με τον οποίο εκτιμάται ότι θα χρησιμοποιηθεί μέσα στον οργανισμό ενώ το προϊόν της είναι μια αναφορά που προτείνει ή απορρίπτει την υλοποίηση της συγκεκριμένης ιδέας. Για φτάσει στο συμπέρασμα αυτό θα πρέπει να απαντηθούν ερωτήματα όπως το αν το υπό ανάπτυξη σύστημα συμβαδίζει με τους γενικότερους στόχους του οργανισμού, αν είναι δυνατή η υλοποίησή του με τη παρούσα τεχνολογία και μέσα στα δεδομένα χρονικά και οικονομικά πλαίσια και αν είναι δυνατή η ενσωμάτωση του συστήματος με τα συστήματα που ήδη λειτουργούν στον οργανισμό. Η απάντηση των ερωτημάτων αυτών συνεπάγεται και την απόκτηση των απαραίτητων πληροφοριών που θα πρέπει να αναζητηθούν κυρίως από τους διευθυντές των τμημάτων του οργανισμού, τους μηχανικούς λογισμικού, τους τελικούς χρήστες αλλά και άλλους ανθρώπους που έχουν καλή γνώση του χώρου αλλά και των τεχνολογιών.

Αφού επιβεβαιωθεί η εφικτότητα του εγχειρήματος το επόμενο στάδιο είναι η απόκτηση και ανάλυση των απαιτήσεων. Σε αυτή τη φάση οι υπεύθυνοι για την ανάπτυξη του λογισμικού έρχονται σε επαφή με τους πελάτες και τους χρήστες του τελικού συστήματος. Σκοπός τους είναι να μάθουν για το πεδίο εφαρμογής του λογισμικού, τις υπηρεσίες που θα πρέπει να παρέχει το σύστημα, την απόδοση που θα πρέπει να έχει καθώς και τους πιθανούς περιορισμούς που υπάρχουν. Δραστηριότητες

της φάσης αυτής είναι η κατανόηση του πεδίου, η συλλογή των απαιτήσεων από τους χρήστες η κατηγοριοποίησή τους, η διευθέτηση αλληλοσυγκρουόμενων απαιτήσεων, η ιεράρχησή τους και τέλος ο έλεγχος των τελικών απαιτήσεων όσον αφορά τη πληρότητα, την ακρίβεια και τη συνέπειά τους ως προς τις πραγματικές απαιτήσεις. Για την εκτέλεση των δραστηριοτήτων αυτών εφαρμόζονται μέθοδοι όπως η απόκτηση βασισμένη στην οπτική γωνία κάθε χρήστη ως προς το σύστημα και τα σενάρια χρήσης. (σχήμα 3.2).



Σχήμα 3.2 Η διαδικασία απόκτησης και ανάλυσης των απαιτήσεων

Κατά την ανάπτυξη ενός συστήματος πολλές ομάδες χρηστών παρουσιάζουν ενδιαφέρον για τον καθορισμό των απαιτήσεων. Οι απαιτήσεις των ομάδων αυτών δεν είναι εντελώς ανεξάρτητες μεταξύ τους αλλά συχνά είναι επικαλυπτόμενες οπότε υπάρχουν κοινές απαιτήσεις. Η μέθοδος απόκτησης των απαιτήσεων με βάση τις διαφορετικές οπτικές γωνίες από τις οποίες κάθε ομάδα χρηστών βλέπει το σύστημα αναγνωρίζει τις διαφορετικές αυτές οπτικές γωνίες και παρέχει τρόπους για τη διευθέτηση αλληλοσυγκρουόμενων απαιτήσεων και τη σύνθεση των συνολικών απαιτήσεων για το σύστημα. Σημαντικότερη μέθοδος αυτής της προσέγγισης για την απόκτηση των απαιτήσεων είναι η VORD (*Viewpoint-Oriented Requirements Definition*) όπου θεωρεί σαν οπτική γωνία για το σύστημα έναν εξωτερικό αποδέκτη υπηρεσιών του συστήματος [1].

Πολλές φορές οι άνθρωποι θεωρούν πιο εύκολο να χειριστούν πραγματικά παραδείγματα παρά αφηρημένες περιγραφές. Τέτοιες μέθοδοι μπορούν να

χρησιμοποιηθούν και στην απόκτηση των απαιτήσεων μέσω των σεναρίων (*scenarios*) [2]. Τα σενάρια είναι περιγραφές πραγματικών περιπτώσεων αλληλεπίδρασης του συστήματος με τους χρήστες του. Διαφορετικοί τύποι σεναρίων μπορούν να δώσουν διαφορετικές πληροφορίες για τις απαιτήσεις του συστήματος. Γενικά ένα σενάριο περιλαμβάνει τη κατάσταση του συστήματος πριν την αλληλεπίδραση, τη περιγραφή της ροής των γεγονότων στο σενάριο, τη περιγραφή ενδεχόμενων ανωμαλιών και τους τρόπους αντιμετώπισής τους, τις πληροφορίες για άλλες ενέργειες που συμβαίνουν παράλληλα και τη περιγραφή της κατάστασης του συστήματος μετά την αλληλεπίδραση. Τα σενάρια μπορούν να χρησιμοποιηθούν από τους μηχανικούς λογισμικού είτε άτυπα συζητώντας απευθείας με τους χρήστες για τα σενάρια είτε χρησιμοποιώντας κάποια πιο δομημένη μέθοδο όπως σενάρια γεγονότων (*event scenarios*) και περιπτώσεις χρήσης (*use-cases*) [2]. Τα σενάρια γεγονότων χρησιμοποιούνται για να περιγράψουν τη συμπεριφορά του συστήματος σε συγκεκριμένα γεγονότα. Κάθε ξεχωριστή αλληλεπίδραση μπορεί να αποτελέσει ένα σενάριο γεγονότος. Τα σενάρια αυτά περιέχουν συνήθως τη ροή των δεδομένων κατά την αλληλεπίδραση, τις ενέργειες του συστήματος καθώς και τα προβλήματα που μπορεί να προκύψουν.

Από την άλλη μεριά οι περιπτώσεις χρήσεις (*use-cases*) στη πιο απλή τους μορφή εντοπίζουν τους τύπους των χρηστών που συμμετέχουν σε μια αλληλεπίδραση και καταγράφουν το τύπο της αλληλεπίδρασης. Το σύνολο των περιπτώσεων χρήσης αναπαριστά το σύνολο των αλληλεπιδράσεων που θα χρησιμοποιηθούν στις απαιτήσεις του συστήματος. Μερικές φορές είναι δύσκολο να αποφασιστεί αν μια περίπτωση χρήσης αποτελεί και ένα σενάριο. Μια περίπτωση χρήσης συνήθως περιλαμβάνει ένα αριθμό σεναρίων, από τα όποια κάθε ένα είναι μια συγκεκριμένη διαδρομή που μπορεί να ακολουθήσει η αλληλεπίδραση. Έτσι για παράδειγμα, σε κάθε περίπτωση χρήσης υπάρχει τουλάχιστον ένα σενάριο για τη φυσιολογική περίπτωση και ένα σενάριο για κάθε πιθανό πρόβλημα που μπορεί να προκύψει.

Τελευταίο στάδιο της διαδικασίας καθορισμού των απαιτήσεων είναι ο έλεγχος και η επικύρωση τους. Ο έλεγχος αυτός έχει να κάνει ουσιαστικά με την επιβεβαίωση του ότι οι απαιτήσεις που έχουν καταγραφεί ορίζουν στη πραγματικότητα το σύστημα που απαιτεί ο χρήστης. Το στάδιο αυτό είναι ιδιαίτερα σημαντικό καθώς τα ενδεχόμενα λάθη στο καθορισμό των απαιτήσεων, θα εμφανιστούν στη συνέχεια της διαδικασίας ανάπτυξης. Το γεγονός αυτό με τη σειρά του συνεπάγεται πολύ μεγαλύτερο κόστος σε σχέση με αυτό των λαθών του

σχεδιασμού ή της υλοποίησης. Για την επικύρωση των απαιτήσεων υπάρχουν μια σειρά ελέγχων που θα πρέπει να πραγματοποιηθούν. Οι έλεγχοι αυτοί είναι [1]:

- έλεγχοι εγκυρότητας, που εξετάζουν το κατά πόσο οι απαιτήσεις είναι αυτές που αντικατοπτρίζουν την πραγματικότητα,
- έλεγχοι συνέπειας, που έχουν σκοπό να εξασφαλίσουν ότι δεν υπάρχουν αλληλοσυγκρουόμενες απαιτήσεις,
- έλεγχοι πληρότητας, που εξασφαλίζουν ότι οι καταγεγραμμένες απαιτήσεις είναι πλήρεις,
- έλεγχοι ρεαλιστικότητας, που εκτιμούν το κατά πόσο η κάλυψη των απαιτήσεων είναι εφικτή με τα υπάρχοντα τεχνολογικά μέσα και
- η δημιουργία ελέγχων για την επαλήθευση της κάλυψης των καθορισμένων απαιτήσεων από το παραδοτέο σύστημα.

Για τη διεξαγωγή αυτών των ελέγχων υπάρχουν συγκεκριμένες τεχνικές επικύρωσης των απαιτήσεων, που μπορεί να χρησιμοποιηθούν ξεχωριστά ή και σε συνδυασμό. Πρώτη από τις μεθόδους αυτές είναι η ανασκόπηση των απαιτήσεων, όπου οι απαιτήσεις αναλύονται συστηματικά από μια ομάδα υπευθύνων για την ανάπτυξη του συστήματος και χρηστών. Υπάρχουν δύο τρόποι ανασκόπησης των απαιτήσεων, η τυπική ανασκόπηση και η ανεπίσημη. Η ανεπίσημη ανασκόπηση είναι ουσιαστικά μια συζήτηση των υπευθύνων για την ανάπτυξη του έργου με τους χρήστες του συστήματος πάνω στις απαιτήσεις που έχουν εντοπιστεί. Η τυπική ανασκόπηση είναι μια διαδικασία όπου η ομάδα ανάπτυξης του έργου καλείται να «ξεναγήσει» τους χρήστες στις απαιτήσεις του συστήματος αναλύοντας το σκεπτικό κάθε απαίτησης.

Μια διαφορετική μέθοδος είναι η δημιουργία προτύπων όπου οι απαιτήσεις μετατρέπονται σε ένα λειτουργικό μοντέλο που παρέχεται στους χρήστες. Οι χρήστες με τη σειρά τους πειραματίζονται με το μοντέλο αυτό με σκοπό να βεβαιωθούν ότι ανταποκρίνεται στις πραγματικές τους απαιτήσεις. Μια ακόμα μέθοδος είναι η δημιουργία δοκιμαστικών σεναρίων για τις απαιτήσεις. Αν για κάποιες απαιτήσεις η δημιουργία τέτοιων δοκιμών παρουσιάζει δυσκολίες πιθανότατα θα υπάρξει ανάλογη δυσκολία κατά την υλοποίηση της συγκεκριμένης απαίτησης. Τέλος μια μέθοδος για τον έλεγχο των απαιτήσεων είναι και η αυτοματοποιημένη ανάλυση της συνέπειας των απαιτήσεων. Ουσιαστικά πρόκειται για χρήση συγκεκριμένων εργαλείων που έχουν σαν στόχο να ελέγξουν τη συνέπεια των απαιτήσεων και εντοπίσουν ασυμφωνίες και αλληλοσυγκρουόμενες απαιτήσεις. Η μέθοδος αυτή μπορεί φυσικά

να εφαρμοστεί μόνο στη περίπτωση που οι απαιτήσεις μπορούν να εκφραστούν ως ένα μοντέλο συστήματος με ένα δομημένο τρόπο περιγραφής (π.χ. μαθηματικό μοντέλο).

Είναι γεγονός ότι ο καθορισμός των απαιτήσεων δεν τελειώνει σε αυτό στο στάδιο του έργου. Κατά τη διάρκεια της ανάπτυξης πολλές φορές θα χρειαστεί να επαναπροσδιοριστούν οι απαιτήσεις καθώς τα προβλήματα θα γίνονται περισσότερο κατανοητά. Είναι σημαντικό λοιπόν οι αλλαγές στις απαιτήσεις να γίνουν κατανοητές και να ελεγχθούν. Για να γίνει αυτό οι απαιτήσεις χωρίζονται σε δυο κατηγορίες, τις σταθερές που παρουσιάζουν μικρή πιθανότητα να αλλάξουν στο μέλλον και τις ευμετάβλητες που η πιθανότητα να μεταβληθούν κατά τη διάρκεια της ανάπτυξης είναι αυξημένη.

Αφού λοιπόν αναγνωριστούν οι απαιτήσεις και καταγραφούν όλες οι απαραίτητες πληροφορίες για τη πηγή και τη σχέση τους με τις υπόλοιπες αλλά και το σχεδιασμό, αποφασίζεται η διαδικασία με την οποία θα γίνεται η μεταβολή τους. Η διαδικασία αυτή περιλαμβάνει τρία στάδια [1]:

- την ανάλυση του προβλήματος και το καθορισμό των αλλαγών που προτείνονται,
- την ανάλυση των αλλαγών και του κόστους που συνεπάγονται και
- την υλοποίηση των αλλαγών που μπορεί να επηρεάσει εκτός από το έγγραφο των απαιτήσεων το σχεδιασμό και την υλοποίηση.

Η χρησιμοποίηση μιας τυπικής διαδικασίας για τις μεταβολές των απαιτήσεων είναι ιδιαίτερα σημαντική καθώς με τον τρόπο αυτό εξασφαλίζεται ότι όλες οι προτάσεις αλλαγών αντιμετωπίζονται με τον ίδιο τρόπο και όλη η διαδικασία της αλλαγής παραμένει υπό έλεγχο.

3.1.2 Καθορισμός του έργου στη περίπτωση του λογιστηρίου

3.1.2.1 Γενικά

Είναι γεγονός ότι στη περίπτωση του λογιστηρίου υπάρχουν ορισμένες ιδιαιτερότητες που διαφοροποιούν αρκετά τη διαδικασία σε σχέση με τη γενική περίπτωση την οποία αφορούν οι μέθοδοι που περιγράφηκαν. Ένα πολύ σημαντικό χαρακτηριστικό της προσπάθειας ανάπτυξης του υποσυστήματος του λογιστηρίου,

που διαφοροποιεί τη συγκεκριμένη περίπτωση είναι η ύπαρξη ενός υπάρχοντος πακέτου λογισμικού που καλύπτει τις ανάγκες των τραπεζών αυτή τη στιγμή. Το στάδιο της ανάπτυξης του νέου συστήματος που θα επηρεαστεί περισσότερο από κάθε άλλο από την ιδιαιτερότητα αυτή είναι αναμφισβήτητα το στάδιο του καθορισμού του έργου, όπου οι διαδικασίες που θα ακολουθηθούν θα είναι πολύ διαφορετικές από τις συνήθεις.

Σκοπός της επιλογής αυτής για τη μη πιστή τήρηση των διαδικασιών, είναι η εκμετάλλευση των πολύτιμων πληροφοριών που μπορεί να μας παρέχει το υπάρχον λογισμικό. Το σύστημα αυτό έχει αναπτυχθεί εδώ και αρκετά χρόνια, έχοντας συνεπώς λειτουργήσει για μεγάλο χρονικό διάστημα σε πραγματικές συνθήκες. Οι αλλαγές και οι προσθήκες που έχουν γίνει όλα αυτά τα χρόνια είναι αποτέλεσμα της εμπειρίας πάνω στη χρήση του συστήματος και αποτελούν πολύτιμες πληροφορίες που σε καμία περίπτωση, δε μπορεί να αποκτηθούν από καμία διαδικασία καθορισμού των απαιτήσεων που θα εφαρμοστεί στα πλαίσια της ανάπτυξης του νέου συστήματος. Με άλλα λόγια το υπάρχον σύστημα είναι μια λειτουργική αναπαράσταση των απαιτήσεων που έχουν οριστεί μέσα από μια πολυετή διαδικασία επαναπροσδιορισμού και προσαρμογής, προϊόν της οποίας είναι η τελική μορφή του λογισμικού που λειτουργεί σήμερα. Γίνεται λοιπόν σαφές ότι δεν είναι δυνατόν καμία διαδικασία ορισμού των απαιτήσεων που θα γινόταν στα πλαίσια της δικής μας προσπάθειας, δεν θα ήταν δυνατό να καταλήξει σε ένα πληρέστερο σύνολο απαιτήσεων από αυτό που υλοποιεί το υπάρχον σύστημα και για το λόγο αυτό δεν πρόκειται να διατεθούν πόροι προς τη κατεύθυνση αυτή.

Κάτι που αξίζει ιδιαίτερης προσοχής για την επιτυχία του εγχειρήματος είναι ο διαχωρισμός των διαφορετικών ειδών απαιτήσεων που υπάρχουν για ένα σύστημα. Οι πληροφορίες που είναι σε θέση να παρέχει το υπάρχον λογισμικό μπορεί να είναι πολύτιμες για τον τομέα των λειτουργικών απαιτήσεων του συστήματος, όμως σε καμία περίπτωση δεν μπορεί να καλύψει τις σύγχρονες απαιτήσεις σε θέματα τεχνολογίας. Επιπρόσθετα, οι σύγχρονες τεχνικές και τα σύγχρονα προϊόντα ανάπτυξης λογισμικού επιτρέπουν τη προσθήκη στο σύστημα νέων δυνατοτήτων που δεν αποτελούσαν απαιτήσεις του υπάρχοντος λόγω της αδυναμίας υλοποίησής τους με την τότε τεχνολογία. Συμπέρασμα των διαπιστώσεων αυτών είναι ότι οι πληροφορίες που μπορούν να εξαχθούν από το υπάρχον λογισμικό, αφορούν μόνο τις απαιτήσεις ως προς τη λειτουργικότητα και όχι όλο το φάσμα των απαιτήσεων ενός σύγχρονου πακέτου λογισμικού. Οι απαιτήσεις του υπό ανάπτυξη λογισμικού για

θέματα όπως οι επιδόσεις, η ευχρηστία, η πλατφόρμα λειτουργίας και άλλα χαρακτηριστικά που επηρεάζονται από τις εξελίξεις στο χώρο της ανάπτυξης λογισμικού, θα είναι αντικείμενο νέας μελέτης στα πλαίσια του καθορισμού των απαιτήσεων.

3.2.1.2 Μελέτη εφικτότητας και καθορισμός των λειτουργικών απαιτήσεων

Πρώτο στάδιο του καθορισμού ενός έργου είναι όπως έχει αναφερθεί η μελέτη εφικτότητας. Οι ιδιαιτερότητες του συγκεκριμένου συστήματος έχουν σημαντική επιρροή στο κομμάτι αυτό που ουσιαστικά μπορούμε να πούμε πως αίρουν την αναγκαιότητα της μελέτης εφικτότητας στη συγκεκριμένη περίπτωση. Το συμπέρασμα αυτό είναι μάλλον αυτονόητο αφού οι δυνατότητες για ανάπτυξη συστημάτων στο τραπεζικό χώρο ήταν και ένα από τα βασικά κίνητρα για την απόφαση της ανάπτυξης του συγκεκριμένου έργου. Επίσης, ο μεγάλος αριθμός των τραπεζικών συστημάτων που αυτή τη στιγμή χρησιμοποιεί το σύνολο των τραπεζών, είναι μια έμπρακτη απόδειξη τόσο της εφικτότητας του εγχειρήματος, όσο και του εύρους των απαιτήσεων που μπορεί να καλυφθούν με την υπάρχουσα τεχνολογία. Έχοντας ως δεδομένη την απουσία του κινδύνου αδυναμίας ανάπτυξης ενός δεδομένου συστήματος, μια μελέτη ως προς την εφικτότητα του εγχειρήματος θα ήταν περισσότερο σπατάλη χρόνου, απαραίτητου για πιο κρίσιμα κομμάτια του καθορισμού του έργου.

Επόμενο βήμα για καθορισμό του έργου είναι η απόκτηση και ανάλυση των απαιτήσεων. Θεωρώντας ως δεδομένη την απόφαση για χρησιμοποίηση των λειτουργικών απαιτήσεων του υπάρχοντος συστήματος, μένει να καθοριστεί ο τρόπος με τον οποίο θα αξιοποιηθεί το σύστημα αυτό και θα εξαχθούν οι απαιτήσεις που θα αποτελέσουν τη βάση για το νέο σύστημα. Μια τέτοια μέθοδος που ταιριάζει στις απαιτήσεις της συγκεκριμένης περίπτωσης είναι ο καθορισμός των απαιτήσεων μέσω προτύπων διεπαφών, που στη περίπτωσή μας παρέχονται από το σύστημα που ήδη υπάρχει. Ουσιαστικά αυτό που πρόκειται να γίνει είναι η παράκαμψη της απόκτησης των απαιτήσεων από τους ίδιους τους τελικούς χρήστες, μιας και η διαδικασία αυτή έχει ήδη ολοκληρωθεί στα πλαίσια της ανάπτυξης του υπάρχοντος συστήματος και οι απαιτήσεις υπάρχουν ήδη εκφρασμένες σε πρότυπα διεπαφών. Είναι προφανές ότι με τον τρόπο αυτό γίνεται δυνατή η πλήρης αξιοποίηση των πληροφοριών που μπορεί να παρέχει το υπάρχον λογισμικό ως προς τη λειτουργικότητα, ενώ παράλληλα δεν

υπάρχει κάποιος περιορισμός που να εμποδίζει τον καθορισμό των υπόλοιπων απαιτήσεων του συστήματος. Επίσης ένα ακόμα πλεονέκτημα μιας τέτοιας προσέγγισης για την απόκτηση των λειτουργικών απαιτήσεων είναι ότι είναι πολύ πιθανό να οδηγήσει στην ανάπτυξη ενός συστήματος με την ίδια φιλοσοφία χρήσης με το προκάτοχό του, κάτι που είναι ιδιαίτερα σημαντικό για τις τράπεζες που χρησιμοποιούν το υπάρχον σύστημα.

Ακολουθώντας τη μέθοδο αυτή στο υποσύστημα της λογιστικής αναγνωρίζονται δύο βασικές οντότητες: οι λογαριασμοί της λογιστικής και οι κινήσεις των λογαριασμών αυτών. Όσο και αν φαίνεται περίεργο η εμπειρία έχει δείξει ότι ολόκληρο το σύστημα ενός λογιστηρίου τραπέζης βασίζεται στα χαρακτηριστικά γνωρίσματα αυτών των δύο οντοτήτων και στους πολλούς διαφορετικούς τρόπους που τα στοιχεία αυτά μπορούν να παρουσιαστούν. Οτιδήποτε άλλο μέσα στο υποσύστημα του λογιστηρίου είναι κάποια συγκεκριμένη προβολή ή κάποιο προϊόν επεξεργασίας των ιδιοτήτων που περιγράφουν τις δύο αυτές οντότητες. Μια προσέγγιση των ιδιοτήτων αυτών που εξάγεται από τα πρότυπα των διεπαφών του παλιού συστήματος σε συνδυασμό με τη γνώση της εσωτερικής λειτουργίας του συστήματος και μαζί με μια σειρά οντοτήτων μικρότερης σημασίας με ρόλο κυρίως βοηθητικό, παρουσιάζονται στο έγγραφο των απαιτήσεων του συστήματος στο παράρτημα Α. Οι ιδιότητες αυτές αποτελούν ουσιαστικά και τα δεδομένα του λογιστηρίου που θα πρέπει είναι αποθηκευμένα στη βάση δεδομένων.

Μετά το καθορισμό των βασικών οντοτήτων και των ιδιοτήτων τους σειρά έχει ο καθορισμός των λειτουργιών που θα πρέπει να υποστηρίζει το σύστημα. Η μελέτη των διεπαφών και πάλι δείχνει δύο βασικές κατηγορίες λειτουργιών που θα πρέπει να υλοποιηθούν. Πρώτη από αυτές τις κατηγορίες είναι οι λειτουργίες που αφορούν την διαχείριση της αποθήκευσης των οντοτήτων στη βάση δεδομένων, δηλαδή οι λειτουργίες όπως η εισαγωγή, η διαγραφή, η τροποποίηση και η ανάκτηση των δεδομένων από τη βάση δεδομένων. Οι απαιτήσεις για το σύστημα στο τομέα αυτό δεν παρουσιάζουν συγκεκριμένες δυσκολίες και ιδιαιτερότητες και περιορίζονται μάλλον στις τέσσερις παραπάνω λειτουργίες που είναι άλλωστε και οι απολύτως απαραίτητες για τη διαχείριση των δεδομένων.

Το δεύτερο μέρος της λειτουργικότητας αφορά την υλοποίηση του συνόλου των περιορισμών που διέπουν τη συμπεριφορά και αλληλεπίδραση των οντοτήτων. Στο κομμάτι αυτό της λειτουργικότητας οι απαιτήσεις είναι αρκετά αυξημένες τόσο λόγω του αριθμού των περιορισμών όσο και λόγω της ανάγκης για απόλυτο έλεγχο

των δεδομένων που εισάγονται στο σύστημα. Αυτού του είδος οι απαιτήσεις μπορεί να ξεκινούν από θεμελιώδεις περιορισμούς όπως είναι για παράδειγμα η απόλυτη τήρηση της ιεραρχικής δομής των λογαριασμών ή η εισαγωγή κινήσεως μόνο στους λογαριασμούς που δέχονται εγγραφές, και να φτάνουν σε μεγάλο βαθμό λεπτομέρειας όπως για παράδειγμα η υποχρέωση όλων των λογαριασμών τελευταίου βαθμού να δέχονται εγγραφές. Για την κάλυψη όλων αυτών των περιορισμών σε κάθε επίπεδο λεπτομέρειας χρειάστηκε ο επαναπροσδιορισμός των απαιτήσεων αυτών αρκετές φορές κατά τη διάρκεια της υλοποίησης. Ξεκινώντας από τη κάλυψη των θεμελιωδών περιορισμών αρχικά, φτάσαμε στο τέλος της υλοποίησης στην ικανοποίηση όλων περιορισμών σε κάθε επίπεδο λεπτομέρειας, κάτι που αποδεικνύει και στη πράξη ότι τα στάδια της ανάπτυξης λογισμικού επαναλαμβάνονται και επανεξετάζονται αρκετές φορές κατά τη διάρκεια διαδικασίας ανάπτυξης.

3.2.1.3 Άλλες απαιτήσεις και ιδιαιτερότητες του συστήματος

Μια θεμελιώδης απαίτηση που φάνηκε ξεκάθαρα κατά τη προσπάθεια εντοπισμού των στατικών οντοτήτων στο υποσύστημα του λογιστηρίου, είναι η δυνατότητα επεξεργασίας και προβολής των δεδομένων του συστήματος. Όπως αναφέρθηκε πιο πάνω, το μεγαλύτερο μέρος της λειτουργικότητας βασίζεται στην επεξεργασία και τις διαφορετικές προβολές των ιδιοτήτων των βασικών οντοτήτων. Χαρακτηριστικό παράδειγμα είναι τα ισοζύγια, που αν και έχουν ιδιαίτερη σημασία για τη λογιστική, δεν είναι τίποτα παραπάνω από αθροίσματα των κινήσεων των λογαριασμών. Το γεγονός αυτό αποδεικνύει με το καλύτερο τρόπο αυτό που έχει δείξει η εμπειρία χρήσης του προηγούμενου συστήματος επί σειρά ετών, ότι δηλαδή πολύ βασικές πληροφορίες και λειτουργίες της λογιστικής έχουν άμεση σχέση με τον τρόπο που τα δεδομένα μπορούν να παρουσιαστούν στους χρήστες.

Η δεδομένη αυτή απαίτηση για παροχή δυνατοτήτων ως προς τη προβολή των δεδομένων, οδηγεί αναπόφευκτα στην απόφαση για ανάπτυξη ενός ξεχωριστού μέρους του συστήματος που θα είναι σε θέση να καλύψει τις αυξημένες ανάγκες. Σημαντικός επίσης παράγοντας που συμβάλλει στην απόφαση αυτή είναι και το γεγονός ότι οι απαιτήσεις ως προς τη προβολή ακόμα και των ίδιων δεδομένων δεν είναι πάντα σταθερές για όλους τους χρήστες. Δεν είναι σπάνιο φαινόμενο διαφορετικές ομάδες χρηστών του λογιστηρίου είτε να έχουν διαφορετικές ανάγκες προβολής των ίδιων ακριβώς δεδομένων, είτε να έχουν ανάγκη από προβολή

διαφορετικών δεδομένων για τη διεκπεραίωση ίδιων διαδικασιών. Τέλος, κάτι που αξίζει να σημειωθεί είναι ότι η χρησιμότητα μιας ανεξάρτητης μονάδας λογισμικού που θα ικανοποιεί τις παραπάνω προϋποθέσεις , δεν περιορίζεται μόνο στη κάλυψη των αναγκών του υποσυστήματος του λογιστηρίου, αλλά εύκολα μπορεί να χρησιμοποιηθεί και από ολόκληρο το σύστημα της τράπεζας για κάλυψη των ανάλογων αναγκών.

Επόμενο βήμα μετά τη παραπάνω απόφαση είναι ο σαφής καθορισμών των δυνατοτήτων και των προδιαγραφών του ανεξάρτητου αυτού κομματιού. Οι δύο βασικές ομάδες απαιτήσεων που εξάγονται από τους δύο βασικούς λόγους που οδήγησαν στην ανάπτυξη της μονάδας αυτής είναι πρώτον οι απαιτήσεις ως δυνατότητες προβολής και επεξεργασίας των δεδομένων και δεύτερον οι απαιτήσεις ως προς την ευκολότερη και αποδοτικότερη προσαρμογή των προβολών στις ανάγκες των διαφορετικών ομάδων χρηστών. Όσον αφορά τη πρώτη ομάδα απαιτήσεων αυτό που βασικά απαιτείται είναι η δυνατότητα διαχείρισης του τι θα εμφανιστεί και με ποιο τρόπο καθώς και η δυνατότητα για επεξεργασία των στοιχείων μιας προβολής όπως για παράδειγμα η άθροιση μιας στήλης. Αρχικά οι δύο δυνατότητες που αποδείχθηκαν απαραίτητες για την εφαρμογή είναι η άθροιση μιας στήλης και η δυνατότητα για εμφάνιση του τελευταίου στοιχείου στήλης στη θέση του αθροίσματος, εκτός φυσικά από τη ταξινόμηση και ομαδοποίηση των δεδομένων προς οποιαδήποτε στήλη, που θεωρούνται δεδομένες σε κάθε περίπτωση προβολής δεδομένων.

Για τη δεύτερη ομάδα των απαιτήσεων αυτό που βασικά είναι το ζητούμενο είναι η δυνατότητα για προσαρμογή των προβολών ως προς τα παραπάνω χαρακτηριστικά τους, ανάλογα με τις ανάγκες κάθε ομάδας χρηστών. Αυτό που προφανώς προϋποθέτει η απαίτηση αυτή είναι η δημιουργία ενός μηχανισμού για τη δημιουργία και συντήρηση των προβολών χωρίς τη παρεμβολή στο κώδικα της εφαρμογής, αφού η ανάγκη για αλλαγές στο τομέα αυτό είναι ιδιαίτερα συχνή και θα συνεχίσει να υφίσταται ακόμα και μετά το τέλος της ανάπτυξης. Ένας τέτοιος μηχανισμός θα επέτρεπε τη συντήρηση και προσαρμογή των προβολών με τρόπο αρκετά απλό ώστε να γίνεται ακόμα και από εξειδικευμένους υπαλλήλους της ίδιας της τράπεζας, χωρίς τη παρέμβαση των υπευθύνων της ανάπτυξης του συστήματος κάτι που καλύπτει με το παραπάνω τις ανάγκες των τραπεζών για προσαρμογή των προβολών.

Εκτός από τις αυξημένες ανάγκες για τη προβολή των δεδομένων, ένα ακόμα ιδιαίτερο χαρακτηριστικό του υποσυστήματος του λογιστηρίου είναι και ο τεράστιος όγκος των δεδομένων που θα πρέπει να είναι σε θέση να διαχειριστεί. Ενδεικτικό του όγκου των δεδομένων είναι ότι οι κινήσεις των λογαριασμών σε συνθήκες κανονικής λειτουργίας του συστήματος φτάνουν σε εκατομμύρια εγγραφές με αποτέλεσμα να επιβαρύνεται σημαντικά το σύστημα σε αρκετούς τομείς της λειτουργίας του. Η παράμετρος αυτή είναι σαφές ότι θα πρέπει να ληφθεί υπόψη για την ανάπτυξη του συστήματος ώστε να μην οδηγηθούμε στην ανάπτυξη ενός συστήματος που θα καλύπτει μεν τις λειτουργικές απαιτήσεις αλλά θα είναι δε πρακτικά αδιάφορο αφού δεν θα είναι σε θέση να αποδώσει σε πραγματικές συνθήκες χρήσης. Αν και η ιδιαιτερότητα αυτή θα πρέπει να εκτιμηθεί στο σύνολο των αποφάσεων της ανάπτυξης, υπάρχουν τομείς όπου θα πρέπει να διαμορφωθούν ειδικά για να καλύψουν την ανάγκη αυτή.

Ένας από τους τομείς που απαιτούν ιδιαίτερη προσοχή λόγω του μεγάλου όγκου των δεδομένων είναι φυσικά η ταχύτητα απόκρισης του συστήματος. Είναι μάλλον προφανές ότι ίσως το πρώτο χαρακτηριστικό του συστήματος που θα επηρεαστεί από το μεγάλο όγκο πληροφοριών είναι η ταχύτητα με την οποία το σύστημα θα είναι σε θέση να ανακτά τα δεδομένα και να τα επεξεργάζεται. Η καθυστέρηση αυτή μπορεί να προέρχεται τόσο από την ανάκτηση των δεδομένων από τη βάση δεδομένων, όσο και από την επεξεργασία των δεδομένων μετά την ανάκτησή τους. Η σχεδίαση και η υλοποίηση επομένως των λειτουργιών σε κάθε επίπεδο θα πρέπει να έχει βασικό γνώμονα τη μέγιστη δυνατή απόδοση ως προς τη ταχύτητα αλλά και τη δυνατότητα χειρισμού μεγάλου όγκου δεδομένων γενικότερα. Το γεγονός αυτό βέβαια δε σημαίνει ότι μπορεί να γίνουν και παραχωρήσεις ως τις λειτουργικές απαιτήσεις όπως αυτές έχουν καθοριστεί, μια και οι απαιτήσεις αυτές είναι εξίσου καθοριστικές για τη επιτυχία του συστήματος. Αυτό που είναι το ζητούμενο, είναι να γίνουν οι κατάλληλες επιλογές τόσο σε επίπεδο σχεδίασης όσο και σε επίπεδο υλοποίησης, ώστε να καλυφθεί το σύνολο των καθορισμένων απαιτήσεων με το βέλτιστο δυνατό τρόπο ως προς τις επιδόσεις.

Ένας δεύτερος τομέας της εφαρμογής που χρειάζεται ιδιαίτερη προσοχή λόγω του μεγάλου όγκου των δεδομένων είναι ο σχεδιασμός των διεπαφών. Είναι πολύ σημαντικό για ένα σύστημα πραγματικού χρόνου να επιτρέπει την εύκολη και κυρίως γρήγορη αλληλεπίδραση με τους χρήστες, εκτός φυσικά από τη δική του ταχύτητα απόκρισης. Ο σχεδιασμός των διεπαφών δεν έχει να κάνει τόσο με τη ταχύτητα του

ίδιου του συστήματος όσο με τους ρυθμούς που επιτρέπει στους χρήστες να εργάζονται στο σύστημα. Είναι προφανές ότι ο μεγάλος όγκος πληροφοριών δε συνεπάγεται μόνο αυξημένο φόρτο εργασίας για το σύστημα, αλλά και για τους χρήστες του. Αυτό που θα πρέπει να γίνει κατά τη διαδικασία ανάπτυξης του συστήματος είναι να καθοριστούν οι πραγματικές καθημερινές ανάγκες των χρηστών και να σχεδιαστούν οι διεπαφές με τρόπο που να εξασφαλίζει τη μέγιστη δυνατή παραγωγικότητα τους με αυτές.

Μια πολύ καλή προσέγγιση των αναγκών αυτών, που θα χρησιμοποιηθεί για τον ακριβή καθορισμό των απαιτήσεων, είναι η μορφή των διεπαφών του υπάρχοντος συστήματος. Οι διεπαφές αυτές θεωρείται ότι καλύπτουν με το παραπάνω τις ανάγκες ευχρηστίας αποτελώντας το βασικό πλεονέκτημα του συστήματος έναντι των ανταγωνιστικών. Με δεδομένο αυτό το πλεονέκτημά τους είναι επόμενο να αποτελέσουν τις κατευθυντήριες γραμμές για το σχεδιασμό των διεπαφών και του νέου συστήματος. Στόχος είναι η σχεδίαση των διεπαφών κατά τέτοιο τρόπο ώστε όχι μόνο να δανείζονται τη λειτουργικότητα από το παλιό σύστημα, αλλά να επιφέρουν τις ελάχιστες δυνατές αλλαγές στο τρόπο που οι χρήστες έχουν συνηθίσει να εργάζονται με το σύστημα. Σκεπτικό της επιλογής αυτής είναι η διατήρηση αφενός των ίδιων επιπέδων παραγωγικότητας με το προηγούμενο σύστημα και αφετέρου η επίτευξη της ταχύτερης δυνατής αποδοχής και προσαρμογής των χρηστών στο νέο σύστημα. Για γίνει δυνατή η δημιουργία ενός γραφικού περιβάλλοντος παρόμοιο στη λειτουργία με το προηγούμενο σύστημα οι διαδικασίες ανάλυσης, σχεδίασης και υλοποίησης του γραφικού περιβάλλοντος επαναλήφθηκαν αρκετές φορές. Ξεκινώντας από την κάλυψη των πλέον προφανών απαιτήσεων μετά από λεπτομερή μελέτη της λειτουργίας των παλιών διεπαφών φτάσαμε σε επίπεδο λεπτομέρειας που φυσικά δεν ήταν αναγνωρίσιμο από την αρχή.

Η πρώτη ματιά στη μορφή και λειτουργία των διεπαφών οδηγεί εύκολα στο συμπέρασμα ότι σημαντικός παράγοντας για τη παραγωγικότητα των χρηστών είναι η εργασία με το πληκτρολόγιο. Είναι σαφές ότι η εκτέλεση των λειτουργιών μόνο με το πληκτρολόγιο επιταχύνει σημαντικά το ρυθμό με τον οποίο ο χρήστης χρησιμοποιεί τις λειτουργίες που παρέχονται. Το συμπέρασμα αυτό βγαίνει μέσα από την εμπειρία αρκετών ετών που δείχνει πως οι χρήστες του υποσυστήματος ενδιαφέρονται πολύ περισσότερο για την επιτάχυνση της εκτέλεσης της εργασίας τους από τις ευκολίες ως προς τη κατανόηση που μπορεί να προσφέρει η εκτεταμένη χρήση του ποντικιού σε ένα γραφικό περιβάλλον. Βασικό αίτιο φυσικά της προτίμησης αυτής των χρηστών

είναι ο όγκος της εργασίας, που από τη μία συμβάλλει στη γρήγορη εξοικείωση με το σύστημα και από την άλλη εντείνει την ανάγκη για αυξημένη ταχύτητα κατά τη χρήση.

Αναπόφευκτο αποτέλεσμα των συμπερασμάτων αυτών είναι η απαίτηση για υποστήριξη της χρήσης των λειτουργιών του συστήματος μόνο από το πληκτρολόγιο, τουλάχιστον για τις συχνότερα χρησιμοποιούμενες λειτουργίες που παίζουν σημαντικό ρόλο στη παραγωγικότητα. Η παράμετρος αυτή θα πρέπει να ληφθεί σοβαρά υπόψη κατά τη σχεδίαση των διεπαφών ώστε να γίνει εφικτός ο πλήρης χειρισμός της εφαρμογής από το πληκτρολόγιο, παράλληλα φυσικά με τη δυνατότητα χειρισμού από το ποντίκι, που υποστηρίζεται ούτως ή άλλως αφού πρόκειται για λογισμικό με γραφικό περιβάλλον. Εκτός φυσικά από το χαρακτηριστικό αυτό κατά την υλοποίηση αποκαλύπτεται ένας μεγάλος αριθμός λεπτομερειών που είναι αδύνατο να καταγραφούν επακριβώς λόγω της λεπτομέρειάς τους και θα ήταν καλύτερα να αναφερθούν κατά τα επόμενα στάδια της ανάπτυξης οπότε και προέκυψαν.

Μια από τις απαιτήσεις που καλυπτόταν εν μέρει από το παλιό σύστημα και πρέπει να αποτελέσει μέρος και του νέου συστήματος είναι η διαχείριση των μηνυμάτων που εμφανίζονται στο χρήστη κατά τη διάρκεια της εφαρμογής και γενικότερα η διαχείριση οποιουδήποτε κειμένου εμφανίζεται οπουδήποτε στην εφαρμογή. Η ανάγκη αυτή για παραμετροποίηση των μηνυμάτων διευρύνεται με τη διατύπωση της απαίτησης για πολυγλωσσική υποστήριξη από το νέο σύστημα δημιουργώντας έτσι ένα σύνολο απαιτήσεων που αφορά τη συντήρηση της εφαρμογής. Η σημασία που έχουν οι δυνατότητες συντήρησης για την εμπορική επιτυχία του συστήματος έχει αποδειχτεί μέσα από την εμπειρία πολύ μεγάλη, αφού σχετίζεται άμεσα με τις υπηρεσίες που μπορούν να προσφερθούν μετά τη πώληση στους πελάτες. Υποθέτοντας ότι το νέο σύστημα θα έχει ανάλογη εμπορική αποδοχή με το προηγούμενο, είναι ευνόητο ότι θα υπάρχει σημαντικό πρόβλημα για τη συντήρηση και υποστήριξη, αν δεν δημιουργηθούν κατά την ανάπτυξη οι μηχανισμοί εκείνοι που θα απλοποιήσουν συντήρηση του, χωρίς φυσικά την επέμβαση στο κώδικα.

Ένα χαρακτηριστικό των τραπεζικών συστημάτων, που θα πρέπει να υποστηριχθεί, είναι και η ύπαρξη χρηστών με διαφορετικά επίπεδα δικαιωμάτων. Το χαρακτηριστικό αυτό αν και αφορά το σύνολο του τραπεζικού συστήματος, θα πρέπει να υλοποιηθεί για τις ανάγκες του λογιστηρίου, αφού είναι απαραίτητο για τις

σημειώσεις των λογαριασμών. Κάθε λογαριασμός έχει επτά επίπεδα σημειώσεων και κάθε χρήστης μπορεί να έχει πρόσβαση σε όλα τα επίπεδα που είναι μικρότερα ή ίσα με το επίπεδο δικαιωμάτων του, δημιουργώντας έτσι την ανάγκη για ύπαρξη χρηστών με διαφορετικά επίπεδα δικαιοδοσίας. Εάν στο δεδομένο αυτό προστεθούν και άλλα χαρακτηριστικά των χρηστών όπως το κατάστημα και η προτιμώμενη γλώσσα, γίνεται σαφές πως θα πρέπει να αναπτυχθεί ένα σύστημα διαχείρισης χρηστών στα πλαίσια του υποσυστήματος της λογιστικής. Σκοπός βέβαια δε μπορεί να είναι η ανάπτυξη ενός υποσυστήματος που θα καλύπτει τις συνολικές ανάγκες της εφαρμογής ως προς τη διαχείριση των χρηστών, αλλά η δημιουργία ενός μηχανισμού που θα καλύπτει τις ανάγκες του λογιστηρίου και θα μπορεί να επεκταθεί για να καλύψει και τις ανάγκες ολόκληρου του τραπεζικού συστήματος. Με άλλα λόγια αυτό που έχει σημασία αυτή τη στιγμή, δεν είναι το τι χαρακτηριστικά θα περιλαμβάνει η οντότητα του χρήστη του συστήματος. Αυτό που πραγματικά πρέπει να γίνει, είναι η δόμηση του μηχανισμού με τέτοιο τρόπο, ώστε για οτιδήποτε χρειαστεί να προστεθεί στο μέλλον, κατά την ανάπτυξη των υπόλοιπων υποσυστημάτων της τράπεζας, να μην χρειαστεί αλλαγή και στο τρόπο λειτουργίας του.

Ολοκληρώνοντας το καθορισμό των απαιτήσεων του συστήματος διατυπώνονται δύο ανάγκες που μπορεί να μην έχουν άμεση σχέση με τη σχεδίαση και την υλοποίηση του συγκεκριμένου συστήματος, όμως θα παίξουν ρόλο στην επιλογή των εργαλείων ανάπτυξης και επομένως μπορεί να θέσουν περιορισμούς στο τι μπορεί τελικά να υλοποιηθεί. Η πρώτη από αυτές τις απαιτήσεις είναι η χρήση των Microsoft Windows σαν πλατφόρμα στην οποία θα λειτουργεί το σύστημα. Η επιλογή της πλατφόρμας αυτής διευκολύνει σημαντικά τη προσπάθεια αφού οποιαδήποτε άλλη και ήταν η επιλογή θα υπήρχαν σημαντικά προβλήματα στην εύρεση των κατάλληλων εργαλείων τόσο για τη βάση δεδομένων όσο και για την ανάπτυξη του λογισμικού. Τέλος, ο δεύτερος περιορισμός είναι η δυνατότητα για δικτυακή χρήση του συστήματος, που μάλλον θεωρείται ως αυτονόητος από τη στιγμή που θα χρησιμοποιηθεί βάση δεδομένων για τις ανάγκες της αποθήκευσης, όμως και αυτός έχει να παίξει το ρόλο του στην επιλογή της τεχνολογίας ανάπτυξης.

3.2.1.4 Συμπεράσματα του καθορισμού του έργου

Συνοψίζοντας τέλος όλη τη διαδικασία της ανάλυσης των απαιτήσεων, γίνεται άμεσα αντιληπτή η καθοριστική επιρροή της ύπαρξης ενός υπάρχοντος συστήματος που μπορεί να λειτουργήσει σαν βάση για την ανάπτυξη του νέου. Αυτό ακριβώς το γεγονός διαφοροποίησε σημαντικά την όλη διαδικασία του καθορισμού του έργου, περιορίζοντας σημαντικά τη προσπάθεια που θα έπρεπε να γίνει σε άλλη περίπτωση και παρέχοντας τις καλύτερες δυνατές εγγυήσεις για την ακρίβεια και πληρότητα των απαιτήσεων. Επειδή η προσέγγιση της απόκτησης των απαιτήσεων μέσω προτύπων είναι μια καθαρά επαναλαμβανόμενη διαδικασία όπου τα στάδια του καθορισμού, του σχεδιασμού και της υλοποίησης επαναλαμβάνονται αρκετές φορές μέχρι να καλυφθούν όλες οι λειτουργικές απαιτήσεις που περιλαμβάνει το πρότυπο, ο καθορισμός των απαιτήσεων σε καμία περίπτωση δεν σταματά εδώ. Αυτό που ουσιαστικά καθορίστηκε είναι το σύνολο των λειτουργικών απαιτήσεων εκφρασμένο σαν πρότυπες διεπαφές και μια σειρά σημαντικών περιορισμών και απαιτήσεων για τη δομή του συστήματος, με καθοριστικό ρόλο στην κάλυψη των πραγματικών αναγκών του λογιστηρίου μιας τράπεζας.

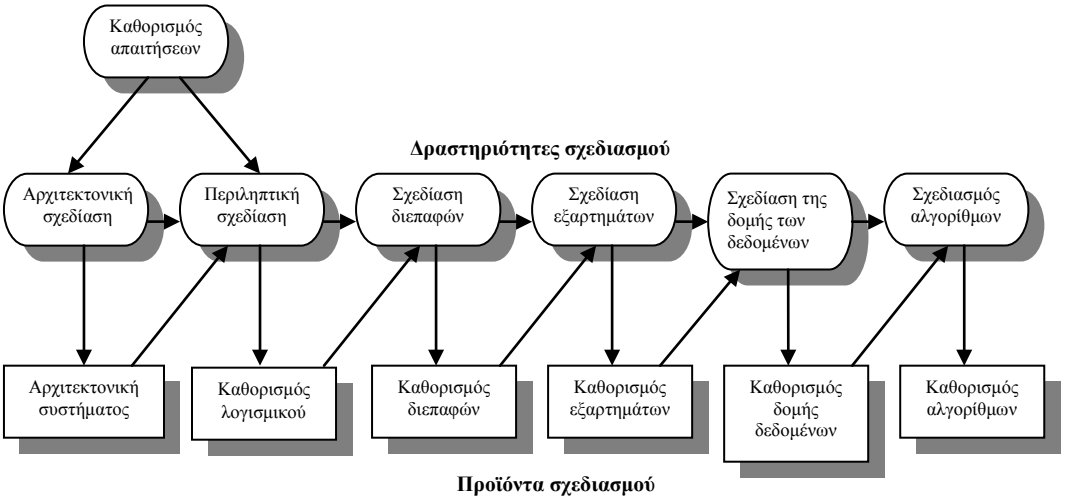
3.2 Σχεδιασμός

3.2.1. Ανάλυση της διαδικασίας σχεδιασμού συστημάτων λογισμικού

Επόμενη φάση στη διαδικασία ανάπτυξης λογισμικού είναι η σχεδίαση του συστήματος. Η σχεδίαση του λογισμικού είναι ουσιαστικά μια περιγραφή της δομής του λογισμικού που πρόκειται να υλοποιηθεί, των δεδομένων που συμμετέχουν στο σύστημα, των διεπαφών μεταξύ των συστατικών μερών του συστήματος καθώς και των αλγορίθμων που πρόκειται να χρησιμοποιηθούν. Φυσικά η διαδικασία του σχεδιασμού δεν ολοκληρώνεται σε ένα μόνο στάδιο, αλλά από μετά πολλές αναθεωρήσεις, σε κάθε μια από τις οποίες διορθώνονται λάθη προηγούμενων εκδόσεων και αυξάνεται το επίπεδο λεπτομέρειας. Είναι συνήθης τακτική στη σχεδίαση λογισμικού να δημιουργείται αρχικά ένα αρκετά αφηρημένο μοντέλο του συστήματος και στη πορεία της διαδικασίας του σχεδιασμού το μοντέλο να διορθώνεται αλλά και να γίνεται πιο συγκεκριμένο αυξάνοντας το επίπεδο λεπτομέρειας.

Η διαδικασία της σχεδίασης μπορεί να χωριστεί σε επιμέρους δραστηριότητες. Αν και οι δραστηριότητες αυτές εμφανίζονται να εκτελούνται σειριακά, στη πραγματικότητα συχνά θα χρειαστεί οπισθοδρόμηση και αναθεώρηση της δουλειάς που έχει γίνει σε προηγούμενα στάδια [1]. Πρώτη από τις δραστηριότητες αυτές είναι η αρχιτεκτονική σχεδίαση (*architectural design*) του συστήματος, που έχει να κάνει με το χωρισμό του σε επιμέρους υποσυστήματα και τη καταγραφή των αλληλεπιδράσεων μεταξύ τους. Στο επόμενο βήμα ακολουθεί η περιγραφή και τεκμηρίωση των υποσυστημάτων αυτών σε χαμηλό βαθμό λεπτομέρειας (*abstract specification*), ενώ στη συνέχεια θα πρέπει να σχεδιαστούν οι διεπαφές των υποσυστημάτων, έτσι ώστε να είναι δυνατή η αλληλεπίδραση με κάποιο υποσύστημα, χωρίς να απαιτείται γνώση των λειτουργιών που εκτελούνται εσωτερικά. Αμέσως μετά, οι υπηρεσίες που καλείται να παρέχει το σύστημα κατανέμονται σε συστατικά μέρη, δημιουργώντας έτσι τη σχεδίαση των συστατικών μερών του λογισμικού (*component design*). Η δραστηριότητα αυτή του σχεδιασμού μπορεί να επηρεαστεί σημαντικά από την ύπαρξη έτοιμων συστατικών, που μπορεί να χρησιμοποιηθούν στην υλοποίηση του συστήματος, μειώνοντας τον αριθμό των μερών που πρέπει να υλοποιηθούν. Τέλος οι δύο τελευταίες δραστηριότητες του σχεδιασμού αφορούν το καθορισμό και τεκμηρίωση των δομών δεδομένων, που απαιτούνται και των

αλγορίθμων που πρόκειται να χρησιμοποιηθούν στα συστατικά μέρη για την υλοποίηση των υπηρεσιών (σχήμα 3.3).



Σχήμα 3.3 Ένα γενικό μοντέλο της διαδικασίας σχεδιασμού

Τις περισσότερες φορές ο σχεδιασμός δεν είναι μια τυποποιημένη διαδικασία. Συνήθως από ένα σύνολο απαιτήσεων γραμμένων σε φυσική γλώσσα (μη τυποποιημένη μορφή) παράγεται μια πρώτη ανεπίσημη έκδοση της σχεδίασης που θα τελειοποιηθεί κατά την υλοποίηση. Παρόλα αυτά υπάρχουν και δομημένες μέθοδοι για τη σχεδίαση λογισμικού. Χαρακτηριστικό των μεθόδων αυτών είναι ότι αν και γενικά παράγουν μεγάλη ποσότητα εγγράφων για το σχεδιασμό, εξασφαλίζουν ότι θα παραχθεί τουλάχιστον η απαραίτητη τεκμηρίωση για το στάδιο αυτό, κάτι που είναι απαραίτητο για έργα μεγάλης κλίμακας. Μια τέτοια μέθοδος περιλαμβάνει ένα μοντέλο της διαδικασίας σχεδιασμού, μια κωδικοποίηση (συμβολισμούς) για την περιγραφή της περιγραφή της σχεδίασης, πρότυπα για τα έγγραφα που παράγονται και κανόνες που θα πρέπει να ακολουθεί η σχεδίαση. Τα τέσσερα βασικά μοντέλα [1] για τη σχεδίαση του συστήματος που ακολουθούν οι μέθοδοι είναι:

- το μοντέλο ροής δεδομένων (*data – flow model*), όπου η σχεδίαση γίνεται με βάση τους μετασχηματισμούς των δεδομένων μέσα στο σύστημα,
- το μοντέλο οντοτήτων – συσχετίσεων (*entity – relation model*), όπου το σύστημα περιγράφεται ως ένα σύνολο οντοτήτων και των μεταξύ τους συσχετίσεων,
- το δομικό μοντέλο (*structural model*) όπου καταγράφονται τα συστατικά μέρη του συστήματος και οι αλληλεπιδράσεις τους και

- το αντικειμενοστραφές μοντέλο (*object – oriented model*), όπου περιλαμβάνει ένα μοντέλο κληρονομικότητας μεταξύ των αντικειμένων, ένα μοντέλο των στατικών και δυναμικών συσχετίσεων μεταξύ των αντικειμένων καθώς και ένα μοντέλο για την αλληλεπίδραση μεταξύ των αντικειμένων του συστήματος.

Στη πράξη οι παραπάνω τεχνικές είναι περισσότερο γενικές κατευθυντήριες γραμμές, που παρέχουν ευρέως αποδεκτούς τρόπους περιγραφής του συστήματος και σπάνια είναι μέθοδοι που ακολουθούνται πιστά. Συνήθως οι σχεδιαστές επιλέγουν ορισμένα στοιχεία από κάθε μέθοδο, που είναι κατάλληλα για τη συγκεκριμένη περίπτωση και δημιουργούν έτσι τη σχεδίαση. Είναι σαφές επομένως, ότι σημαντικός παράγοντας στο τελικό προϊόν του σχεδιασμού είναι η δημιουργικότητα του σχεδιαστή, που καλείται τελικά με τη βοήθεια των παραπάνω κατευθυντήριων γραμμών, να εξασφαλίσει ότι το προϊόν του σχεδιασμού ικανοποιεί τις απαιτήσεις που αναγνωρίστηκαν στο προηγούμενο στάδιο της ανάπτυξης.

Είναι γεγονός ότι σε άλλους τομείς της τεχνολογίας, όπως τα ηλεκτρονικά, η σχεδίαση συστημάτων μέσω συναρμολόγησης έτοιμων κομματιών είναι όχι μόνο διαδεδομένη, αλλά αποτελεί και το κύριο τρόπο ανάπτυξης νέων προϊόντων. Η ανάγκη για μείωση του κόστους και του χρόνου ανάπτυξης του λογισμικού έχει οδηγήσει στην υιοθέτηση μιας ανάλογης προσέγγισης και στην σχεδίαση λογισμικού. Η χρήση βέβαια έτοιμων κομματιών στην ανάπτυξη του λογισμικού είναι εφικτή ακόμα και χωρίς υπάρχει μέριμνα κατά τη σχεδίαση, όταν τυχαίνει να βρεθεί έτοιμο κάποιο κομμάτι, που ικανοποιεί κάποια απαίτηση. Η συστηματική χρήση όμως, της προσέγγισης αυτής, απαιτεί μια διαδικασία σχεδιασμού, που λαμβάνει υπόψη της τον τρόπο που θα συναρμολογηθούν τα έτοιμα μέρη λογισμικού και προσαρμόζει ολόκληρη τη σχεδίαση στη συνένωση διαθέσιμων συστατικών μερών. Το μέγεθος των μερών που μπορεί να χρησιμοποιηθούν μπορεί να ξεκινάει από την υλοποίηση μιας απλής συνάρτησης (π.χ. την εκτέλεση ενός μαθηματικού υπολογισμού) και να φτάνει ως την ενσωμάτωση ολόκληρων εφαρμογών στο σύστημα.

Η υιοθέτηση μιας τέτοιας προσέγγισης στο στάδιο του σχεδιασμού έχει σαφή πλεονεκτήματα για την ανάπτυξη [1]:

- αυξημένη αξιοπιστία, χρησιμοποιώντας έτοιμα μέρη που έχουν ελεγχθεί,

- μείωση του ρίσκου αποτυχίας, αφού είναι πολύ πιο ασφαλές το να συναρμολογηθούν έτοιμα κομμάτια από το να αναπτυχθεί ένα σύστημα από την αρχή,
- πιο αποδοτική χρήση των προγραμματιστών με την ανάπτυξη κομματιών λογισμικού που μπορούν να ενσωματωθούν σε πολλά διαφορετικά συστήματα,
- βελτιωμένη συμβατότητα μεταξύ των συστημάτων, μια και συχνά χρησιμοποιούν τα ίδια συστατικά μέρη και
- σημαντική επιτάχυνση της διαδικασίας ανάπτυξης.

Παρόλα αυτά για να είναι δυνατή η συστηματική χρήση συναρμολογούμενων μερών στην ανάπτυξη ενός συστήματος πρέπει να τηρούνται τρία απαραίτητα κριτήρια [1]:

- η ύπαρξη σημαντικού αριθμού συναρμολογούμενων μερών λογισμικού που μπορούν να φανούν χρήσιμα,
- η βεβαιότητα ότι τα μέρη που θα χρησιμοποιηθούν παρέχουν την απαραίτητη αξιοπιστία και
- η ύπαρξη επαρκών πληροφοριών για τα συστατικά μέρη που θα χρησιμοποιηθούν.

Οι τρεις αυτές προϋποθέσεις είναι εξαιρετικά σημαντικές για την εξασφάλιση των πλεονεκτημάτων που αναφέρθηκαν πιο πάνω.

Η ανάπτυξη λογισμικού μέσω ανεξάρτητων συστατικών μερών (*component-based development και component-based software engineering*), εμφανίστηκε προς το τέλος της δεκαετίας του ’90 ως μια προσέγγιση για την ανάπτυξη λογισμικού μέσω συναρμολόγησης. Αιτία για την εμφάνιση της προσέγγισης αυτής, ήταν η διαπίστωση ότι τα αντικείμενα, όπως χρησιμοποιούνταν μέχρι τότε, ήταν πολύ εξειδικευμένα με αποτέλεσμα να μην είναι δυνατή η συναρμολόγησή τους σε διαφορετικά συστήματα. Για να γίνει δυνατή η χρησιμοποίηση των αντικειμένων, απαιτούνταν η προσαρμογή των λεπτομερειών τους στις ανάγκες των συστημάτων, που με τη σειρά του συνεπάγεται τη γνώση του πηγαίου κώδικα των αντικειμένων, κάτι που εμπορικά παρουσιάζει αρκετά προβλήματα.

Σε αντίθεση με τα αντικείμενα, τα εξαρτήματα λογισμικού (*software components*) είναι πιο αφηρημένα και μπορούν να θεωρηθούν ως ανεξάρτητοι κομμάτια λογισμικού, που μπορούν να παρέχουν υπηρεσίες. Όταν το σύστημα χρειάζεται τις υπηρεσίες ενός εξαρτήματος απλά το καλεί, χωρίς να χρειάζεται να έχει

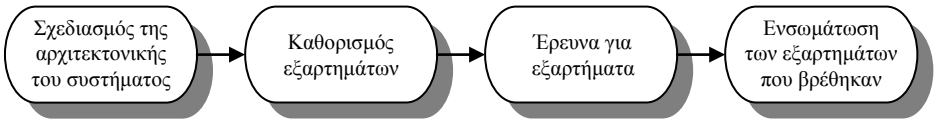
γνώση του τι εκτελεί το εξάρτημα ή ακόμα και σε τι γλώσσα προγραμματισμού έχει αναπτυχθεί. Ένα απλό παράδειγμα ενός εξαρτήματος θα μπορούσε να είναι η υλοποίηση μιας μαθηματικής έκφρασης όπως η τετραγωνική ρίζα ενός αριθμού, το ημίτονο κ.τ.λ. . Όταν το σύστημα θα χρειαστεί τη τετραγωνική ρίζα ενός αριθμού, απλά θα καλέσει το συγκεκριμένο εξάρτημα να του την παρέχει.

Η αντιμετώπιση των εξαρτημάτων σαν ανεξάρτητων παροχέων υπηρεσιών δίνει έμφαση σε δύο συγκεκριμένα χαρακτηριστικά τους [1]:

- ένα εξάρτημα είναι μία ανεξάρτητη εκτελέσιμη οντότητα που ο πηγαίος κώδικάς της δεν είναι γνωστός και δεν συμπεριλαμβάνεται στο πηγαίο κώδικα του συστήματος και
- η αλληλεπίδραση του εξαρτήματος με το σύστημα γίνεται μέσω της διεπαφής, που παρέχει το εξάρτημα, χωρίς να γίνεται ποτέ γνωστή η εσωτερική του κατάσταση.

Κατά τη προσέγγιση αυτή, ένα εξάρτημα ορίζεται από τη διεπαφή του, που αποτελείται από δύο μέρη [1]:

- τη διεπαφή που αφορά τις υπηρεσίες που μπορεί το εξάρτημα να παρέχει στο σύστημα και
- τη διεπαφή που αφορά τις απαιτήσεις που έχει το εξάρτημα από το σύστημα για είναι σε θέση να παρέχει τις υπηρεσίες του.

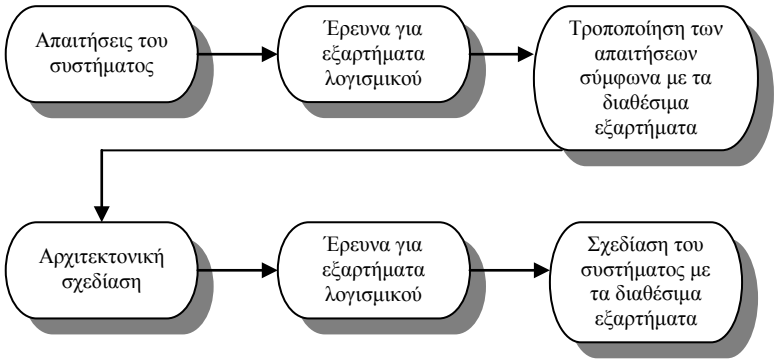


Σχήμα 3.4 Μια ιδανική εκδοχή της διαδικασίας συναρμολόγησης

Η διαδικασία ανάπτυξης λογισμικού με τη χρήση έτοιμων εξαρτημάτων λογισμικού μπορεί να ενσωματωθεί στην όλη διαδικασία της ανάπτυξης με τον τρόπο που φαίνεται στο σχήμα (3.4). Αφού ο σχεδιαστής έχει ολοκληρώσει τη σχεδίαση του συστήματος και έχει καθορίσει τα χαρακτηριστικά των εξαρτημάτων, ξεκινάει η έρευνα για τα κατάλληλα εξαρτήματα. Η ενσωμάτωση των εξαρτημάτων μπορεί να γίνει είτε σε επίπεδο αρχιτεκτονικού σχεδιασμού, είτε σε πιο λεπτομερή επίπεδα της σχεδίασης. Η παραπάνω διαδικασία αν και οδηγεί σε κάποιο βαθμό στη συναρμολόγηση έτοιμων μερών, έχει σημαντικές διαφορές με τη προσέγγιση των

άλλων τεχνολογικών επιστημών στη συναρμολόγηση, αφού σε αυτές η ύπαρξη συγκεκριμένων εξαρτημάτων είναι αυτή που οδηγεί τη διαδικασία του σχεδιασμού.

Στην ανάπτυξη λογισμικού που είναι βασισμένη στην ιδέα της συναρμολόγησης εξαρτημάτων, οι απαιτήσεις του συστήματος είναι δυνατό να προσαρμοστούν στα υπάρχοντα εξαρτήματα. Αφού ολοκληρωθεί ο καθορισμός των απαιτήσεων, αρχίζει η έρευνα για εξαρτήματα που μπορούν να χρησιμοποιηθούν. Στη συνέχεια, αφού επανεξετασθούν οι απαιτήσεις σε σχέση με τα υπάρχοντα συστατικά μέρη που βρέθηκαν, σχεδιάζεται το σύστημα γύρω από τα εξαρτήματα αυτά (σχήμα 3.5). Η σχεδιαστική αυτή προσέγγιση είναι βέβαια λιγότερο αποδοτική, από ότι θα ήταν μια σχεδίαση προσανατολισμένη ακριβώς στις απαιτήσεις του συγκεκριμένου προβλήματος, όμως το εξαιρετικά μειωμένο κόστος, η ταχύτητα ανάπτυξης καθώς και αυξημένη αξιοπιστία του συστήματος είναι ικανά να αντισταθμίσουν το μειονέκτημα αυτό.



Σχήμα 3.5 Η ανάπτυξη λογισμικού μέσω συναρμολόγησης

3.2.2 Σχεδίαση του υποσυστήματος του λογιστηρίου

3.2.2.1 Γενικά

Έχοντας μελετήσει τις σύγχρονες τεχνικές και προσεγγίσεις για τη σχεδίαση συστημάτων λογισμικού, είμαστε σε θέση να επιλέξουμε την κατάλληλη προσέγγιση, που θα οδηγήσει στην καλύτερη δυνατή κάλυψη των καθορισμένων απαιτήσεων. Αν και ο σχεδιασμός είναι μια διαδικασία που σε πολλά σημεία του δεν μπορεί να διαχωριστεί από την υλοποίηση, ο αρχικός σχεδιασμός έστω και σε μικρό επίπεδο λεπτομέρειας, έχει άμεση σχέση με τη προσπάθεια που θα χρειαστεί στο στάδιο της

υλοποίησης για την κάλυψη των απαιτήσεων. Μια κακή προσέγγιση στη σχεδίαση του συστήματος, μπορεί να οδηγήσει τόσο σε μια κακή ποιοτικά σχεδίαση, όσο και στη σπατάλη πόρων για την ολοκλήρωσή της σχεδίασης αυτής, αλλά της υλοποίησης αργότερα. Στη περίπτωση που εξετάζουμε, είναι γεγονός πως τόσο τα χρονικά πλαίσια, όσο και οι υπόλοιποι πόροι που διατίθενται για την ολοκλήρωση της προσπάθειας, είναι μάλλον περιορισμένοι για το μέγεθος του εγχειρήματος. Για το λόγο αυτό, έχει μεγάλη σημασία η επιλογή της προσέγγισης εκείνης, που θα οδηγήσει στη κάλυψη των απαιτήσεων με τη καλύτερη δυνατή διαχείριση των πόρων.

Λαμβάνοντας υπόψη λοιπόν τα παραπάνω, αυτό που φαίνεται να είναι η καταλληλότερη επιλογή για το σχεδιασμό του συστήματος, είναι μια προσέγγιση που θα βασίζεται ως ένα βαθμό στη συναρμολόγηση για την κάλυψη μερών της λειτουργικότητας. Στόχος είναι περισσότερο η χρησιμοποίηση εξαρτημάτων λογισμικού, που θα παρέχουν τις υπηρεσίες που απαιτούνται για την κάλυψη των δεδομένων απαιτήσεων, παρά η προσαρμογή των απαιτήσεων στο τι μπορούν να υλοποιήσουν να υπάρχοντα εξαρτήματα. Αποτέλεσμα είναι να καταλήγουμε σε μια ενδιάμεση λύση, που συνδυάζει στοιχεία από διάφορες προσεγγίσεις, όπως άλλωστε γίνεται και στις περισσότερες περιπτώσεις ανάπτυξης συστημάτων. Η χρήση δομημένων μεθόδων στα πλαίσια του σχεδιασμού, θεωρείται μάλλον πολυτέλεια, αφού οι πόροι που απαιτούνται για τη σωστή εφαρμογή τους είναι πολύ πιο απαραίτητοι σε άλλα στάδια της ανάπτυξης. Για το λόγο αυτό δε θα περιληφθούν τέτοιες μέθοδοι στη διαδικασία της σχεδίασης. Το γεγονός αυτό δεν είναι καθόλου ασυνήθιστο, αφού σπάνια κατά την ανάπτυξη εμπορικών εφαρμογών λογισμικού διατίθενται πόροι για την εφαρμογή δομημένων μεθόδων στο σχεδιασμό.

3.2.2.2 Αρχιτεκτονική του συστήματος

Μετά την απόφαση για τη γενικότερη σχεδιαστική προσέγγιση, πρώτο βήμα είναι ο καθορισμός του αρχιτεκτονικού σχεδίου της εφαρμογής. Η διαδικασία αυτή έχει να κάνει με την αναγνώριση και καθορισμό των υποσυστημάτων και του τρόπου αλληλεπίδρασής τους. Σκοπός στη φάση αυτή είναι η απόφαση για το τι ακριβώς θα αποτελέσει ανεξάρτητο υποσύστημα και γενικότερα πως θα κατανεμηθεί η κάλυψη των απαιτήσεων μεταξύ των μονάδων του συστήματος. Με άλλα λόγια, αυτό που γίνεται στο σημείο αυτό, είναι ο καθορισμός των ανεξάρτητων μερών συστήματος, ο καταμερισμός της λειτουργικότητας ανάμεσα στα μέρη αυτά και ο τρόπος που όλα

αυτά τα μέρη θα ενοποιηθούν για να αποτελέσουν το τελικό σύστημα. Αυτό ακριβώς το κομμάτι του σχεδιασμού είναι που παίζει το μεγαλύτερο ρόλο στη μείωση της προσπάθειας που θα απαιτηθεί για την υλοποίηση. Μια σωστή και ευέλικτη αρχιτεκτονική δεν απαιτεί τη χρήση δύσκολων και χρονοβόρων προγραμματιστικών τεχνικών για την υλοποίησή της και για το λόγο αυτό προσφέρει το καλύτερο δυνατό καταμερισμό των πόρων ανάμεσα στα επιμέρους διαδικασίες της ανάπτυξης.

Με μια πρώτη ματιά στο καθορισμό του έργου γίνεται φανερό ότι κορμό του συστήματος θα αποτελέσουν οι οντότητες και οι λειτουργίες, που έχουν να κάνουν με τη ίδια τη λογιστική. Βάση του συστήματος, επομένως, θα αποτελέσει η διαχείριση του λογιστικού σχεδίου και οι κινήσεις των λογαριασμών, μαζί φυσικά με οτιδήποτε άλλο σχετίζεται με αυτά. Αυτό που στη πραγματικότητα θα περιλαμβάνει το υποσύστημα αυτό, είναι όλες οι απαιτούμενες λειτουργίες για τη διαχείριση των οντοτήτων της λογιστικής, δηλαδή η εισαγωγή, η τροποποίηση, η διαγραφή και η ανάκτηση των δεδομένων, καθώς επίσης και την υλοποίηση όλων των περιορισμών που διέπει τη διαχείρισή τους. Ιδιαίτερα χαρακτηριστικά του υποσυστήματος αυτού, που θα πρέπει να δοθεί μεγάλη προσοχή κατά το λεπτομερέστερο σχεδιασμό, είναι οι επιδόσεις και ο σχεδιασμός του γραφικού περιβάλλοντος. Αυτό οφείλεται στο γεγονός ότι υποσύστημα αυτό είναι το κομμάτι της εφαρμογής, το οποίο θα κληθεί να διαχειριστεί το μεγαλύτερο όγκο δεδομένων, με ότι συνεπάγεται αυτό τόσο για την επιβάρυνση του ίδιου του υποσυστήματος, όσο και την επιβάρυνση των χρηστών του.

Εκτός από το κεντρικό υποσύστημα της εφαρμογής, που θα αποτελείται από τις λειτουργίες της λογιστικής, ένα άλλο κομμάτι των λειτουργικών απαιτήσεων που θα πρέπει να αποτελέσει ανεξάρτητο υποσύστημα, βάσει των καθορισμένων απαιτήσεων, είναι η διαχείριση των προβολών. Είναι σαφές, ότι η συχνότητα εμφάνισης των προβολών και η απαίτηση για συχνή προσαρμογή τους, δεν επιτρέπει σε καμία περίπτωση την ενσωμάτωση της λειτουργίας αυτής στο υποσύστημα της λογιστικής. Επίσης, το γεγονός ότι ένας ανεξάρτητος μηχανισμός διαχείρισης των προβολών θα μπορούσε να χρησιμοποιηθεί για να καλύψει τις ανάγκες ολόκληρου του συστήματος της τραπέζης, συμβάλλει σημαντικά στην απόφαση αυτή. Αυτό που θα δημιουργηθεί, είναι μια μονάδα ανεξάρτητη της συγκεκριμένης προβολής, αλλά και του αντικειμένου που ζητάει τη προβολή, η οποία θα μπορεί να κληθεί σε οποιοδήποτε σημείο του συστήματος και να εμφανίσει δεδομένα στην οθόνη με μορφή ελεγχόμενη, χωρίς την επέμβαση στο πηγαίο κώδικα.

Κάτι ακόμα που πρέπει να αποφασιστεί στα πλαίσια του αρχιτεκτονικού σχεδιασμού της εφαρμογής, είναι το κατά πόσο η κάλυψη των υπολοίπων απαιτήσεων της εφαρμογής απαιτεί την ανάπτυξη ξεχωριστών υποσυστημάτων. Συγκεκριμένα, οι δύο απαιτήσεις που είναι υποψήφιες για την ανάπτυξη ανεξάρτητων μονάδων, είναι η διαχείριση των μηνυμάτων, που περιλαμβάνει και την υποστήριξη πολλών γλωσσών, και η διαχείριση των χρηστών της εφαρμογής. Και στις δύο περιπτώσεις είναι φανερό ότι θα πρέπει να υπάρξει υποστήριξη των λειτουργιών αυτών σε επίπεδο βάσης δεδομένων, επομένως το πρόβλημα περιορίζεται στο κατά απαιτείται ξεχωριστό υποσύστημα σε επίπεδο λογισμικού. Μελετώντας τις λειτουργίες που απαιτούνται για τη κάλυψη των δύο απαιτήσεων και λαμβάνοντας υπόψη ότι στόχος της προσπάθειας δεν είναι η κάλυψη των αναγκών ολόκληρου του συστήματος της τράπεζας, κρίνεται πως η ανάπτυξη ξεχωριστών υποσυστημάτων για τις δύο αυτές απαιτήσεις είναι μάλλον έξω από τους στόχους της προσπάθειας, τουλάχιστον σε πρώτη φάση. Αρχικά αυτό που θα γίνει, είναι η υποστήριξη των λειτουργιών αυτών από τη βάση δεδομένων και η ενσωμάτωση στο σύστημα της λογιστικής της δυνατότητας προσαρμογής των μηνυμάτων και αναγνώρισης των χρηστών.

Συνοψίζοντας λοιπόν την αρχιτεκτονική σχεδίαση του συστήματος, βλέπουμε πως αποτελείται από δύο ανεξάρτητα υποσυστήματα. Το πρώτο από αυτά αποτελεί τη βάση του νέου συστήματος και καλύπτει τις λειτουργικές απαιτήσεις της λογιστικής, εκτός από τις προβολές των δεδομένων και το δεύτερο θα καλύψει τις αυξημένες ανάγκες για προβολή των δεδομένων στην οθόνη, αλλά και για επεξεργασία πάνω στα εμφανιζόμενα δεδομένα. Τέλος, η κάλυψη των υπόλοιπων απαιτήσεων θα ολοκληρωθεί, ενσωματώνοντας στο υποσύστημα της λογιστικής τη λειτουργικότητα που απαιτείται. Επόμενος στόχος μετά την ολοκλήρωση των παραπάνω και στα πλαίσια της ανάπτυξης ολόκληρου του συστήματος της τραπέζης, θα πρέπει να είναι η υλοποίηση ανεξάρτητων υποσυστημάτων για τη διαχείριση των λειτουργιών για τις οποίες στα πλαίσια της δικής μας προσπάθειας υλοποιήθηκε μόνο η πλευρά που αφορά τη λογιστική.

3.2.2.3 Έρευνα για εξαρτήματα λογισμικού και καθορισμός των εξαρτημάτων που θα αναπτυχθούν

Μια από τις πιο καθοριστικές επιλογές έχουν γίνει, σχετικά με το σχεδιασμό του συστήματος, είναι η υιοθέτηση της ανάπτυξης με τη βοήθεια εξαρτημάτων λογισμικού. Το πρώτο βήμα, μετά το καθορισμό των απαιτήσεων, για την εφαρμογή μιας τέτοιας προσέγγισης, είναι η έρευνα για τα εξαρτήματα λογισμικού που είναι διαθέσιμα και μπορούν να παρέχουν υπηρεσίες στο σύστημα. Αυτό που θα πρέπει να γίνει είναι μια προσεκτική μελέτη για το τι εξαρτήματα λογισμικού υπάρχουν, τι υπηρεσίες μπορούν να προσφέρουν και το κατά πόσο και με ποιο τρόπο μπορούν για την ανάπτυξη του συστήματος. Η σημασία της μελέτης αυτής για τη πορεία της ανάπτυξης είναι καθοριστική, αφού θα ήταν αδύνατο, δεδομένων των διαθέσιμων πόρων, να αναπτυχθεί στα πλαίσια της προσπάθειας οτιδήποτε απαιτείται για την υλοποίηση του συστήματος.

Οι λειτουργίες εκείνες του συστήματος που προσφέρονται για τη χρήση εξαρτημάτων λογισμικού είναι δύο: η δημιουργία των διεπαφών και γενικότερα ότι έχει σχέση με το γραφικό περιβάλλον της εφαρμογής και η επικοινωνία με τη βάση δεδομένων. Οι λόγοι που οι δύο αυτές ομάδες λειτουργιών κρίνονται ως πιο κατάλληλες είναι αρκετοί. Πιο σημαντικοί από αυτούς είναι η δυσκολία υλοποίησης των λειτουργιών αυτών χωρίς τη χρήση εξειδικευμένων εξαρτημάτων και το μεγάλο βαθμό ανεξαρτησίας τους από τη λογική του συστήματος. Είναι μάλλον ευνόητο, πως η προσπάθεια δημιουργίας γραφικού περιβάλλοντος ή η προσπάθεια χειρισμού της επικοινωνίας με τη βάση δεδομένων από το μηδέν, είναι μια διαδικασία που παρουσιάζει όχι μόνο προγραμματιστικές δυσκολίες και μεγάλη εξειδίκευση στους συγκεκριμένους τομείς, αλλά και χρονικά περιθώρια που δεν είναι διαθέσιμα. Παράλληλα, οι δύο αυτές λειτουργίες μπορούν εύκολα να διαχωριστούν από τη λογική του συστήματος, αφού είναι δυνατό να αντιμετωπιστούν σαν γενικής φύσης προβλήματα. Για παράδειγμα, όταν υπάρχει ανάγκη για πρόσβαση στη βάση δεδομένων, το μόνο που έχει σχέση με τη λογική της εφαρμογής είναι το ερώτημα που θα εκτελεστεί, ενώ όλα τα άλλα είναι κοινά σε οποιαδήποτε προσπάθεια πρόσβασης. Αυτό σημαίνει, ότι εύκολα θα μπορούσε κανείς να δεχτεί τις υπηρεσίες αυτές από ένα εξάρτημα, που θα ρυθμίζει όλες τις λεπτομέρειες της επικοινωνίας και θα εκτελεί οποιοδήποτε ερώτημα του δοθεί σαν είσοδος.

Εξετάζοντας λοιπόν τη διαθεσιμότητα εξαρτημάτων λογισμικού για τους δύο παραπάνω σκοπούς, διαπιστώνει κανείς, ότι οι περισσότερες ανάγκες καλύπτονται από όλα σχεδόν τα σύγχρονα περιβάλλοντα ανάπτυξης λογισμικού. Πιο συγκεκριμένα, για το θέμα της πρόσβασης στη βάση δεδομένων, υπάρχουν μια σειρά από τεχνολογίες και τα αντίστοιχα εξαρτήματα, που παρέχουν υπηρεσίες σε πολύ πιο υψηλό επίπεδο από αυτό που απαιτεί το υποσύστημα του λογιστηρίου. Επομένως κρίνεται πως τα υπάρχοντα εξαρτήματα καλύπτουν με το παραπάνω τις ανάγκες του νέου συστήματος στο τομέα αυτό. Όσον αφορά το θέμα της δημιουργίας του γραφικού περιβάλλοντος, υπάρχει ένα πολύ μεγάλο σύνολο τεχνολογιών και εξαρτημάτων, που καλύπτουν τη συντριπτική πλειοψηφία των απαιτήσεων στο τομέα αυτό. Το μόνο πρόβλημα που παρουσιάζεται, είναι ότι τα υπάρχοντα εξαρτήματα είναι γενικής χρήσης και δεν είναι σε θέση να καλύψουν ακριβώς, ορισμένους περιορισμούς του περιβάλλοντος της λογιστικής και πιο συγκεκριμένα την εισαγωγή και εμφάνιση των λογαριασμών της λογιστικής και την εισαγωγή και εμφάνιση αριθμών ελεγχόμενης ακρίβειας σε δεκαδικά ψηφία.

Η παραπάνω διαπίστωση μας οδηγεί στο συμπέρασμα, ότι υπάρχει η ανάγκη για δημιουργία ή, αν αυτό είναι δυνατό, προσαρμογή των υπαρχόντων εξαρτημάτων λογισμικού. Το γεγονός αυτό δεν είναι καθόλου ασυνήθιστο στην ανάπτυξη λογισμικού μέσω συναρμολόγησης, αφού ένα σημαντικό κομμάτι της ανάπτυξης με τη προσέγγιση αυτή βασίζεται στη προσαρμογή υπάρχοντος λογισμικού. Βασικός παράγοντας για τη δημιουργία των εξαρτημάτων είναι ο σωστός καθορισμός των αναγκών, που μπορεί εύκολα να γίνει εκτιμώντας τους λόγους για τους οποίους τα υπάρχοντα εξαρτήματα κρίνονται ανεπαρκή για τις ανάγκες. Έχοντας καθορίσει το τι ακριβώς παραπάνω θα πρέπει να έχει το νέο εξάρτημα, μπορεί εύκολα να γίνει ένας αρχικός σχεδιασμός, που όμως θα τροποποιηθεί κατά την υλοποίηση ανάλογα με το εργαλείο ανάπτυξης.

Πρώτη περίπτωση, όπου τα υπάρχοντα εξαρτήματα δεν καλύπτουν το σύνολο των αναγκών της εφαρμογής, είναι η διαχείριση, σε επίπεδο διεπαφών, των λογαριασμών λογιστικής. Οι λογαριασμοί στη λογιστική έχουν πέντε επίπεδα και ιεραρχική δομή. Για τα τέσσερα πρώτα επίπεδα χρησιμοποιούνται δύο μόνο ψηφία ενώ για το τελευταίο πέντε (π.χ. 30.00.00.00.00001). Ιεραρχική δομή σημαίνει, ότι ένας λογαριασμός έχει σαν υπολογαριασμούς, όλους τους λογαριασμούς μεγαλύτερου επιπέδου, που έχουν το λογαριασμό αυτό σαν πρόθεμα (π.χ. ο 30.00 και ο 30.00.00.00.00001 είναι υπολογαριασμοί του 30). Οι ανάγκες που υπάρχουν όσον

αφορά την εισαγωγή και τη γραφική απεικόνιση των λογαριασμών, εξάγονται από τους παραπάνω περιορισμούς και έχουν να κάνουν με την διασφάλιση της σωστής εισαγωγής του λογαριασμού. Αυτό που βασικά μας ενδιαφέρει είναι να εξασφαλίζεται, ότι αυτό που εισάγει ο χρήστης είναι έγκυρο ως προς τη μορφή του και υπακούει στους κανόνες της ιεραρχίας του λογιστικού σχεδίου.

Έχοντας πλέον αναγνωρίσει το τι ακριβώς είναι το ζητούμενο στη πρώτη αυτή περίπτωση, επόμενο βήμα είναι η έρευνα για εξαρτήματα που θα μπορούσαν να αποτελέσουν βάση για το εξειδικευμένο εξάρτημα. Συμπέρασμα της έρευνας αυτής είναι ότι υπάρχουν εξαρτήματα στα περισσότερα περιβάλλοντα ανάπτυξης, που μπορούν να εξασφαλίσουν ότι αυτό που θα εισαχθεί θα είναι σύμφωνο με μια συγκεκριμένη μορφή που μπορεί να οριστεί (π.χ. μπορεί να απαιτηθεί στη πρώτη θέση να εισαχθεί αλφαριθμητικό, στη δεύτερη να υπάρχει μια τελεία και στη τρίτη αριθμός). Η δυνατότητα αυτή μπορεί μεν να εξασφαλίσει ότι δεν θα εισαχθούν μη αποδεκτοί χαρακτήρες στο λογαριασμό, αλλά δεν μπορεί από μόνη της να εξασφαλίσει τη λογική συνέπεια του λογαριασμού (π.χ. η είσοδος 30. .00. .00001 είναι αποδεκτή, αφού δεν περιλαμβάνει μη επιτρεπτούς χαρακτήρες, όμως φυσικά δεν αποτελεί λογαριασμό). Επίσης, με ένα τέτοιο εξάρτημα δεν μπορεί σε καμία περίπτωση να εξασφαλιστεί η τήρηση της ιεραρχίας του λογιστικού σχεδίου.

Η λύση που προτείνεται για τα παραπάνω προβλήματα είναι η δημιουργία ενός νέου εξαρτήματος, που θα κληρονομεί τη λειτουργικότητα από ένα από τα υπάρχοντα εξαρτήματα που αναφέρθηκαν παραπάνω. Το κέρδος από μια τέτοια απόφαση είναι ότι δεν χρειάζεται σε καμία περίπτωση η υλοποίηση λειτουργικότητας, που ήδη καλύπτεται από τα υπάρχοντα εξαρτήματα, επικεντρώνοντας έτσι τη προσπάθεια μόνο στην ικανοποίηση των δικών μας περιορισμών. Επίσης, εξασφαλίζεται ένα πολύ καλό επίπεδο αξιοπιστίας αφού το νέο εξάρτημα βασίζεται σε ένα δοκιμασμένο προϊόν που ήδη κυκλοφορεί στην αγορά. Επιπλέον, ο τρόπος που το νέο εξάρτημα δανείζεται τη λειτουργικότητα, είναι σύμφωνος με τη λογική της αντικειμενοστραφούς σχεδίασης και απολαμβάνει επομένως όλα τα πλεονεκτήματά της. Οι προσθήκες, τέλος, που θα πρέπει να γίνουν στο νέο εξάρτημα, είναι μάλλον προφανείς και αφορούν την υλοποίηση των περιορισμών που θα εξασφαλίζουν τη λογική συνέπεια του εισαγόμενου λογαριασμού και την τήρηση της ιεραρχίας του λογιστικού σχεδίου. Επειδή μια λεπτομερέστερη σχεδίαση θα απαιτούσε τη γνώση των δυνατοτήτων του συγκεκριμένου περιβάλλοντος, στο οποίο θα γίνει η ανάπτυξη,

η λεπτομερής σχεδίαση του νέου αυτού εξαρτήματος θα ολοκληρωθεί στα πλαίσια της υλοποίησης του συστήματος.

Η δεύτερη και τελευταία περίπτωση, όπου κρίνεται αναγκαία η δημιουργία νέου εξαρτήματος λογισμικού, είναι η ανάγκη για προβολή και εισαγωγή αριθμών με ελεγχόμενη ακρίβεια στα δεκαδικά ψηφία. Η ανάγκη αυτή προέρχεται, στη περίπτωση της λογιστικής, από την ανάγκη για εισαγωγή και προβολή χρηματικών ποσών, που έχουν έως δύο δεκαδικά ψηφία. Ακολουθώντας την ίδια πορεία με την προηγούμενη περίπτωση, η έρευνα δείχνει ότι υπάρχει ένας αριθμός εξαρτημάτων που μπορεί να εξασφαλίσει την εισαγωγή μόνο αριθμών από το χρήστη. Αυτό που μένει επομένως να υλοποιηθεί στα πλαίσια της ανάπτυξης του νέου εξαρτήματος, είναι ο έλεγχος των δεκαδικών ψηφίων του αριθμού που εισάγει ο χρήστης. Και στη περίπτωση αυτή, είναι σαφές, ότι λεπτομερέστερη σχεδίαση του νέου εξαρτήματος δεν είναι εφικτό να υπάρξει πριν το στάδιο της υλοποίησης.

3.2.2.4 Σχεδιασμός των υποσυστημάτων

3.2.2.4.1 Κοινή λειτουργικότητα των υποσυστημάτων και η διεπαφή – πρότυπο

Αφού έχει λοιπόν εξασφαλιστεί, ότι τα απαραίτητα για την ανάπτυξη του συστήματος συστατικά μέρη είναι διαθέσιμα, επόμενο βήμα είναι ο λεπτομερέστερος σχεδιασμός των υποσυστημάτων που καθορίστηκαν κατά τον αρχιτεκτονικό σχεδιασμό. Αυτό που θα πρέπει να γίνει σε αυτή τη φάση, είναι μια πρώτη προσέγγιση του πως θα υλοποιηθούν οι καθορισμένες απαιτήσεις για κάθε υποσύστημα. Βέβαια, όπως έχει ήδη αναφερθεί, ο σχεδιασμός και ιδιαίτερα ο σχεδιασμός σε μεγάλο επίπεδο λεπτομέρειας, είναι μια διαδικασία που επικαλύπτεται σε μεγάλο βαθμό με το στάδιο της υλοποίησης. Είναι επόμενο λοιπόν, ότι οι γενικές κατευθύνσεις που θα δοθούν στο στάδιο αυτό, θα πάρουν τη μορφή της τελικής σχεδίασης κατά το στάδιο της υλοποίησης, όπου θα γίνει φανερό το τι ακριβώς μπορεί να φτιαχτεί.

Το πρώτο πράγμα που θα πρέπει να καθοριστεί πριν από οτιδήποτε άλλο, είναι η δομή του συστήματος. Δομή του συστήματος, στη δική μας περίπτωση, εννοείται ο τρόπος με τον οποίο θα ρέουν τα δεδομένα μέσα στην εφαρμογή. Η δομή αυτή έχει να κάνει με τα επίπεδα της εφαρμογής και το βαθμό ανεξαρτησίας των διεπαφών από τη λογική του συστήματος. Μια σημαντική παρατήρηση, που έχει καθοριστικό ρόλο

στην απόφαση αυτή, είναι η φιλοσοφία με την οποία είναι σχεδιασμένες οι διεπαφές του παλιού συστήματος. Μετά από μελέτη των διεπαφών αυτών, το συμπέρασμα που βγαίνει είναι ότι στόχος του σχεδιαστή είναι να οδηγήσει το χρήστη στο σωστό και όχι να χειριστεί όλα τα πιθανά λάθη του. Η λογική αυτή της πρόληψης των λαθών και όχι χειρισμού τους, αναπόφευκτα μεταφέρει την υλοποίηση των περιορισμών και γενικότερα της λογικής του συστήματος στις διεπαφές. Είναι φανερό ότι για να σχεδιαστεί μια διεπαφή, που ουσιαστικά δεν θα επιτρέπει στο χρήστη να κάνει κάτι που να έρχεται σε αντίθεση με τη λογική και τους περιορισμούς του συστήματος, θα πρέπει τους περιορισμούς αυτούς να τους υλοποιήσει και να τους διαχειριστεί η ίδια η διεπαφή. Το γεγονός αυτό οδηγεί στο συμπέρασμα, ότι δεν κρίνεται σκόπιμος ο διαχωρισμός της λογικής του συστήματος από τις διεπαφές και επομένως στοιχειώδης μονάδα λογισμικού της εφαρμογής θα αποτελεί η ίδια η διεπαφή.

Συνεχίζοντας τη διαδικασία καθορισμού των λεπτομερειών της σχεδίασης, είναι σκόπιμο να γίνει η αρχή από τις γενικές λεπτομέρειες, που αφορούν το σύνολο του συστήματος. Η απόφαση να συμπεριληφθούν οι δυνατότητες για πολυγλωσσική υποστήριξη και για διαχείριση όλων των μηνυμάτων που εμφανίζονται στην οθόνη, στη λειτουργικότητα των υπόλοιπων δύο υποσυστημάτων, δημιουργεί την ανάγκη να ενσωματωθούν οι δυνατότητες αυτές στη λειτουργικότητα των διεπαφών τους και συνεπώς στις διεπαφές ολόκληρου του συστήματος. Με άλλα λόγια, αυτό που είναι το ζητούμενο, είναι η συμπερίληψη κάποιων κοινών λειτουργιών σε όλες τις διεπαφές του συστήματος. Οι λειτουργίες αυτές γίνονται περισσότερες, αν συνυπολογιστούν και τα άλλα κοινά σημεία, όπως για παράδειγμα η πρόσβαση στη βάση δεδομένων.

Η κοινή αυτή λειτουργικότητα θα μπορούσε να υλοποιηθεί ξεχωριστά σε κάθε διεπαφή, με το ανάλογο φυσικά κόστος σε κόπο, χρόνο αλλά και ευκολία συντήρησης της εφαρμογής λόγω του επαναλαμβανόμενου κώδικα. Είναι φανερό, ότι η καλύτερη λύση είναι η υλοποίηση της κοινής αυτής λειτουργικότητας μία μόνο φορά και η παροχή της δυνατότητας σε όλες τις διεπαφές, να τη χρησιμοποιούν. Ο βέλτιστος τρόπος για να πραγματοποιηθεί, είναι η χρησιμοποίηση της κληρονομικότητας σε επίπεδο διεπαφών. Αυτό που μπορεί να γίνει, είναι να δημιουργηθεί ένα αντικείμενο το οποίο θα κληρονομεί τις ιδιότητες μιας διεπαφής από το περιβάλλον λογισμικού και θα υλοποιεί στη συνέχεια τη κοινή λειτουργικότητα που απαιτείται. Στη συνέχεια, όλες οι διεπαφές της εφαρμογής θα κληρονομούν από την διεπαφή – πρότυπο και έτσι θα έχουν διαθέσιμες όλες τις λειτουργίες που είναι υλοποιημένες σε αυτή. Τα πλεονεκτήματα, βέβαια, της επιλογής αυτής είναι μάλλον προφανή σε όλα τα

επίπεδα, αφού αποφεύγεται έτσι η πολλαπλή υλοποίηση των ίδιων λειτουργιών, ενώ για μία ακόμα φορά γίνεται εκμετάλλευση όλων των πλεονεκτημάτων μιας αντικειμενοστραφούς σχεδίασης

Επόμενο βήμα, μετά και το καθορισμό του γενικού πλαισίου για τη κάλυψη των κοινών για όλη την εφαρμογή αναγκών, είναι η σχεδίαση του μηχανισμού που θα περιληφθεί στη διεπαφή – πρότυπο. Για να γίνει αυτό, θα πρέπει πρώτα να καθοριστεί το τι ακριβώς είναι αυτό που πρέπει να είναι παραμετρικό σχετικά με ότι εμφανίζεται σε μια διεπαφή. Μια πρώτη ανάλυση στις ήδη υπάρχουσες διεπαφές, δείχνει ότι οι στατικές περιγραφές, που εμφανίζονται στην οθόνη, χωρίζονται σε τρεις ομάδες:

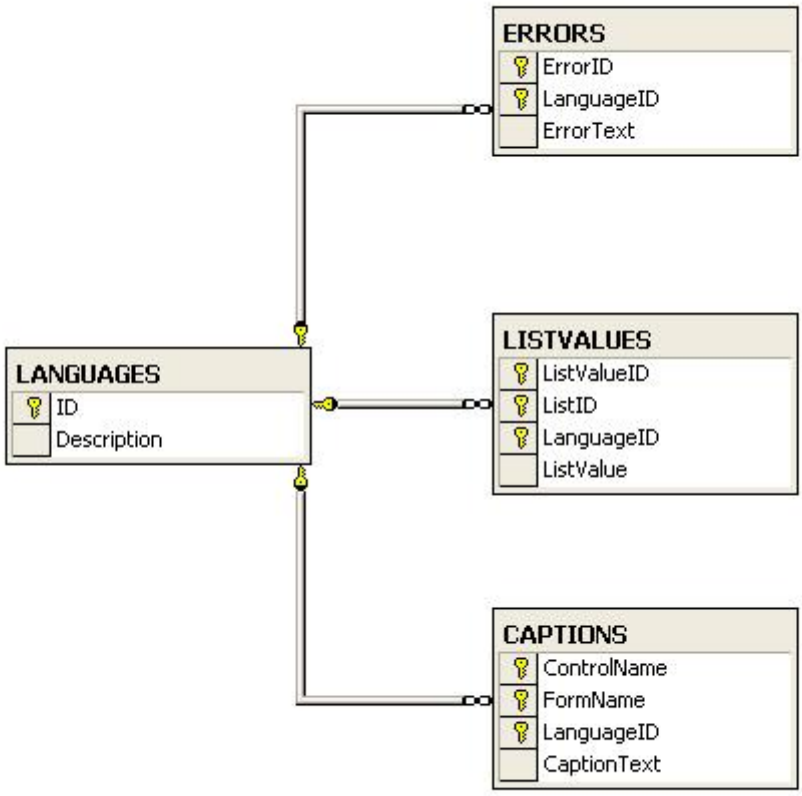
- τα μηνύματα που εμφανίζονται είτε λόγω λάθους, είτε για οποιοδήποτε άλλο λόγο,
- τις στατικές περιγραφές που βρίσκονται πάνω σε μια διεπαφή και έχουν σαν στόχο να γνωστοποιήσουν στο χρήστη το τι ακριβώς αντιπροσωπεύει κάθε αντικείμενο της διεπαφής και
- τις περιγραφές των προεπιλογών στα σημεία που ο χρήστης καλείται να επιλέξει κάτι από ένα στατικό σύνολο επιλογών.

Αυτό λοιπόν που πρέπει να παρέχει ο μηχανισμός που θα σχεδιαστεί, είναι τη δυνατότητα καθορισμού των παραπάνω περιγραφών, χωρίς την επέμβαση στο πηγαίο κώδικα της εφαρμογής. Επίσης, η δυνατότητα υποστήριξης πολλών γλωσσών, επιβάλλει την ταυτόχρονη ύπαρξη περισσότερων από μίας περιγραφών για τον ίδιο σκοπό, σε διαφορετικές γλώσσες.

Η ανάγκη για παραμετροποίηση των περιγραφών που εμφανίζονται στις γραφικές διεπαφές του συστήματος, ώστε να μην είναι αναγκαία η επέμβαση στο πηγαίο κώδικα, επιβάλλει τη χρήση της βάσης δεδομένων για τη διαχείρισή τους. Ο ρόλος που θα παίξει η βάση δεδομένων, είναι να αποθηκεύσει τις περιγραφές και να τις παρέχει στο μηχανισμό όταν ζητηθούν. Με τον τρόπο αυτό είναι φανερό, πως για να αλλάξει κάποια περιγραφή αρκεί να αλλάξει κάποια εγγραφή στη βάση δεδομένων. Αυτό που μένει, λοιπόν, να γίνει για την ολοκλήρωση της σχεδίασης του μηχανισμού αυτού, είναι ο καθορισμός της δομής που θα έχει η σχετική βάση δεδομένων, καθώς και ο τρόπος με τον οποίο θα ανακτώνται οι περιγραφές σε επίπεδο λογισμικού.

Πρώτο βήμα για το καθορισμό της δομής που θα έχει η βάση δεδομένων, είναι η απόφαση για το τι ακριβώς θα πρέπει να αποθηκεύεται, για κάθε μια από τρεις

ομάδες περιγραφών που αναφέρθηκαν. Ξεκινώντας από τα μηνύματα, εξάγεται το συμπέρασμα, ότι για να καθοριστεί μοναδικά η περιγραφή ενός μηνύματος χρειάζεται ένας κωδικός που θα καθορίζει το μήνυμα και φυσικά ο κωδικός που δηλώνει τη γλώσσα στην οποία αναφέρεται αφού το σύστημα υποστηρίζει πολλές γλώσσες ταυτόχρονα. Αντίστοιχα, στη περίπτωση των στατικών περιγραφών πάνω στις γραφικές διεπαφές, αυτό που απαιτείται είναι ένας κωδικός που θα καθορίζει τη διεπαφή στην οποία αναφέρεται η περιγραφή, ένα κωδικό που να δηλώνει σε ποιο αντικείμενο της διεπαφής και φυσικά το κωδικό της γλώσσας. Τέλος, για τη περίπτωση των περιεχομένων σε λίστες, όπου υπάρχουν, αυτό που χρειάζεται είναι ένας κωδικός για τη λίστα που αναφέρεται, ένας κωδικός για το αντικείμενο της λίστας που αντιστοιχεί και το κωδικό της γλώσσας. Λαμβάνοντας υπόψη όλα τα παραπάνω, καταλήξαμε στο σχήμα βάσης του σχήματος 3.6.



Σχήμα 3.6 Το σχήμα βάσης για τη διαχείριση των περιγραφών

Έχοντας καθορίσει μια δομή για τη βάση, αυτό που μένει είναι να σχεδιαστεί, σε γενικές γραμμές, ο μηχανισμός ανάκτησης των περιγραφών, που θα περιληφθεί

στη διεπαφή – πρότυπο. Το κομμάτι της ανάκτησης των δεδομένων στο μηχανισμό, μπορεί να καλυφθεί πλήρως, με τη χρήση εξαρτημάτων λογισμικού που προσφέρουν πρόσβαση σε βάσεις δεδομένων. Το μόνο που θα πρέπει να γίνει από τη διεπαφή – πρότυπο, είναι η ανάκτηση, κατά τη δημιουργία της, όλων των περιγραφών, που ανήκουν σε αυτή (από τον πίνακα *Captions*) και στη συνέχεια η εκχώρηση στα αντικείμενά της, των περιγραφών που τους αντιστοιχούν. Όσον αφορά τις λίστες, θα πρέπει για κάθε αντικείμενο λίστας, να ανακτά τις τιμές από τη βάση (πίνακας *ListValues*) και να τις εκχωρεί στη λίστα. Τέλος, όσον αφορά τα μηνύματα, τα πράγματα είναι λίγο διαφορετικά. Αυτό που μπορεί να κάνει το αντικείμενο της διεπαφής – πρότυπο, είναι να παρέχει μια συνάρτηση που θα μπορεί να χρησιμοποιηθεί από τις διεπαφές οι οποίες κληρονομούν από αυτή και θα ανακτά τη περιγραφή ενός μηνύματος δίνοντας το κωδικό του. Η ιδιαιτερότητα αυτή στη περίπτωση των μηνυμάτων, δημιουργείται από το γεγονός ότι τα μηνύματα δεν ανήκουν στατικά σε μια διεπαφή, αλλά δημιουργούνται δυναμικά μέσα στο κώδικα της εφαρμογής, όταν είναι απαραίτητο.

Ένα ακόμα θέμα που αφορά τη γενική λειτουργία της εφαρμογής και θα πρέπει να ρυθμιστεί είναι η διαχείριση των χρηστών στο υποσύστημα του λογιστηρίου. Η διαχείριση των χρηστών από τη διεπαφή – πρότυπο δεν προσφέρει σε καμία περίπτωση ευχρηστία, αφού θα πρέπει κάθε φορά που φορτώνεται μια διεπαφή να ανακτώνται και τα στοιχεία του χρήστη, που όμως θα είναι πάντα τα ίδια. Η καλύτερη λύση σε τέτοιες περιπτώσεις, είναι τα στοιχεία αυτά να μένουν σε καθολικές μεταβλητές της εφαρμογής, έτσι ώστε οποιαδήποτε διεπαφή να είναι σε θέση να χρησιμοποιήσει τις πληροφορίες αυτές. Με τον τρόπο αυτό επιτυγχάνεται ταυτόχρονα και η βέλτιστη δυνατή χρήση των υπολογιστικών πόρων του συστήματος, αφού τα δεδομένα του χρήστη ανακτώνται μόνο μια φορά κατά την εισαγωγή του, ενώ είναι διαθέσιμα σε όλες τις διεπαφές του συστήματος.

3.2.2.4.2 Το υποσύστημα της λογιστικής

Μετά και τη σχεδίαση του γενικού πλαισίου του συστήματος, σειρά έχει η λεπτομερής σχεδίαση των δύο υποσυστημάτων. Πρώτο από τα υποσυστήματα που θα σχεδιαστεί, είναι το υποσύστημα που αφορά τη λειτουργικότητα της λογιστικής. Τα θέματα τα οποία θα πρέπει να ρυθμιστούν κατά το στάδιο του σχεδιασμού για το υποσύστημα αυτό, χωρίζονται σε τρεις ομάδες:

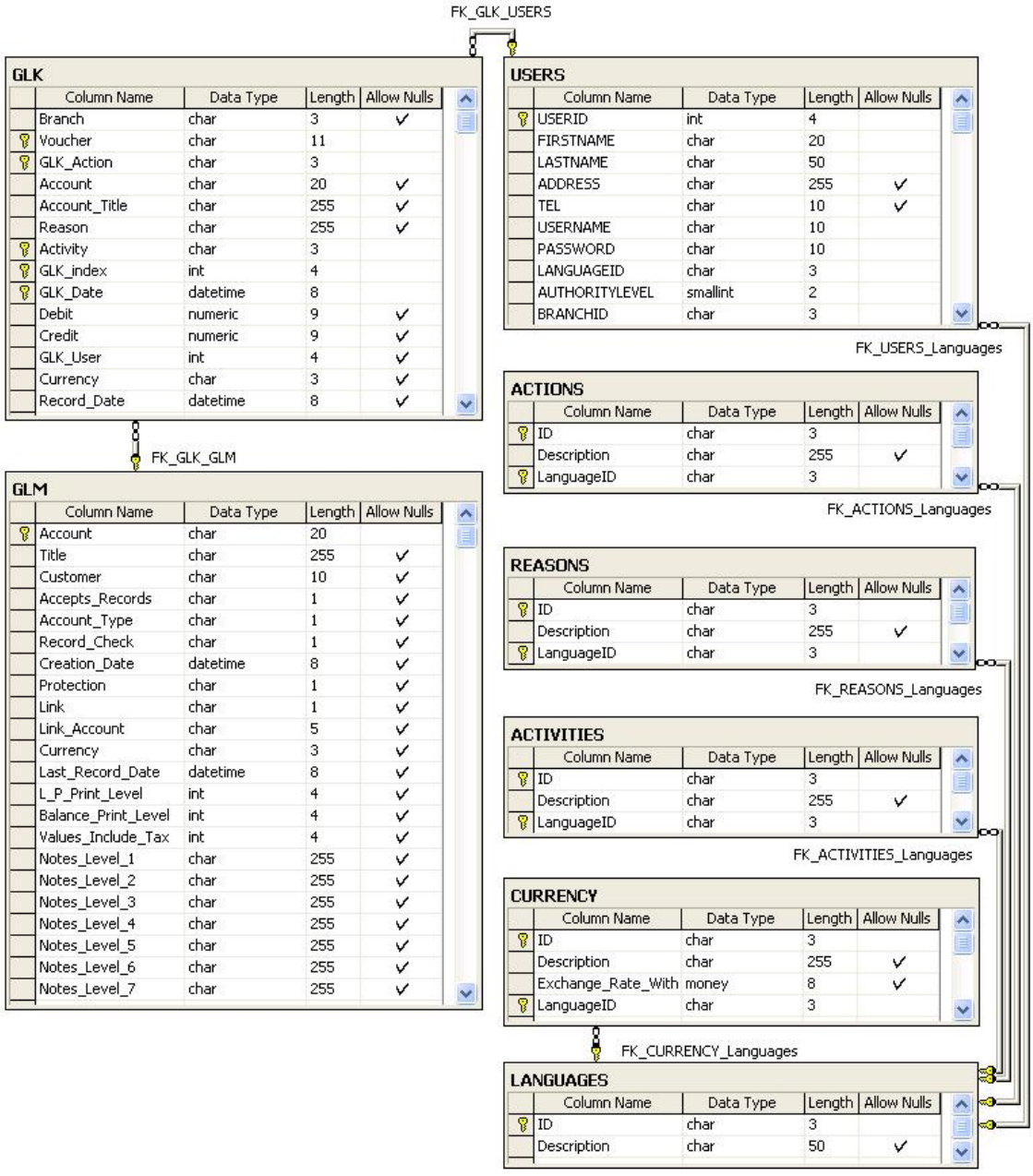
- η σχεδίαση της βάσης δεδομένων του υποσυστήματος,
- η σχεδίαση των διεπαφών του υποσυστήματος και
- η σχεδίαση των μηχανισμών που θα εξασφαλίζουν την ικανοποίηση των περιορισμών που θέτει η λογιστική.

Έχοντας ολοκληρώσει μια αρχική σχεδίαση των παραπάνω, θα είμαστε σε θέση να προχωρήσουμε στην υλοποίηση του συγκεκριμένου υποσυστήματος, οπότε και η αρχική αυτή σχεδίαση, θα πάρει και τη τελική της μορφή.

Από τα πιο σημαντικά κομμάτια του σχεδιασμού ενός συστήματος είναι ο σχεδιασμός της βάσης δεδομένων. Το μέρος αυτό του σχεδιασμού, είναι ίσως το μέρος που θα δεχτεί τις λιγότερες αλλαγές κατά την υλοποίηση, αφού επιδίωξη είναι να γίνει από την αρχή μια σχεδίαση που να μπορεί να καλύψει τις ανάγκες πετυχαίνοντας παράλληλα και τη μέγιστη δυνατή απόδοση του συστήματος. Θεμέλιο για τη σχεδίαση της βάσης δεδομένων είναι οι ιδιότητες των βασικών οντοτήτων της λογιστικής, μαζί αυτές των βοηθητικών οντοτήτων. Μελετώντας τα παραπάνω, και έχοντας πάντα υπόψη την ανάγκη για βέλτιστη δυνατή απόδοση, εξάγεται η δομή της βάσης που φαίνεται στο σχήμα 3.7. Είναι εμφανής η προσπάθεια που έχει γίνει ώστε η δομή αυτή να κρατηθεί όσο πιο απλή γίνεται, ώστε η επιρροή που θα έχει ο όγκος των δεδομένων στην απόδοση, να είναι η μικρότερη δυνατή.

Μια ιδιαιτερότητα της σχήματος της βάσης είναι η έλλειψη συσχετίσεων μεταξύ των κύριων και των βοηθητικών οντοτήτων (π.χ. πίνακας REASONS με GLK). Οι συσχετίσεις αυτές δεν θα υλοποιηθούν σε επίπεδο βάσης δεδομένων, αλλά σε επίπεδο λογισμικού και αιτία της απόφασης αυτής είναι η ευελιξία που απαιτείται, λόγω της ανάγκης για υποστήριξη πολλών γλωσσών. Για να είναι εφικτή η υλοποίηση συσχέτισης σε επίπεδο βάσης, θα έπρεπε για κάθε πεδίο που αναφέρεται σε βοηθητική οντότητα (π.χ. το πεδίο REASON στον GLK αναφέρεται στο ID του πίνακα REASONS), να περιλαμβάνεται στο βασικό πίνακα, ολόκληρο το κύριο κλειδί του (π.χ. για το REASON, θα έπρεπε στον GLK να υπάρχει επιπλέον και το πεδίο REASON_LANG που μαζί με το REASON θα συσχετίζονται με το κύριο κλειδί του REASONS). Το γεγονός αυτό θα αύξανε τον όγκο των πληροφοριών στους κύριους πίνακες, κάτι που φυσικά θα πρέπει να αποφευχθεί, δεδομένου ότι υπάρχει ήδη σημαντική επιβάρυνση σε όγκο δεδομένων. Με την απόφαση να μη δημιουργηθούν οι συσχετίσεις στη βάση δεδομένων, ο έλεγχος των λογικών συσχετίσεων μεταξύ των οντοτήτων περνάει στο λογισμικό, χωρίς όμως επιβάρυνση στην απόδοση αφού ο

αριθμός των εγγραφών στους πίνακες των βοηθητικών οντοτήτων θα είναι σχετικά περιορισμένος.

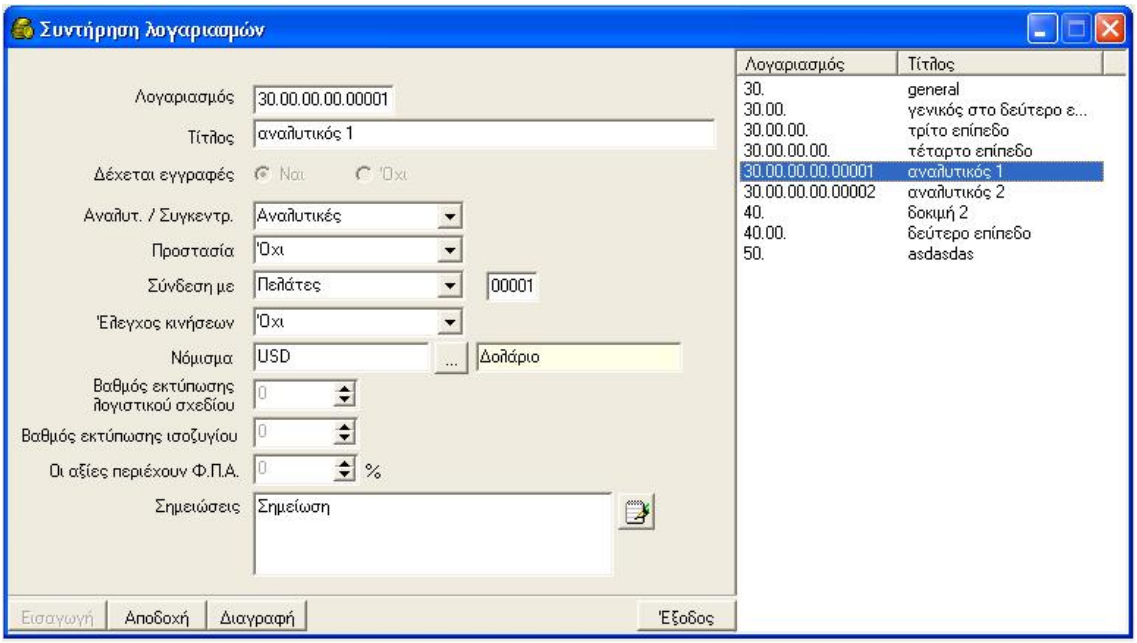


Σχήμα 3.7 Το σχήμα βάσης των οντοτήτων της λογιστικής

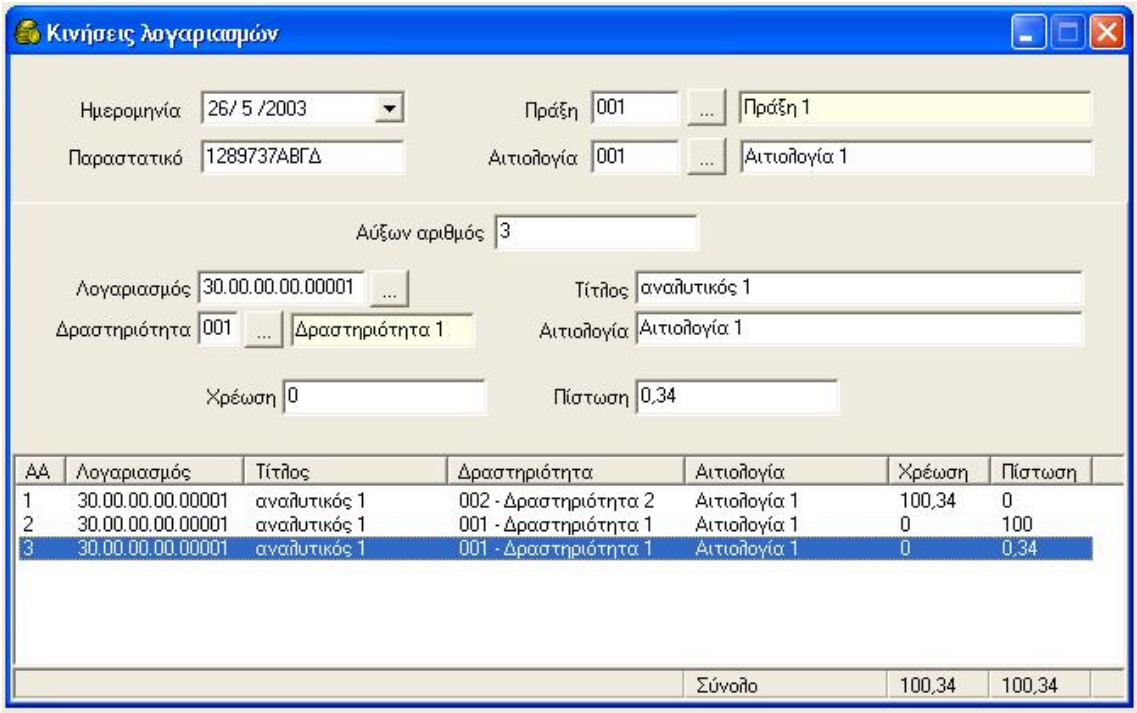
Έχοντας έτοιμο το σχήμα της βάσης για το υποσύστημα της λογιστικής, αυτό που ακολουθεί είναι η σχεδίαση των μηχανισμών χειρισμού των δεδομένων σε επίπεδο λογισμικού. Στην ουσία, πρόκειται για τη σχεδίαση των μηχανισμών εισαγωγής, τροποποίησης και διαγραφής των δεδομένων της λογιστικής. Το θέμα της πρόσβασης στη βάση δεδομένων έχει ήδη καλυφθεί από τη λειτουργικότητα που παρέχει η διεπαφή – πρότυπο και το μόνο που έχει να κάνει το υποσύστημα της

λογιστικής, είναι να περνάει σαν παραμέτρους τα ερωτήματα που πρέπει αν εκτελεστούν. Σημαντικό κομμάτι στο στάδιο αυτό, είναι και ο καθορισμός του τρόπου με τον οποίο θα εξασφαλίζεται η τήρηση των περιορισμών που διέπουν το χειρισμό των δεδομένων, τη λογική ουσιαστικά του συστήματος. Στόχος είναι η εφαρμογή των περιορισμών με τέτοιο τρόπο, ώστε περισσότερο να αποτρέπει το χρήστη από το κάνει λάθος, παρά να προσπαθεί να αναγνωρίσει το σφάλμα, αφού έχει συμβεί. Το γεγονός αυτό δείχνει τη στενή σχέση της λογικής του συστήματος με τις γραφικές διεπαφές και επομένως τη σημασία της σχεδίασης των διεπαφών, όχι μόνο για την ευχρηστία, αλλά και τη λειτουργικότητα του συστήματος.

Το συμπέρασμα, που βγαίνει από τις παραπάνω παρατηρήσεις, είναι πως το μεγαλύτερο βάρος, στα πλαίσια της σχεδίασης του λογισμικού του συστήματος, περνάει πλέον στο σχεδιασμό των γραφικών διεπαφών. Εκτός από την ανάγκη για πλήρη έλεγχο των αντικειμένων που ανήκουν στις διεπαφές, ώστε να είναι δυνατή η υλοποίηση των περιορισμών της λογιστικής, σημαντική απαίτηση είναι η βέλτιστη δυνατή εργονομία των διεπαφών. Οι διεπαφές που θα αναλάβουν τη διαχείριση των δύο βασικών οντοτήτων, απαιτούν ιδιαίτερη προσοχή τόσο στο λειτουργικό, όσο και στο σχεδιαστικό μέρος, αφού αυτές είναι οι διεπαφές, με τη μεγαλύτερη συχνότητα χρήσης σε ολόκληρο το σύστημα μιας τράπεζας. Λαμβάνοντας λοιπόν υπόψη όλα τα παραπάνω, μια πρώτη σχεδιαστική προσέγγιση των δύο αυτών διεπαφών της λογιστικής φαίνεται στα σχήματα 3.8 και 3.9. Όπως είναι φανερό, στα πρότυπα των διεπαφών του σχήματος έχουν χρησιμοποιηθεί αντικείμενα των οποίων τη λειτουργικότητα θα παρέχουν τα υπό ανάπτυξη, στα πλαίσια της εφαρμογής, εξαρτήματα



Σχήμα 3.8 Η διεπαφή διαχείρισης των λογαριασμών



Σχήμα 3.9 Η διεπαφή διαχείρισης των κινήσεων

Ξεκινώντας με τη διεπαφή της διαχείρισης των λογαριασμών, αυτό που γίνεται άμεσα φανερό, είναι το γεγονός ότι ουσιαστικά χωρίζεται σε δύο μέρη:

- το αριστερό μέρος της διεπαφής όπου χρησιμοποιείται για την εισαγωγή δεδομένων στους λογαριασμούς και
- το δεξιό μέρος όπου χρησιμεύει για τη προβολή του λογιστικού ή ενός μέρους του.

Η παροχή της δυνατότητας για προβολή του λογιστικού σχεδίου, θεωρείται πολύ σημαντική, αφού διευκολύνει σημαντικά την εύρεση και διαχείριση ενός ήδη υπάρχοντος λογαριασμού. Στο γεγονός αυτό συμβάλλει και η λειτουργία της διεπαφής, που εξασφαλίζει το συγχρονισμό του επιλεγμένου στο δεξιό μέρος λογαριασμού, με τα στοιχεία του αριστερού, αλλά και τη προσαρμογή των λογαριασμών που εμφανίζονται στο δεξιό μέρος, ανάλογα τα δεδομένα που εισάγει ο χρήστη στα πεδία του λογαριασμού και του τίτλου (π.χ. αν ο χρήστης εισάγει στο πεδίο λογαριασμού το 30 τότε στο δεξιό μέρος το σύνολο των λογαριασμών περιορίζεται σε αυτούς που ξεκινούν από 30. Το ίδιο συμβαίνει και με το τίτλο). Σημαντικός παράγοντας για τη χρησιμότητα και των δύο διεπαφών, είναι η παροχή του συνόλου της λειτουργικότητας με τη χρήση του πληκτρολογίου, κάτι που καθοδήγησε την διαδικασία του σχεδιασμού της διεπαφής.

Η δεύτερη γραφική διεπαφή του υποσυστήματος του λογιστηρίου, ακολουθεί και αυτή μια παρόμοια λογική ως προς τη λειτουργία της. Η διεπαφή των κινήσεων χωρίζεται σε τρία μέρη:

- το επάνω μέρος όπου καθορίζονται τα γενικά στοιχεία της κίνησης,
- το μεσαίο μέρος όπου καθορίζονται οι λεπτομέρειες κάθε εγγραφής της κίνησης και
- το κάτω μέρος όπου εμφανίζονται οι εγγραφές που έχουν καταχωρηθεί στη κίνηση.

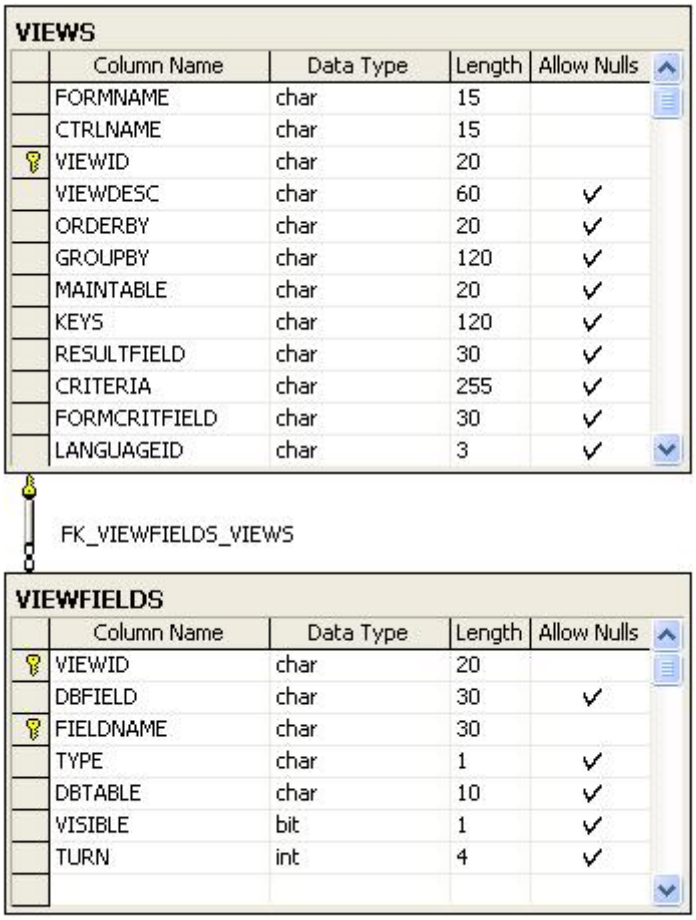
Η καταχώρηση των εγγραφών στη βάση γίνεται συνολικά με την έξοδο από τη διεπαφή και μόνο στη περίπτωση που το συνολικό χρέωσης είναι ίσο με το συνολικό ποσό πίστωσης. Η πλοήγηση ανάμεσα στις εγγραφές που έχουν ήδη καταχωρηθεί στη κίνηση (όχι στη βάση αφού αυτό γίνεται συνολικά κατά την έξοδο), μπορεί να γίνει από το κάτω μέρος της διεπαφής. Σημαντικό στοιχείο και εδώ ήταν η τοποθέτηση των αντικειμένων σε τέτοια σειρά ώστε να είναι δυνατή, σε επίπεδο υλοποίησης, η υποστήριξη του χειρισμού της διεπαφής από το πληκτρολόγιο.

Κλείνοντας τη περιγραφή των διεπαφών του υποσυστήματος της λογιστικής και μαζί το σύνολο της σχεδίασης του συγκεκριμένου υποσυστήματος, κάτι που θα

πρέπει να εξεταστεί είναι η διεπαφή της προβολής του λογιστικού σχεδίου. Η διεπαφή αυτή στην ουσία, στερείται κάθε λειτουργικότητας, αφού το μόνο που καλείται να κάνει είναι να καλέσει το μηχανισμό παραγωγής των προβολών, περνώντας φυσικά τα κατάλληλα ορίσματα. Διεπαφές σαν και αυτή, που έχουν δηλαδή ως μοναδικό σκοπό να δημιουργήσουν μια προβολή, υπάρχουν και άλλες στο χώρο της λογιστικής. Η μόνη λειτουργία των διεπαφών αυτών, είναι να περνούν αυτά που τους εισάγει ο χρήστης σαν όρισμα στο μηχανισμό των προβολών, σε μια μορφή που θα αναλυθεί στα πλαίσια του σχεδιασμού του συστήματος των προβολών. Ο λόγος που δεν υλοποιήθηκαν οι υπόλοιπες διεπαφές στα πλαίσια της δικής μας προσπάθειας, είναι αφενός η ευκολία δημιουργίας τους από οποιονδήποτε, εάν είναι ήδη διαθέσιμος ο μηχανισμός των προβολών, και αφετέρου η επιλογή για ανάθεση περισσότερων πόρων στην ανάπτυξη του ίδιου του μηχανισμού των προβολών. Με άλλα λόγια αυτό που προτιμήθηκε είναι να φτιαχτεί ένα σύστημα το οποίο θα χρειάζεται μεν επέκταση, αλλά θα αποτελεί το ίδιο βάση για επέκταση, παρά να αναπτυχθεί ένα σύστημα στο οποίο υπάρχουν μέρη που θα πρέπει να αναπτυχθούν από την αρχή κατά τη διαδικασία της επέκτασης του.

3.2.2.4.3 Το υποσύστημα διαχείρισης των προβολών

Τελευταίο κομμάτι του σχεδιασμού του συστήματος είναι ο σχεδιασμός του μηχανισμού δημιουργίας των προβολών. Και στη περίπτωση αυτή, όπως και στη περίπτωση του υποσυστήματος της λογιστικής, οι τρεις βασικότερες εργασίες που θα πρέπει να ολοκληρωθούν είναι ο καθορισμός του σχήματος της βάσης, η σχεδίαση της λειτουργίας του μηχανισμού σε επίπεδο λογισμικού και η σχεδίαση των γραφικών διεπαφών των προβολών. Αν και οι εργασίες είναι οι ίδιες και στη περίπτωση των προβολών, το μεγαλύτερο εδώ βάρος δίνεται στη σχεδίαση της λειτουργίας από πλευράς λογισμικού, αφού σε αυτό ουσιαστικά βασίζεται η δημιουργία προβολών δυναμικά. Η σχεδίαση του μηχανισμού θα πρέπει είναι τέτοια, ώστε να καλύπτει προς το παρόν τις απαιτήσεις της λογιστικής, αλλά να μπορεί να επεκταθεί για να καλύψει και τις ανάγκες ολόκληρου του συστήματος της τράπεζας, μια και θα αποτελεί ανεξάρτητη της λογιστικής μονάδα λογισμικού που μπορεί να χρησιμοποιηθεί και από άλλα υποσυστήματα.



Σχήμα 3.10 Το σχήμα βάσης για το μηχανισμό των προβολών

Σημαντικό μέρος του μηχανισμού των προβολών είναι η βάση δεδομένων που θα αποθηκεύει τις ιδιότητες των προβολών. Θεμέλιο για το σχεδιασμού του σχήματος της βάσης αυτής θα αποτελέσει η ανάλυση των ιδιοτήτων μιας προβολής, σε συνδυασμό με τις ιδιαίτερες απαιτήσεις για επεξεργασία της. Αναλύοντας λοιπόν τα βασικά χαρακτηριστικά μιας προβολής δεδομένων και προσθέτοντας τις επιπλέον απαιτήσεις που ορίστηκαν κατά το καθορισμό του έργου, εξάγεται το σχήμα της βάσης του σχήματος 3.10. Ο πίνακας των VIEWS αποτελεί ουσιαστικά την αναπαράσταση μιας προβολής, εμπεριέχοντας όλα εκείνα τα στοιχεία που είναι απαραίτητα για τη δημιουργία και εμφάνιση της προβολής στην οθόνη. Ο πίνακας VIEWFIELDS αποτελεί την αναπαράσταση των χαρακτηριστικών ενός πεδίου προβολής. Η λογική συσχέτιση μεταξύ μιας προβολής και των πεδίων που ανήκουν σε αυτή, υλοποιείται σε επίπεδο βάσης με την ένα προς πολλά συσχέτιση του πίνακα VIEWS με τον πίνακα VIEWFIELDS (ένα VIEW έχει πολλά VIEWFIELDS).

Όσον αφορά το πίνακα VIEWS, αποτελείται από τις ιδιότητες εκείνες που απαιτούνται για οριστεί μια προβολή και τα υπόλοιπα χαρακτηριστικά της. Τα πεδία επομένως του πίνακα VIEWS χωρίζονται σε δύο κατηγορίες, τα πεδία που εξυπηρετούν το σαφή ορισμό μιας προβολής και τα πεδία εκείνα που έχουν σχέση με τον τρόπο εμφάνισης των δεδομένων στη προβολή. Στη πρώτη κατηγορία από τις δύο ανήκουν τα πεδία FORMNAME, CTRLNAME, VIEWID και VIEWDESC που αντιστοιχούν στο όνομα της διεπαφής όπου ανήκει η προβολή, το όνομα του αντικειμένου που αναφέρεται, το κωδικό της προβολής και τη περιγραφή της αντίστοιχα. Στη δεύτερη κατηγορία, που είναι και η πιο ενδιαφέρουσα, ανήκουν ουσιαστικά τα υπόλοιπα πεδία του πίνακα VIEWS. Τα πεδία ORDERBY και GROUPBY αποθηκεύουν σε μορφή SQL (πεδίο1,πεδίο2,…) τα ονόματα των πεδίων της προβολής, βάσει των οποίων θα ταξινομούνται και θα ομαδοποιούνται τα δεδομένα. Τα πεδία MAINTABLE και KEYS έχουν ως σκοπό να υποστηρίξουν τη δυνατότητα μετάβασης από μια εγγραφή της προβολής, στη διεπαφή που διαχειρίζεται αυτή την εγγραφή. Τέλος, από τα υπόλοιπα πεδία, το πεδίο CRITERIA αποθηκεύει τα προκαθορισμένα κριτήρια για τη προβολή, τα πεδία FORMCRITFIELD και RESULTFIELD δηλώνουν το πεδίο που αφορά το κριτήριο που εισάγεται από τη διεπαφή και το πεδίο που επιστρέφει η προβολή αντίστοιχα και το LANGUAGEID δηλώνει τη γλώσσα της προβολής.

Ο δεύτερος πίνακας, που εξυπηρετεί τις ανάγκες του μηχανισμού των προβολών, είναι ο VIEWFIELDS, του οποίου ο ρόλος είναι η αναπαράσταση των ιδιοτήτων ενός πεδίου της προβολής. Στη περίπτωση του VIEWFIELDS πεδία που ορίζουν μια στήλη προβολής είναι το VIEWID, που αντιστοιχεί στο κωδικό της προβολής και το FIELDNAME, που αντιστοιχεί στο όνομα της στήλης. Σημαντικές ιδιότητες μιας στήλης, που είναι χρήσιμες για την αναπαράστασή της, είναι το πεδίο στο οποίο αντιστοιχεί στη βάση και αναπαρίσταται από το πεδίο DBFIELD και ο πίνακας στον οποίο ανήκει το πεδίο αυτό που αντιστοιχεί στο πεδίο DBTABLE. Τέλος, τα υπόλοιπα χαρακτηριστικά μιας στήλης είναι τα εξής:

- το είδος της στήλης (μπορεί να είναι N,S ή L και δηλώνει το αν η στήλη αθροίζεται στο τέλος «S», αν το αντίστοιχο πεδίο της τελευταίας εγγραφής μπαίνει στη θέση του αθροίσματος «L» ή είναι μια απλή στήλη «N»), που αντιστοιχεί στο πεδίο TYPE,

- η εμφάνιση ή όχι του πεδίου στη προβολή (παρέχεται η δυνατότητα για μη ορατά πεδία), που αντιστοιχεί στο πεδίο VISIBLE και
- η σειρά εμφάνισης της στήλης στη προβολή, που δηλώνεται με το πεδίο TURN.

Μετά και την περιγραφή της δομής των πινάκων του μηχανισμού των προβολών, σειρά έχει η περιγραφή του ρόλου των ιδιοτήτων αυτών και γενικότερα η περιγραφή της λειτουργίας του υποσυστήματος. Ξεκινώντας λοιπόν τη διαδικασία αυτή, το πρώτο που θα πρέπει να καθοριστεί, είναι ο τρόπος αλληλεπίδρασης του μηχανισμού με το υπόλοιπο σύστημα. Λαμβάνοντας υπόψη ότι ο μηχανισμός αυτός έχει σχεδιαστεί για να είναι ανεξάρτητος από τις διεπαφές που τον χρησιμοποιούν, η αλληλεπίδρασή του με την υπόλοιπη εφαρμογή περιορίζεται στις παραμέτρους, με τις οποίες μπορεί να κληθεί από μια διεπαφή. Μελετώντας τις ανάγκες για προβολή με περισσότερη λεπτομέρεια, γίνεται φανερό ότι η χρήση προβολών γίνεται σε δύο γενικές περιπτώσεις:

- η κλήση από κάποια διεπαφή, που έχει σαν αποκλειστικό σκοπό τη δημιουργία της προβολής με κάποια κριτήρια (η περίπτωση της διεπαφής προβολής του λογιστικού σχεδίου) και
- η κλήση της προβολής από κάποιο αντικείμενο, που απαιτεί την εισαγωγή κωδικού, και καλεί μια προβολή με σκοπό να βοηθήσει το χρήστη να επιλέξει το σωστό κωδικό (για παράδειγμα η κλήση μιας προβολής από το αντικείμενο που δέχεται κωδικό νομίσματος, για να προβληθεί στο χρήστη η αντιστοιχία των κωδικών με τις περιγραφές των νομισμάτων).

Αν και οι δύο αυτές περιπτώσεις θα μπορούσαν να καλυφθούν μαζί, κρίνεται σκόπιμο για λόγους λειτουργικότητας, να υπάρχει διαφορετικός τρόπος κλήσης του μηχανισμού των προβολών για κάθε μια από τις παραπάνω περιπτώσεις.

Στη πρώτη από τις δύο περιπτώσεις, υπάρχουν δύο βασικά χαρακτηριστικά. Το πρώτο από τα χαρακτηριστικά αυτά είναι, ότι στη περίπτωση αυτή ο μηχανισμός των προβολών δεν έχει τίποτα να επιστρέψει στη διεπαφή και επομένως η αλληλεπίδρασή του με τη διεπαφή που τον καλεί, περιορίζεται στις παραμέτρους κλήσης του. Η δεύτερη ιδιαιτερότητα είναι το γεγονός ότι η διεπαφή που καλεί τη προβολή, χρειάζεται μια προβολή σύμφωνη με τα κριτήρια που δίνει ο χρήστης μέσω αυτής, κάτι που με τη σειρά του σημαίνει ότι τα κριτήρια αυτά θα πρέπει με κάποιο

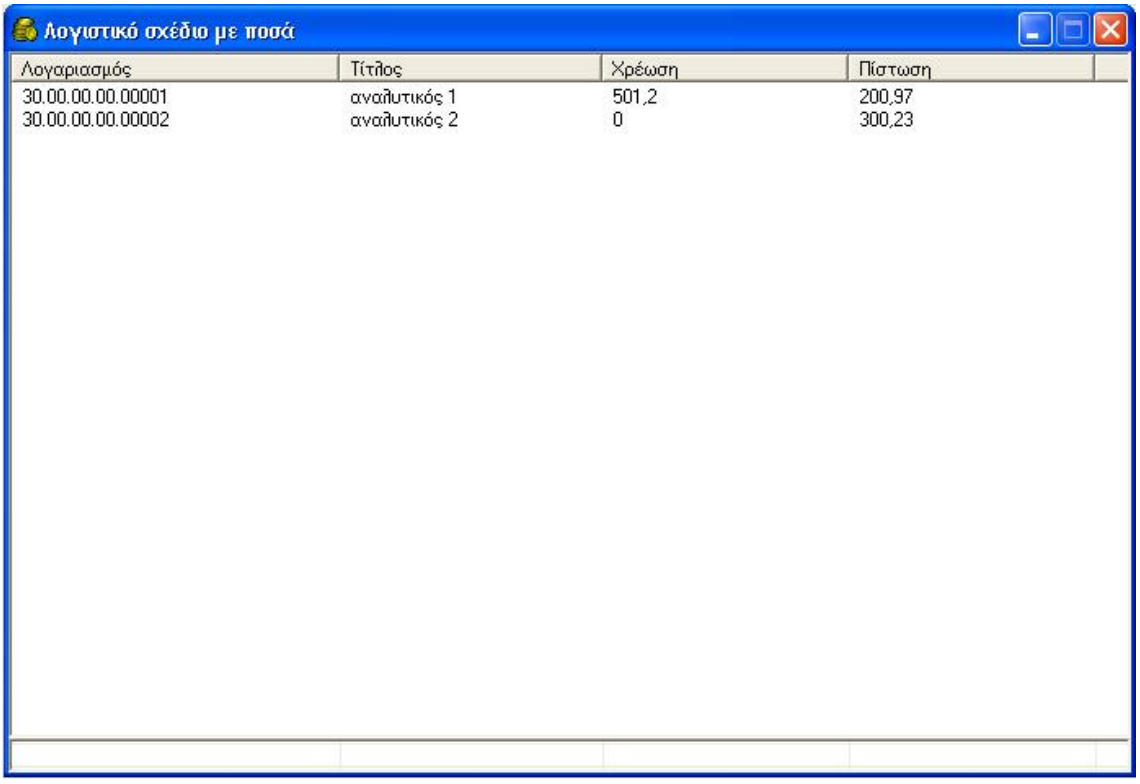
τρόπο να περάσουν στο μηχανισμό των προβολών. Είναι σαφές, ότι τα δύο αυτά χαρακτηριστικά παίζουν σημαντικό ρόλο τόσο στην αλληλεπίδραση του μηχανισμού με το σύστημα, όσο και στις υπηρεσίες που πρέπει να παρέχει ο μηχανισμός στην εφαρμογή.

Λαμβάνοντας υπόψη τα παραπάνω, παράμετρος για την κλήση μιας τέτοιας προβολής, είναι εκτός από τα στοιχεία της διεπαφής που την καλεί και τα κριτήρια που έχει διαμορφώσει ο χρήστης. Στη περίπτωση αυτή, ο μηχανισμός ανακτά από τη βάση δεδομένων όλες τις προβολές, που αναφέρονται στη συγκεκριμένη διεπαφή και δίνει τη δυνατότητα στο χρήστη να επιλέξει. Στη συνέχεια ανακτά το σύνολο των χαρακτηριστικών της επιλεγμένης επαφής, καθώς επίσης και τις στήλες που ανήκουν σε αυτή, και δημιουργεί το ερώτημα της προβολής, προσθέτοντας τα κριτήρια που έχουν εισαχθεί στο μηχανισμό σαν όρισμα. Τελευταίο στάδιο της διαδικασίας δημιουργίας της προβολής, είναι η εμφάνιση των δεδομένων που επέστρεψε το ερώτημα, σύμφωνα πάντα με τα αποθηκευμένα στη βάση χαρακτηριστικά για τη προβολή και τις στήλες της.

Στη δεύτερη περίπτωση χρήσης του μηχανισμού των προβολών από το σύστημα, ο ρόλος της προβολής είναι διαφορετικός. Σημαντικό στοιχείο της περίπτωσης αυτής είναι ότι η αλληλεπίδραση του μηχανισμού με το υπόλοιπο σύστημα δεν περιορίζεται στα ορίσματα τα οποία δέχεται σαν είσοδο. Αντίθετα, περιλαμβάνει και την επιστροφή από το μηχανισμό στο σύστημα κάποιων στοιχείων για τις επιλογές του χρήστη. Επίσης, τα κριτήρια που μπορεί να εισάγει η διεπαφή που καλεί το μηχανισμό, είναι ως ένα βαθμό προκαθορισμένα, με αποτέλεσμα να δίνεται η δυνατότητα για ένα πιο απλό τρόπο κλήσης. Γενικότερα, το γεγονός ότι στη περίπτωση αυτή η διεπαφή που καλεί το μηχανισμό δεν είναι ειδικά φτιαγμένη για το σκοπό αυτό, αλλά πρόκειται για μια οποιαδήποτε διεπαφή του συστήματος, που ζητάει μια προβολή σαν βοήθεια για τη συμπλήρωση κάποιου πεδίου, δημιουργεί την ανάγκη για ελαχιστοποίηση των απαιτήσεων του μηχανισμού από τη διεπαφή που τον καλεί.

Συνυπολογίζοντας όλες τις παραμέτρους και ιδιαιτερότητες της συγκεκριμένης περίπτωσης κλήσης, τα ορίσματα για τη κλήση του μηχανισμού θα πρέπει να είναι αφενός το όνομα της διεπαφής και αφετέρου η περιγραφή που ενδεχομένως έχει εισάγει ο χρήστης για την εύρεση του κωδικού. Εκτός από τα ορίσματα εισόδου, ο μηχανισμός θα πρέπει να επιστρέφει στη διεπαφή, το προκαθορισμένο πεδίο από την εγγραφή της προβολής που επέλεξε ο χρήστης. Η

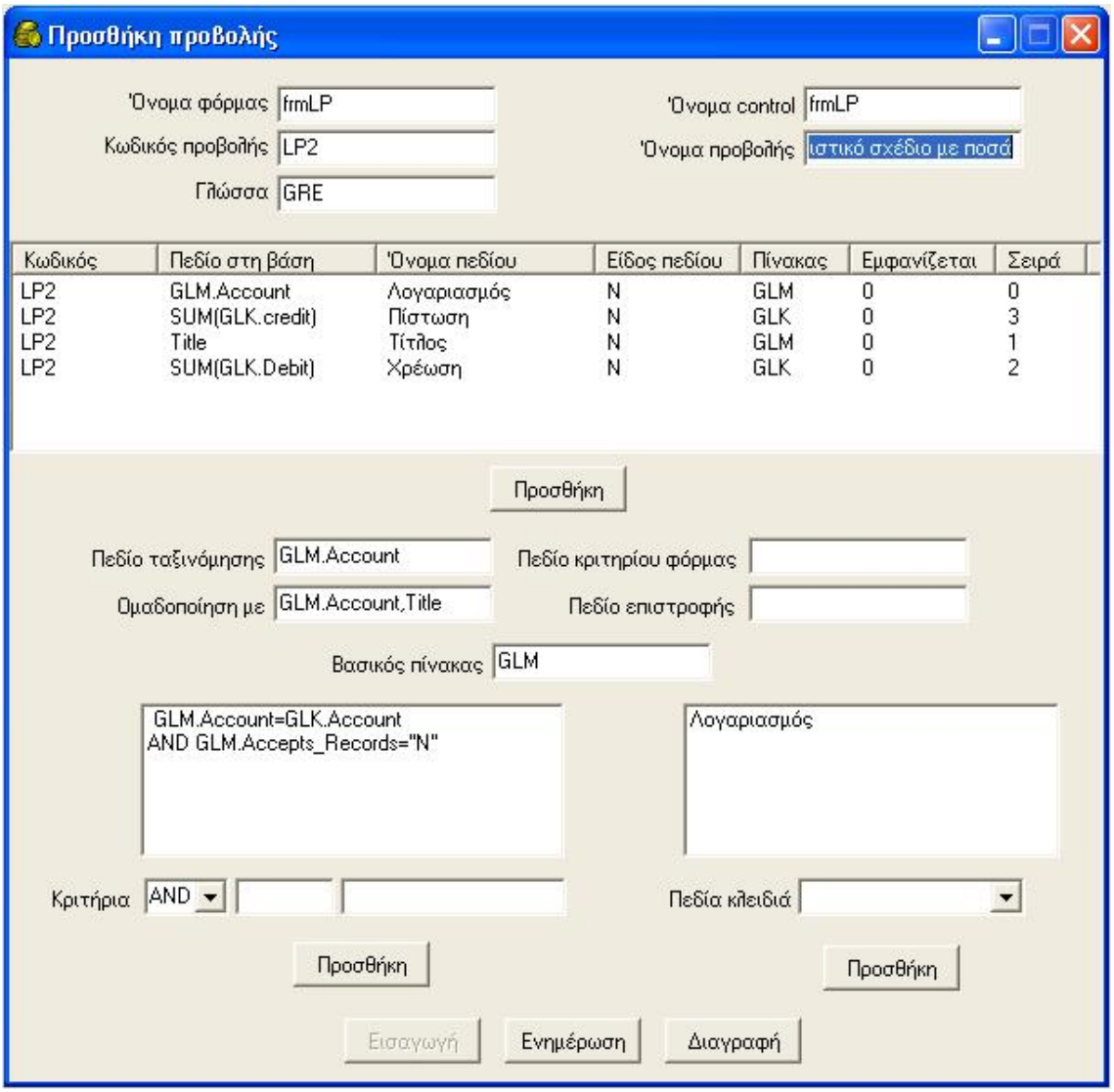
διαδικασία της δημιουργίας είναι παρόμοια με αυτή της πρώτης περίπτωσης, με τη διαφορά ότι τώρα κατά το κλείσιμο της προβολής επιστρέφεται το περιεχόμενο του πεδίου της επιλεγμένης εγγραφής, που καθορίζεται στο RESULTFIELD του πίνακα VIEWS. Μια ακόμα διαφοροποίηση είναι η δημιουργία των κριτηρίων του ερωτήματος της προβολής. Στη περίπτωση αυτή, τα κριτήρια δεν έρχονται έτοιμα από τη διεπαφή, αλλά δημιουργούνται συνδυάζοντας τη περιγραφή που έχει εισάγει ο χρήστης και το πεδίο που περιέχεται στο FORMCRITFIELD του πίνακα VIEWS (ουσιαστικά δημιουργείται ένα Like στο πεδίο που δείχνει το FORMCRITFIELD με όρισμα τη περιγραφή που έχει εισάγει ο χρήστης).



Σχήμα 3.11 Η γραφική διεπαφή εμφάνισης των προβολών

Ολοκληρώνοντας τη περιγραφή του μηχανισμού των προβολών, σειρά έχει η σχεδίαση των διεπαφών του. Αυτό που αξίζει να αναφερθεί στα πλαίσια της σχεδίασης των διεπαφών, είναι οι δυνατότητες της διεπαφής προβολής για διαχείριση των δεδομένων, όπως η εύρεση ανάμεσα στα δεδομένα της προβολής και η μετάβαση στη διεπαφή, που διαχειρίζεται την επιλεγμένη εγγραφή. Η μορφή των δύο διεπαφών του μηχανισμού των προβολών φαίνεται στα σχήματα 3.11 και 3.12. Στη πρώτη, που είναι και η διεπαφή προβολής των δεδομένων χωρίζεται σε δύο μέρη, το κομμάτι που

εμφανίζονται τα αναλυτικά δεδομένα του ερωτήματος και το κάτω μέρος όπου εμφανίζονται τα αθροίσματα ή άλλα συγκεντρωτικά δεδομένα, ανάλογα με τον τύπο της κάθε στήλης. Τέλος, η δεύτερη αφορά τη συντήρηση των προβολών και επομένως απευθύνεται στο συντηρητή του συστήματος και όχι στους τελικούς χρήστες. Το γεγονός αυτό μειώνει τη σημασία μιας λεπτομερούς μελέτης για τη σχεδίαση της διεπαφής αυτής. Αυτό που ουσιαστικά δημιουργήθηκε, είναι μια διεπαφή που απλά καλύπτει τη λειτουργικότητα, χωρίς την υλοποίηση όλων των ελέγχων για την εγκυρότητα των δεδομένων που εισάγονται.



Σχήμα 3.12 Η γραφική διεπαφή διαχείρισης των προβολών

3.2.2.5 Συμπεράσματα για το στάδιο του σχεδιασμού

Συνοψίζοντας ολόκληρη τη διαδικασία του σχεδιασμού του συστήματος, γίνεται φανερό ότι οι επιλογές όσον αφορά τη σχεδιαστική προσέγγιση, αλλά και οι επιμέρους επιλογές στα διάφορα στάδια της σχεδίασης, προσέφεραν σε ικανοποιητικό βαθμό την ευελιξία που απαιτούνταν για τη κάλυψη των απαιτήσεων. Το αποτέλεσμα της σχεδίασης φαίνεται σε πρώτη φάση εύκολα υλοποιήσιμο με τα υπάρχοντα προϊόντα ανάπτυξης λογισμικού και βάσεων δεδομένων, ενώ έχει σε μεγάλο βαθμό αποφευχθεί η πολυπλοκότητα στη σχεδίαση. Επόμενο βήμα στη διαδικασία ανάπτυξης, είναι η προσπάθεια υλοποίησης των αποτελεσμάτων του σταδίου αυτού, όπου και θα φανερωθούν ενδεχόμενες δυσκολίες, παραλήψεις ή και σφάλματα της σχεδίασης. Εξάλλου, όπως έχει ήδη αναφερθεί, ένα μεγάλο μέρος σχεδίασης ενός συστήματος λογισμικού, ανήκει περισσότερο στο στάδιο της υλοποίησης παρά του σχεδιασμού, με αποτέλεσμα το προϊόν της σχεδίασης να παίρνει τη τελική του μορφή μετά και την υλοποίηση του συστήματος.

3.3. Υλοποίηση

3.3.1. Επιλογή του περιβάλλοντος ανάπτυξης

Μετά και την ολοκλήρωση του σχεδιασμού, ακολουθεί το στάδιο της υλοποίησης του συστήματος. Στη πράξη τα τελευταία στάδια του σχεδιασμού επικαλύπτονται με την υλοποίηση, αφού σπάνια το προϊόν του έχει τέτοιο επίπεδο λεπτομέρειας, ώστε να είναι απευθείας υλοποιήσιμο. Αυτό που συνήθως συμβαίνει, είναι ο καθορισμός των λεπτομερειών του σχεδιασμού κατά την υλοποίηση, εκτός ίσως από συγκεκριμένα κομμάτια του συστήματος με ιδιαίτερες απαιτήσεις, όπως για παράδειγμα τα μέρη του συστήματος όπου υπάρχει ζήτημα ασφάλειας των δεδομένων. Σε κάθε περίπτωση βέβαια, η υλοποίηση του συστήματος είναι περισσότερο προσωπική υπόθεση του προγραμματιστή. Για το λόγο αυτό δεν υπάρχουν γενικές μέθοδοι και προσεγγίσεις, που μπορούν να εφαρμοστούν στο στάδιο αυτό της ανάπτυξης.

Η σημαντικότερη ίσως απόφαση, που μπορούμε να πούμε πως ανήκει στην ευθύνη του σταδίου αυτού, είναι η επιλογή του εργαλείου ανάπτυξης λογισμικού, που θα χρησιμοποιηθεί για την υλοποίηση του σχεδιασμού. Η υιοθέτηση μιας προσέγγισης ανάπτυξης βασισμένης στην συναρμολόγηση μερών λογισμικού, συνεπάγεται και την εύρεση του κατάλληλου περιβάλλοντος ανάπτυξης. Το περιβάλλον αυτό θα πρέπει τόσο να επιτρέπει τη συναρμολόγηση των συστατικών μερών του συστήματος, όσο και να παρέχει ένα σημαντικό αριθμό έτοιμων εξαρτημάτων λογισμικού που είναι απαραίτητα για την υλοποίηση της σχεδίασης. Το τελευταίο αυτό χαρακτηριστικό, δηλαδή η διαθεσιμότητα έτοιμων εξαρτημάτων λογισμικού, καθώς και η δυνατότητα προσαρμογής τους σε συγκεκριμένες ανάγκες του συστήματος, αποτελεί θεμελιώδη προϋπόθεση για την επιτυχία της ανάπτυξης λογισμικού μέσω συναρμολόγησης.

Μια σύντομη μελέτη της αγοράς προϊόντων ανάπτυξης λογισμικού, δείχνει σαν πιο κατάλληλο προϊόν για τη περίπτωση τη Borland Delphi. Η Delphi, που βρίσκεται σήμερα στην έκδοση στην έκδοση 7, πρωτοεμφανίστηκε το 1995 εισάγοντας για πρώτη φορά στην ανάπτυξη εφαρμογών την έννοια του Rapid Application Development (*RAD*). Η Delphi, σε αντίθεση με τα ανταγωνιστικά προϊόντα της εποχής εκείνης, παρείχε ένα εύχρηστο οπτικό περιβάλλον, συνδυασμένο με ένα πολύ καλό μεταφραστή και γρήγορη πρόσβαση σε βάσεις δεδομένων. Από

τότε μέχρι και σήμερα οι σημαντικές βελτιώσεις που έχουν γίνει τόσο στην αποδοτικότητα όσο και στην ευχρηστία του περιβάλλοντος, διατηρούν τη Delphi ως ένα από τα καλύτερα περιβάλλοντα προγραμματισμού σε Windows [7].

Η επιλογή της Delphi σαν περιβάλλον ανάπτυξης εις βάρος άλλων παρόμοιων εργαλείων βασίζεται σε αρκετά επιχειρήματα που θα μπορούσαν ίσως να συνοψιστούν με τον όρο «παραγωγικότητα». Είναι γεγονός και αποδεκτό από το μεγαλύτερο κομμάτι της αγοράς λογισμικού, ότι η Delphi είναι ίσως το παραγωγικότερο εργαλείο για ανάπτυξη εφαρμογών σε Windows. Οι λόγοι και τα χαρακτηριστικά που κάνουν το συγκεκριμένο εργαλείο τόσο παραγωγικό σε σχέση με τον ανταγωνισμό είναι αυτά που θα αναλυθούν στη συνέχεια.

Για να είναι εφικτή η σύγκριση και η εκτίμηση των εργαλείων ανάπτυξης λογισμικού ως προς τη παραγωγικότητα τους, είναι αρχικά απαραίτητο να εντοπίσουμε τα χαρακτηριστικά εκείνα ενός περιβάλλοντος προγραμματισμού που είναι καθοριστικά για την απόδοσή του ως προς τη παραγωγικότητα. Μια γενικά αποδεκτή προσέγγιση, καθορίζει 5 τέτοια χαρακτηριστικά [7]:

- η ποιότητα του οπτικού περιβάλλοντος προγραμματισμού,
- η ταχύτητα του μεταφραστή σε σχέση με την απόδοση του μεταφρασμένου κώδικα,
- οι δυνατότητες της γλώσσας σε σχέση με τη πολυπλοκότητάς της,
- η ευελιξία της αρχιτεκτονικής του όσον αφορά τη πρόσβαση σε βάσεις δεδομένων και
- η δυνατότητα του ως προς τη χρήση αλλά και την ανάπτυξη εξαρτημάτων λογισμικού.

Αυτά τα χαρακτηριστικά, σε συνδυασμό με ενδεχόμενες ιδιαίτερες απαιτήσεις του συγκεκριμένου συστήματος, θα είναι σε γενικές γραμμές τα κριτήρια σύγκρισης, που θα καθορίσουν τελικά την επιλογή του εργαλείου ανάπτυξης. Υπάρχουν βέβαια και άλλοι παράγοντες, που έχουν να κάνουν με τη παραγωγικότητα ενός εργαλείου, πολλοί από τους οποίους είναι υποκειμενικοί. Παρόλα αυτά οι παραπάνω παράγοντες είναι σε θέση να δώσουν μια καλή προσέγγιση της αντικειμενικής απόδοσης ενός εργαλείου ως προς την παραγωγικότητα.

Σημαντικότατος παράγοντας, για τη παραγωγικότητα ενός περιβάλλοντος ανάπτυξης, είναι το γραφικό του περιβάλλον. Το γραφικό περιβάλλον σε αυτού του είδους τα εργαλεία, αποτελείται από τις δυνατότητες ως προς τη συγγραφή κώδικα

(*editor*), τις δυνατότητες για εντοπισμό και διόρθωση των λαθών (*debugger*) και τις δυνατότητες για σχεδιασμό των φορμών (*form designer*). Εξετάζοντας λοιπόν τη Delphi και τον ανταγωνισμό, μπορούμε να δούμε πως στους δύο πρώτους τομείς οι επιδόσεις είναι μάλλον παρόμοιες, χωρίς να μπορεί να θεωρηθεί ότι κάποιο εργαλείο υπερέχει αισθητά του ανταγωνισμού. Όσον αφορά το τρίτο χαρακτηριστικό του γραφικού περιβάλλοντος, τα μόνα εργαλεία της αγοράς, που δίνουν αξιοποιήσιμες δυνατότητες, είναι η Delphi και η Visual Basic. Προϊόντα όπως η Visual C++ προσφέρουν μόνο υποτυπώδεις δυνατότητες στο τομέα αυτό. Συγκρίνοντας τα δύο αυτά προϊόντα, γίνεται αισθητή η υπεροχή της Delphi, κυρίως λόγω της πλήρως αντικειμενοστραφούς δομής της, που επιτρέπει τη κληρονομικότητα σε επίπεδο φορμών. Το χαρακτηριστικό αυτό μπορεί να θεωρηθεί καθοριστικό για την παραγωγικότητα, αφού επιτρέπει τη δημιουργία βασικών φορμών-προτύπων, από τις οποίες θα κληρονομήσουν τη λειτουργικότητα όλες οι άλλες φόρμες, εξοικονομώντας έτσι χρόνο και κώδικα. Η νέα Visual Basic.Net υπόσχεται και αυτή πλήρως αντικειμενοστραφές μοντέλο, υστερώντας όμως και αυτή σε σχέση με τις δυνατότητες της Delphi, τουλάχιστον στο τομέα της σχεδίασης των φορμών.

Είναι γεγονός, ότι ένα ακόμα θεμελιώδες χαρακτηριστικό για την απόδοση ενός προγραμματισμού ανάπτυξης, είναι ο μεταφραστής του. Από τη μια μεριά, ένας γρήγορος μεταφραστής επιτρέπει συχνές αλλαγές στο κώδικα, αφού ο χρόνος που απαιτείται για τη μετάφραση είναι περιορισμένος. Το χαρακτηριστικό αυτό αποδεικνύεται ιδιαίτερα σημαντικό, αφού η διαδικασία της ανάπτυξης λογισμικού είναι γενικά μια επαναλαμβανόμενη διαδικασία, που συχνά απαιτεί διορθώσεις και αναθεωρήσεις. Από την άλλη μεριά, ιδιαίτερα σημαντικό χαρακτηριστικό για ένα μεταφραστή, είναι η αποδοτικότητα του παραγόμενου κώδικα. Σε καμία περίπτωση, η επιδίωξη της ταχύτητας δε θα πρέπει να έχει επιπτώσεις στη ποιότητα του παραγόμενου κώδικα, ή τουλάχιστον όχι τέτοιες που να ξεφεύγουν από τις απαιτήσεις που υπάρχουν για το σύστημα. Όσον αφορά τη Delphi, οι επιδόσεις ως προς την ταχύτητα του μεταφραστή είναι αρκετά μπροστά από σχεδόν όλο τον ανταγωνισμό. Οι μεταφραστές για C++ που ακολουθούν, αν και κάνουν αρκετά βήματα προόδου, υστερούν ακόμα αισθητά από την απόκριση του μεταφραστή της Delphi. Επιπλέον, σημαντικό πλεονέκτημα της Delphi είναι ότι η ταχύτητα αυτή συνδυάζεται με πολύ καλές επιδόσεις στην αποδοτικότητα του κώδικα, αφού χρησιμοποιεί τον πυρήνα του μεταφραστή του C++ Builder. Βέβαια, είναι γεγονός πως σύμφωνα με τις συγκριτικές δοκιμές που έχουν γίνει μέχρι σήμερα, τις καλύτερες επιδόσεις ως προς την

αποδοτικότητα του παραγόμενου κώδικα έχει η Visual C++. Ωστόσο, για μια σειρά από λόγους που αφορούν κυρίως την ευχρηστία και τις δυνατότητες για τη σχεδίαση φορμών, η Visual C++ σε καμία περίπτωση δεν προσφέρεται για ανάπτυξη εμπορικού λογισμικού γενικής χρήσης. Για το τέλος, αφήσαμε τη Visual Basic και τη Java, που έχουν ορισμένες ιδιαιτερότητες ως προς τη τεχνολογία των μεταφραστών. Η Visual Basic κατά τη διαδικασία της ανάπτυξης λειτουργεί με διερμηνευτή (*interpreter*), με αποτέλεσμα να έχει καλούς χρόνους απόκρισης ως προς τη ταχύτητα, κοντά σε αυτούς της Delphi. Όταν πλέον έχει ολοκληρωθεί η διαδικασία ανάπτυξης, χρησιμοποιείται μεταφραστής (*compiler*), που όμως ο κώδικας που παράγει υστερεί κατά πολύ σε αποδοτικότητα και από τη Delphi και φυσικά από κάθε μεταφραστή C++. Η Java τέλος, είναι παρόμοια περίπτωση με τη διαφορά ότι χρησιμοποιεί μόνο διερμηνευτή, με αποτέλεσμα η αποδοτικότητα του κώδικα που παράγει να μην είναι συγκρίσιμη, προς το παρόν τουλάχιστον, με την αντίστοιχη των γλωσσών με μεταφραστή.

Η δυνατότητες της γλώσσας προγραμματισμού και η πολυπλοκότητά της, είναι δύο παράμετροι ιδιαίτερα αμφιλεγόμενοι, κυρίως γιατί έχουν να κάνουν με μια υποκειμενική αντίληψη του τι είναι αρκετό και του τι είναι πολύπλοκο. Παίρνοντας λοιπόν τα πράγματα από την αρχή βρίσκουμε τη Assembly, που έχει απεριόριστες δυνατότητες από τη μια μεριά, αλλά και ένα βαθμό πολυπλοκότητας από την άλλη, που ουσιαστικά απαγορεύει τη χρήση στην ανάπτυξη οποιουδήποτε σύγχρονου συστήματος. Άλλη μια γλώσσα με ιδιαίτερα αυξημένες δυνατότητες είναι η C++, παρέχοντας χαρακτηριστικά που τη καθιστούν χωρίς αμφιβολία την πληρέστερη αντικειμενοστραφή γλώσσα προγραμματισμού. Το γεγονός αυτό όμως που είναι και το κύριο πλεονέκτημά της είναι ταυτόχρονα και το βασικό της μειονέκτημα, αφού τελικά στη πράξη καταλήγουμε σε μέτρια προγράμματα με κατάχρηση των ισχυρών χαρακτηριστικών της.

Οι δύο γλώσσες προγραμματισμού που έχουν αυτή τη στιγμή τον καλύτερο δυνατό συνδυασμό δυνατοτήτων και ευχρηστίας είναι αναμφίβολα η Object Pascal και Java. Και δύο αυτές γλώσσες είναι πλήρως αντικειμενοστραφείς, δίνοντας στο προγραμματιστή ακριβώς τις δυνατότητες εκείνες που θα τον οδηγήσουν σε μια καλή σχεδίαση της εφαρμογής του, αποφεύγοντας χαρακτηριστικά όπως για παράδειγμα η πολλαπλή κληρονομικότητα, που στη πράξη το μόνο που μπορούν να προσφέρουν είναι μια κακή δόμηση του προγράμματος. Τέλος, και στο τομέα αυτό η Visual Basic αποτελεί εξαίρεση, αφού είναι προϊόν της σταδιακής εξέλιξης της Basic, μιας

γλώσσας που αρχικά αποσκοπούσε στο εισάγει αρχαρίους στο χώρο του προγραμματισμού. Αυτά τα ελλιπή της θεμέλια, καθώς και άναρχος τρόπος με τον οποίο επεκτάθηκε σταδιακά, έχει οδηγήσει σε μια γλώσσα που ελάχιστα βοηθάει στην ανάπτυξη καλών εφαρμογών.

Ίσως ο καθοριστικότερος παράγοντας για την επιλογή του περιβάλλοντος ανάπτυξης συστημάτων που χρησιμοποιούν βάσεις δεδομένων, είναι οι δυνατότητες που δίνει το περιβάλλον για πρόσβαση στη βάση, καθώς και η απόδοσή του στο τομέα αυτό. Η Delphi σε αντίθεση με τα προϊόντα της Microsoft, που παρέχουν πολύ καλές δυνατότητες για πρόσβαση στα προϊόντα βάσεων δεδομένων της ίδιας εταιρίας, είναι σε θέση σήμερα να παρέχει ένα ευρύ φάσμα δυνατοτήτων, που καλύπτουν σχεδόν το σύνολο της αγοράς συστημάτων βάσεων δεδομένων. Επιπλέον, η Delphi παρέχει σήμερα υψηλού επιπέδου εξαρτήματα λογισμικού, που επιτρέπουν γρήγορη πρόσβαση σε κάθε βάση, δίνοντας στο χρήστη τη δυνατότητα να δουλεύει σε ένα καθαρά γραφικό περιβάλλον, χωρίς να χρειάζεται να χειριστεί περιττές λεπτομέρειες. Το χαρακτηριστικό αυτό έχει σαν αποτέλεσμα την επίτευξη, με τη Delphi, της μέγιστης δυνατής παραγωγικότητας σε εφαρμογές σχετικές με βάσεις δεδομένων.

Τελευταίο και ίσως σημαντικότερο όλων χαρακτηριστικό, στη περίπτωση τουλάχιστον της ανάπτυξης λογισμικού με συναρμολόγηση έτοιμων μερών, είναι η δυνατότητα ενός περιβάλλοντος για χρήση, αλλά και ανάπτυξη εξαρτημάτων λογισμικού. Θεωρώντας τη Delphi ισοδύναμη του ανταγωνισμού σε όλους τους προαναφερθέντες παράγοντες, η βιβλιοθήκη εξαρτημάτων λογισμικού της, καθώς και οι δυνατότητες χρήσης και ανάπτυξης τέτοιων εξαρτημάτων, είναι σε θέση από μόνα τους να οδηγήσουν στην τελική επιλογή της. Διαφορετικές προσεγγίσεις στα εξαρτήματα λογισμικού όπως τα ActiveX controls, μπορεί να δίνουν τις ίδιες περίπου δυνατότητες σε επίπεδο χρήσης, όμως η δυνατότητα της βιβλιοθήκης VCL της Delphi για ανάπτυξη νέων εξαρτημάτων που κληρονομούν λειτουργικότητα από αυτά της βιβλιοθήκης, είναι μοναδική στην αγορά και απαραίτητη προϋπόθεση για την επίτευξη αυξημένης παραγωγικότητας στην ανάπτυξη συστημάτων μέσω συναρμολόγησης. Και στο τομέα αυτό σημαντική βελτίωση υπόσχεται το Visual Studio.Net, που μένει να δοκιμαστεί στη χρήση για να διαπιστωθεί κατά πόσο τα βήματα προόδου είναι ικανά ανατρέψουν την υπάρχουσα κατάσταση.

Συνοψίζοντας, τέλος, τη σύγκριση αυτή των παραμέτρων που επηρεάζουν τη παραγωγικότητα των εργαλείων ανάπτυξης λογισμικού, εύκολα καταλήγει κανείς στο συμπέρασμα πως η Delphi έχει αυτή τη στιγμή το καλύτερο δυνατό συνδυασμό

χαρακτηριστικών, που την καθιστούν το παραγωγικότερο ίσως εργαλείο για ανάπτυξη εφαρμογών σε περιβάλλον Windows. Βέβαια είναι γεγονός ότι αρκετά ανταγωνιστικά προϊόντα αποδίδουν καλύτερα από τη Delphi σε συγκεκριμένους τομείς, όμως καθένα από αυτά υστερεί σε τουλάχιστον ένα από τους υπόλοιπους σε τέτοιο βαθμό, ώστε τελικά να μην μπορεί στο σύνολο να μας οδηγήσει στην επιλογή του έναντι της Delphi. Αν στα παραπάνω προστεθεί και το γεγονός ότι το μοντέλο ανάπτυξης που έχει επιλεγεί για το συγκεκριμένο σύστημα, βασίζεται στη συναρμολόγηση εξαρτημάτων, η Delphi μοιάζει να είναι η πρώτη επιλογή με αρκετή διαφορά από τη δεύτερη.

3.3.2. Η επιλογή του DBMS

Αναμφισβήτητα, ένα πολύ σημαντικό κομμάτι του υπό ανάπτυξη συστήματος είναι η βάση δεδομένων και επομένως η επιλογή του συστήματος διαχείρισης της. Στη περίπτωση των συστημάτων διαχείρισης βάσεων δεδομένων, σε αντίθεση με τα περιβάλλοντα ανάπτυξης λογισμικού, αυτό που διαφοροποιεί τα εμπορικά προϊόντα του χώρου είναι λιγότερο οι δυνατότητες που παρέχονται και πολύ περισσότερο η απόδοση των συστημάτων αυτών. Ειδικά σε περιπτώσεις όπως του λογιστηρίου, που οι απαιτήσεις ως προς τις δυνατότητες του DBMS είναι μάλλον οι στοιχειώδεις, τα προϊόντα που μπορούν να καλύψουν από άποψης λειτουργιών τις ανάγκες, είναι μάλλον το σύνολο των προϊόντων της αγοράς. Το γεγονός αυτό είναι που θα καθορίσει και σαν βασικές παραμέτρους για την επιλογή του συστήματος δεδομένων την απόδοση και φυσικά τη παραγωγικότητα.

Στη περίπτωση του λογιστηρίου υπάρχει ακόμα ένα χαρακτηριστικό που επηρεάζει αναμφισβήτητα την επιλογή του DBMS. Το χαρακτηριστικό αυτό είναι ο μεγάλος όγκος δεδομένων που θα πρέπει να διαχειριστεί η βάση δεδομένων και φυσικά η απαίτηση για αποδεκτές, ως προς την απόκριση, επιδόσεις παρά τον όγκο αυτό. Οι ιδιαιτερότητες αυτές οδηγούν αμέσως στον αποκλεισμό όλων των προϊόντων της αγοράς που αφορούν τη διαχείριση μικρού όγκου δεδομένων και αποσκοπούν στη κάλυψη των αναγκών μικρών και μεσαίων επιχειρήσεων, αλλά σίγουρα όχι των τραπεζών. Επίσης η αξιοπιστία που απαιτείται από ένα σύστημα σαν το υπό ανάπτυξη, σε καμία περίπτωση δε μπορεί να καλυφθεί από ένα DBMS που απευθύνονται σε μικρές εφαρμογές των βάσεων δεδομένων. Το μεγάλο μειονέκτημα από τις παραπάνω διαπιστώσεις είναι σίγουρα η σημαντική αύξηση του κόστους του

λογισμικού, αφού τα συστήματα που μπορούν να καλύψουν τις ανάγκες του είναι πολύ πιο ακριβά σε σχέση με αυτά που απευθύνονται σε μικρότερης κλίμακας εφαρμογές.

Το συμπέρασμα λοιπόν που εξάγεται για την επιλογή του DBMS είναι ότι η αναζήτηση θα πρέπει να στραφεί σε προϊόντα με την απαιτούμενη απόδοση και αξιοπιστία. Μια πρόχειρη ματιά στα πλέον καταξιωμένα προϊόντα της κατηγορίας αυτής, δείχνει τρία βασικά υποψήφια: MS SQL Server, Oracle 9i και DB2. Τα τρία αυτά προϊόντα είναι αναμφισβήτητα σε θέση να καλύψουν τις ανάγκες, έχοντας παρόμοιες επιδόσεις ως προς την απόδοση και την αξιοπιστία. Διαφορές βέβαια μεταξύ των προϊόντων υπάρχουν ακόμα και στους πιο πάνω τομείς, όμως σε καμία περίπτωση δεν μπορούν να αποτελέσουν αποκλειστική αιτία επιλογής ενός από αυτά. Η παράμετρος που φαίνεται να καθορίζει την επιλογή, είναι περισσότερο η ευχρηστία και γενικότερα η παραγωγικότητα κατά την ανάπτυξη με το συγκεκριμένο εργαλείο. Σημαντικό επίσης στοιχείο είναι και η ευκολία, αλλά και το κόστος συντήρησης των συστημάτων μετά την ανάπτυξη. Στις δύο αυτές παραμέτρους, δηλαδή τη παραγωγικότητα και τη συντήρηση, το DBMS που υπερέχει έστω και ελαφρά έναντι του ανταγωνισμού είναι ο MS SQL Server, αφού τα υπόλοιπα δύο προϊόντα απαιτούν πιο εξειδικευμένο στα εργαλεία αυτά προσωπικό και περισσότερο χρόνο τόσο κατά την ανάπτυξη, αλλά και κατά τη συντήρηση του συστήματος.

Ολοκληρώνοντας τη μελέτη για την επιλογή του DBMS αξίζει να σημειωθεί η πραγματική διάσταση του ρόλου της επιλογής αυτής για το σύστημα. Είναι προφανές ότι σε καμία περίπτωση η μελέτη αυτή δεν είχε την έκταση και τη λεπτομέρεια που είχε η ανάλογη μελέτη για την επιλογή του εργαλείου ανάπτυξης του λογισμικού. Το γεγονός αυτό δεν είναι καθόλου τυχαίο, αφού οι δύο αυτές αποφάσεις έχουν τελείως διαφορετικό ρόλο στην επιτυχία του εγχειρήματος. Το εργαλείο ανάπτυξης του λογισμικού από τη μια μεριά, εκτός από σημαντικό ρόλο που έχει τόσο στο τι μπορεί να υλοποιηθεί από πλευράς δυνατοτήτων και φυσικά το ρόλο του στις επιδόσεις του συστήματος, είναι μια επιλογή που θα ακολουθεί το σύστημα σε κάθε προσπάθεια επέκτασης ή τροποποίησης του. Με άλλα λόγια, το σύστημα είναι άμεσα εξαρτημένο από το προγραμματιστικό εργαλείο στο οποίο έγινε η ανάπτυξη, και για το λόγο αυτό η επιλογή του πρέπει να είναι πολύ προσεκτική. Από την άλλη μεριά, στόχος κατά την ανάπτυξη του συστήματος, είναι η επίτευξη της μεγαλύτερης δυνατής ανεξαρτησίας του λογισμικού από το σύστημα διαχείρισης της βάσης δεδομένων. Η χρήση της προτυποποιημένης SQL και η απλή σχετικά σχεδίαση της βάσης, κάνει τη

μετάβαση του συστήματος από ένα DBMS σε ένα άλλο, εξαιρετικά εύκολη, πετυχαίνοντας με αυτό τον τρόπο τη καλύτερη δυνατή ανεξαρτησία από το συγκεκριμένο DBMS. Το γεγονός αυτό μειώνει σημαντικά τη σημασία της αρχικής επιλογής του συστήματος της βάσης δεδομένων, αφού είναι φανερό πως η απόφαση αυτή εύκολα μπορεί να αναθεωρηθεί στη συνέχεια.

3.3.3 Η υλοποίηση του συστήματος.

3.3.3.1 Γενικά

Μετά και την απόφαση για τα εργαλεία που θα χρησιμοποιηθούν κατά το στάδιο της υλοποίησης, σειρά έχει η περιγραφή της διαδικασίας αυτής. Από τη πλευρά της βάσης δεδομένων, η διαδικασία της υλοποίησης δεν παρουσιάζει ιδιαίτερο ενδιαφέρον, αφού η σχεδίαση της βάσης είναι άμεσα υλοποιήσιμη σε γλώσσα SQL και επομένως και στον MS SQL Server. Αυτό που πραγματικά έχει ενδιαφέρον, είναι η υλοποίηση της σχεδίασης του λογισμικού, αφού αφενός θα πρέπει να αντιστοιχηθούν όλα τα νοητά αντικείμενα, που χρησιμοποιήθηκαν σε επίπεδο σχεδιασμού, με πραγματικά εξαρτήματα του περιβάλλοντος της Delphi και αφετέρου, να υλοποιηθούν προγραμματιστικά όλοι οι μηχανισμοί, των οποίων η δομή και λειτουργία περιγράφεται στη σχεδίαση. Στο στάδιο αυτό, δεν υπάρχουν γενικές καθορισμένες μέθοδοι για τη μετατροπή της σχεδίασης σε εκτελέσιμο λογισμικό, αφού η υλοποίηση του λογισμικού γενικότερα είναι μια διαδικασία που έχει άμεση σχέση με τον ίδιο το προγραμματιστή και τον τρόπο που αντιλαμβάνεται τη σχεδίαση.

3.3.3.2 Περιγραφή των εξαρτημάτων

Βασική προϋπόθεση για την υλοποίηση των διεπαφών και των επιμέρους μηχανισμών, είναι ο καθορισμός των εξαρτημάτων που θα χρησιμοποιηθούν και ο ρόλος τους μέσα στο σύστημα. Επειδή τα εξαρτήματα που χρησιμοποιηθούν έχουν ήδη περιγραφεί κατά το σχεδιασμό, αυτό που στη πραγματικότητα πρέπει να γίνει, είναι η αντιστοίχιση των αντικειμένων και εξαρτημάτων που χρησιμοποιήθηκαν κατά τη σχεδίαση, με πραγματικά εξαρτήματα και αντικείμενα του περιβάλλοντος της Delphi. Η διαδικασία αυτή δεν παρουσιάζει ιδιαίτερες δυσκολίες γιατί, όπως θα γίνει

φανερό στη συνέχεια, κατά τη περιγραφή της σχεδίασης έγινε προσπάθεια να χρησιμοποιηθούν αντικείμενα που γενικά είναι διαθέσιμα σε όλα τα σύγχρονα εργαλεία ανάπτυξης λογισμικού και επομένως και στη Delphi.

Μια από τις βασικές λειτουργικές ανάγκες, που θα πρέπει να καλυφθούν με εξαρτήματα της Delphi, είναι η πρόσβαση στη βάση δεδομένων. Η ανάγκες στο τομέα αυτό, μεταφράζονται ουσιαστικά στη δυνατότητα να εκτελούνται ερωτήματα σε γλώσσα SQL και φυσικά να παρέχεται προγραμματιστικά πρόσβαση στα αποτελέσματα, όταν υπάρχουν. Για την κάλυψη τέτοιων αναγκών, η Delphi παρέχει δύο εξαρτήματα που χρησιμοποιούνται σε συνδυασμό: το TDatabase και το TQuery. Το εξάρτημα TDatabase έχει ως ρόλο τη διαχείριση μιας σύνδεσης με τη βάση δεδομένων, ενώ το TQuery την εκτέλεση ερωτημάτων SQL. Ένα αντικείμενο TQuery συνδέεται με ένα αντικείμενο TDatabase, που του παρέχει τη σύνδεση με τη βάση, και έτσι το μόνο που μένει, είναι να εκχωρηθεί στην ιδιότητα "SQL" του αντικειμένου TQuery, η εντολή SQL και να εκτελεστεί το ερώτημα. Τα αποτελέσματα του ερωτήματος είναι διαθέσιμα μέσα από το αντικείμενο TQuery και κάθε πεδίο του μπορεί να προσπελαστεί με το όνομα της στήλης που επιστρέφει το ερώτημα (π.χ. στο ερώτημα "SELECT πεδίο FROM πίνακα" αφού εντοπίσουμε την εγγραφή και στείλουμε το TQuery να δείχνει εκεί, μπορούμε να ανακτήσουμε τα δεδομένα του πεδίου ως εξής: (όνομα TQuery).FieldByName('πεδίο')) . Αν και η Delphi έχει και άλλα εξαρτήματα για τη πρόσβαση σε βάσεις δεδομένων με διαφορετικό τρόπο λειτουργίας, η προσέγγιση αυτή είναι προτιμώμενη κυρίως λόγω της απλότητας και της ευελιξίας που μπορεί να προσφέρει.

Ακόμα ένας τομέας, όπου απαιτείται η εκτεταμένη χρήση των εξαρτημάτων της Delphi, είναι η υλοποίηση των γραφικών διεπαφών. Στο σημείο αυτό πρέπει να αναφερθεί ότι εκτός από τα εξαρτήματα που παρέχει το προγραμματιστικό περιβάλλον, χρησιμοποιήθηκαν και τα δύο εξαρτήματα που αναπτύχθηκαν στα πλαίσια της ανάπτυξης του συστήματος. Το πρώτο από τα εξαρτήματα αυτά ονομάζεται TAccountEdit και κληρονομεί τη βασική λειτουργικότητά του από το TCustomMaskEdit της Delphi. Αυτό που έχει προστεθεί στο εξάρτημα αυτό σε σχέση με αυτό που κληρονομεί, είναι ουσιαστικά η υλοποίηση όλων των ελέγχων για την εξασφάλιση της λογικής συνέπειας των λογαριασμών. Αυτό που τελικά επιτυγχάνεται με τον τρόπο αυτό, είναι η εφαρμογή όλων των περιορισμών που διέπουν τη μορφή ενός λογαριασμού, μέσα σε ένα εξάρτημα γραφικής διεπαφής, με αποτέλεσμα να γίνεται εφικτή η πρόληψη των λαθών του χρήστη κατά την εισαγωγή.

Παρόμοια είναι και η περίπτωση του δεύτερου εξαρτήματος που ονομάζεται TLimitSpinEdit και κληρονομεί από το εξάρτημα TspinEdit. Στη δεύτερη αυτή περίπτωση, στόχος είναι η κάλυψη της ανάγκης για εισαγωγή αριθμών με ελεγχόμενη ακρίβεια και αυτό επιτυγχάνεται με την υλοποίηση των κατάλληλων ελέγχων μέσα στο εξάρτημα. Η χρησιμότητα των δύο εξαρτημάτων θα γίνει περισσότερο κατανοητή κατά την περιγραφή των γραφικών διεπαφών του συστήματος. Ο πηγαίος κώδικάς τους παρατίθεται στη παράγραφο 1 του παραρτήματος Γ.

3.3.3.3 Η υλοποίηση της διεπαφής - πρότυπο

Έχοντας καθορίσει τα εξαρτήματα που θα χρησιμοποιηθούν στα πλαίσια της υλοποίησης, σειρά έχει η περιγραφή της υλοποίησης των μερών του συστήματος, όπως αυτά περιγράφηκαν κατά το σχεδιασμό. Πρώτο από τα μέρη αυτά που θα πρέπει να υλοποιηθεί είναι αναμφισβήτητα το κομμάτι της διεπαφής-πρότυπο, αφού από αυτήν θα κληρονομούν όλες οι άλλες γραφικές διεπαφές του συστήματος. Όπως έχει καθοριστεί κατά το σχεδιασμό, η λειτουργικότητα της διεπαφής αυτής περιλαμβάνει το χειρισμό της επικοινωνίας με τη βάση δεδομένων, την ανάκτηση των περιγραφών που εμφανίζονται πάνω στη διεπαφή και τη παροχή της δυνατότητας για ανάκτηση των περιγραφών των μηνυμάτων από τη βάση. Από τις λειτουργίες αυτές, η δεύτερη λειτουργία θα είναι σχεδόν αυτόματη, αφού η ίδια η διεπαφή-πρότυπο θα ανακτά για κάθε αντικείμενο που βρίσκεται επάνω της, τη περιγραφή του από τη βάση, ανάλογα με το όνομά του (στο onCreate event της διεπαφής ελέγχει τα αντικείμενα που βρίσκονται επάνω της και ανάλογα με το τύπο και το όνομά τους ανακτά την ανάλογη περιγραφή, χρησιμοποιώντας και τη συνάρτηση FillComboBox όταν πρόκειται για αντικείμενο τύπου TComboBox). Για τις υπόλοιπες δύο λειτουργίες παρέχονται συναρτήσεις, που θα μπορούν να χρησιμοποιηθούν από τις διεπαφές που κληρονομούν από αυτή. Όσον αφορά την επικοινωνία με τη βάση, παρέχονται δύο συναρτήσεις, με ορίσματα το αντικείμενο TQuery που θα χρησιμοποιηθεί και το SQL ερώτημα που θα εκτελεστεί. Από τις δύο αυτές συναρτήσεις, η μία αφορά την εκτέλεση ενός ερωτήματος SQL χωρίς την επιστροφή αποτελεσμάτων (η ExecQuery) και η άλλη αφορά αποκλειστικά την ανάκτηση δεδομένων (η GetData). Για την ανάκτηση των περιγραφών των μηνυμάτων παρέχεται μια συνάρτηση (η GetError) με όρισμα το κωδικό του λάθους, η οποία ανακτά από το πίνακα Errors την εγγραφή με το κωδικό που δόθηκε σαν όρισμα.

Μια ακόμα λειτουργία που προέκυψε κατά την υλοποίηση, είναι η δημιουργία μιας συνάρτησης (CheckIfExists), που θα δέχεται σαν όρισμα ένα πίνακα και ένα κωδικό (ID), θα ελέγχει αν υπάρχει εγγραφή με αυτό το κωδικό στο πίνακα και θα επιστρέφει τη περιγραφή της. Η δομή των πινάκων που μπορεί να χειριστεί η συνάρτηση αυτή, είναι η δομή των πινάκων των βοηθητικών οντοτήτων της λογιστικής και στόχος είναι ο έλεγχος στη βάση για την ύπαρξη ενός συγκεκριμένου κωδικού σε μια βοηθητική οντότητα (π.χ. ο χρήστης εισάγει σε μια διεπαφή το κωδικό 001 για τη δραστηριότητα. Η συνάρτηση αυτή θα κληθεί από μια διεπαφή με ορίσματα «001» και «activities» και θα ελέγξει αν στο πίνακα activities υπάρχει εγγραφή με ID = 001, και αν ναι θα επιστρέψει το πεδίο «Description» του πίνακα για την εγγραφή αυτή). Τέλος κάτι ακόμα που παρουσιάστηκε κατά την υλοποίηση, είναι η ανάγκη για μια συνάρτηση, που επιστρέφει τη περιγραφή ενός αντικειμένου, το οποίο ανήκει σε κάποιο ComboBox. Η συνάρτηση αυτή (GetListValue), δέχεται σαν όρισμα το ID της λίστας και το ID ενός αντικειμένου της και επιστρέφει τη περιγραφή του συγκεκριμένου αντικειμένου από το πίνακα ListValues. Ο πηγαίος κώδικας της διεπαφής-πρότυπο, αλλά και των υπόλοιπων διεπαφών που εξυπηρετούν τις ανάγκες ολόκληρου του συστήματος, παρατίθεται στην παράγραφο 2 του παραρτήματος Α.

3.3.3.4 Η υλοποίηση του υποσυστήματος των προβολών

Το επόμενο μέρος του συστήματος που θα υλοποιηθεί, είναι το υποσύστημα δημιουργίας των προβολών. Έχοντας περιγράψει τη δομή της βάσης, αλλά και τη λειτουργία του μηχανισμού στα πλαίσια της σχεδίασης, το μόνο που μένει είναι η κωδικοποίηση των παραπάνω στο περιβάλλον της Delphi. Η διεπαφή που αναλαμβάνει την εμφάνιση των προβολών και υλοποιεί επομένως και το μηχανισμό δημιουργίας τους, αντιστοιχεί στο αντικείμενο TfrmViews. Τα εξαρτήματα που έχουν χρησιμοποιηθεί για τη προβολή των δεδομένων είναι δύο αντικείμενα τύπου TListView (ένα για τα δεδομένα και ένα με μία μόνο γραμμή για τα αθροίσματα). Οι στήλες των δύο αυτών αντικειμένων δημιουργούνται ανάλογα με τα πεδία που είναι καταχωρημένα για τη συγκεκριμένη προβολή στο πίνακα ViewFields, ενώ ανάλογα με τις ιδιότητες των πεδίων υπολογίζονται και τα αθροίσματα στα δεύτερο ListView. Μια ακόμα λειτουργία που υλοποιείται πάνω στη διεπαφή αυτή, είναι η μετάβαση

από μια εγγραφή της προβολής, στη διεπαφή από την οποία ο χρήστης μπορεί να διαχειριστεί την εγγραφή αυτή (συνάρτηση MoveToTable).

Εκτός από τη συγκεκριμένη διεπαφή, στα πλαίσια του μηχανισμού των προβολών έχουν υλοποιηθεί οι διεπαφές TfrmSelView και TfrmSearchOnView, που χρησιμεύουν στην επιλογή από το χρήστη της επιθυμητής προβολής, όταν υπάρχουν για την ίδια χρήση πάνω από μια προβολές καταχωρημένες και στην εύρεση πάνω στα δεδομένα μιας προβολής αντίστοιχα. Για τη διαχείριση των προβολών, απαιτήθηκε η υλοποίηση των διεπαφών TfrmAddViews και TfrmAddFieldToView που αναλαμβάνουν τη διαχείριση των προβολών και των πεδίων τους αντίστοιχα. Είναι σαφές, ότι οι δύο αυτές διεπαφές, δεν διακρίνονται από τη ίδια φιλικότητα ως προς το χρήστη σε σχέση με τις υπόλοιπες, κάτι που είναι αναμενόμενο εφόσον δεν απευθύνονται στο απλό χρήστη, αλλά στο διαχειριστή ή συντηρητή του συστήματος. Το σύνολο των διεπαφών του μηχανισμού των προβολών, μαζί με τη περιγραφή των εξαρτημάτων που χρησιμοποιήθηκαν φαίνονται στη παράγραφο 2 του παραρτήματος Β. Ο πηγαίος κώδικας των όλων των διεπαφών, καθώς και η αναλυτική περιγραφή της διαδικασίας δημιουργίας μιας προβολής, παρατίθενται στη παράγραφο 3 του παραρτήματος Γ.

3.3.3.5 Η υλοποίηση του υποσυστήματος της λογιστικής

Τελευταίο μέρος του συστήματος που μένει να υλοποιηθεί είναι το υποσύστημα που αφορά αποκλειστικά τη λογιστική. Οι διεπαφές που διαχειρίζονται τις δύο βασικότερες οντότητες της λογιστικής, είναι η TfrmAccounts, που αναλαμβάνει τη διαχείριση των λογαριασμών και η TfrmAccountRecords, που διαχειρίζεται τις κινήσεις των λογαριασμών. Οι περιορισμοί που διέπουν τη διαχείριση των οντοτήτων αυτών, υλοποιούνται στις διεπαφές με τέτοιο τρόπο, ώστε να μην επιτρέπεται στο χρήστη να εισάγει δεδομένα χωρίς λογική συνέπεια. Αυτό επιτυγχάνεται με την ενεργοποίηση και απενεργοποίηση των αντικειμένων που ανήκουν στις διεπαφές, ανάλογα με τις επιλογές του χρήστη. Ένα παράδειγμα της προσέγγισης αυτής στην εξασφάλιση των περιορισμών, είναι η περίπτωση που στη διεπαφή των λογαριασμών ο χρήστης επιλέγει ότι ο λογαριασμός δεν δέχεται εγγραφές. Στη περίπτωση αυτή, τα αντικείμενα για την εισαγωγή του τρόπου που δέχεται εγγραφές ο λογαριασμός (Αναλυτικές / Συγκεντρωτικές), του τύπου προστασίας, της σύνδεσης και του ελέγχου κινήσεων του λογαριασμού,

απενεργοποιούνται αφού οι παραπάνω ιδιότητες έχουν νόημα μόνο για λογαριασμούς που δέχονται εγγραφές. Με παρόμοιο τρόπο έχει υλοποιηθεί επάνω στις διεπαφές το σύνολο των περιορισμών που διέπουν τις ιδιότητες των οντοτήτων, πετυχαίνοντας έτσι την ελαχιστοποίηση των πόρων που δαπανώνται για έλεγχο των λαθών.

Δύο σημεία που επίσης αξίζει να αναφερθούν στα πλαίσια της υλοποίησης του υποσυστήματος της λογιστικής, είναι η χρήση των δύο εξαρτημάτων που αναπτύχθηκαν και η αλληλεπίδραση των διεπαφών με το μηχανισμό των προβολών, όπου αυτό είναι απαραίτητο. Όσον αφορά τα εξαρτήματα, είναι φανερή η συνεισφορά τους στη προσπάθεια για εξασφάλιση της λογικής συνέπειας των δεδομένων πάνω στις διεπαφές, που περιγράφηκε παραπάνω. Είναι σαφές, ότι η εφαρμογή όλων των περιορισμών που διέπουν τη μορφή των λογαριασμών της λογιστικής και οι έλεγχοι της ακρίβειας των αριθμών που εισάγονται θα έπρεπε, στη περίπτωση που δεν είχαν αναπτυχθεί τα εξαρτήματα, να γίνει μετά την εισαγωγή των δεδομένων, επιβαρύνοντας την απόκριση του συστήματος και αυξάνοντας και την προγραμματιστική πολυπλοκότητα. Το γεγονός αυτό θα είχε αναπόφευκτες επιπτώσεις τόσο στη δυσκολία κατά την ανάπτυξη, όσο και δυσκολία ως προς τη συντήρηση.

Το δεύτερο σημείο που αξίζει να αναφερθεί είναι ο τρόπος με τον οποίο γίνεται η χρήση του μηχανισμού των προβολών από το υποσύστημα της λογιστικής. Ο ρόλος των προβολών στο σημείο αυτό είναι βοηθητικός και έχει να κάνει με τη προβολή των κωδικών και των αντίστοιχων περιγραφών ώστε να διευκολύνεται ο χρήστης στην επιλογή ενός κωδικού. Για παράδειγμα, αν ο χρήστης πατήσει το κουμπί δίπλα από το πεδίου κωδικού του νομίσματος ή αν επιχειρήσει να φύγει από το πεδίο της περιγραφής του χωρίς να έχει συμπληρωθεί ο κωδικός, καλείται ο μηχανισμός των προβολών και δημιουργείται μια προβολή με τους κωδικούς των νομισμάτων και τη περιγραφή τους. Όταν ο χρήστης κλείσει την προβολή επιστρέφεται από το μηχανισμό ο επιλεγμένος κωδικός και η διεπαφή των λογαριασμών αναλαμβάνει να τον τοποθετήσει στο κατάλληλο πεδίο της. Με τον τρόπο αυτό λειτουργούν όλα τα αντίστοιχα πεδία λογιστικής τα οποία πρέπει να συμπληρωθούν με κωδικό.

Εκτός από τις δύο βασικές διεπαφές που αναφέρθηκαν πιο πάνω, δύο ακόμα διεπαφές που ανήκουν στο υποσύστημα της λογιστικής, είναι η TfrmLP και η TfrmNotes. Η χρησιμότητα της δεύτερης από αυτές, είναι η διαχείριση των σημειώσεων όλων των επιπέδων που έχει δικαίωμα να διαχειριστεί ο συγκεκριμένος

χρήστης και καλείται από τη διεπαφή διαχείρισης των διεπαφών (το πεδίο σημειώσεων που βρίσκεται επάνω στη διεπαφή των λογαριασμών εμφανίζει μόνο τις σημειώσεις στο επίπεδο του χρήστη). Η πρώτη φόρμα είναι η υλοποίηση της διεπαφής για τη προβολή του λογιστικού σχεδίου, που περιγράφηκε κατά το σχεδιασμό. Ο ρόλος της φόρμας αυτής, όπως έχει ήδη καθοριστεί, είναι ουσιαστικά η προετοιμασία των κριτηρίων με τα οποία θα κληθεί ο μηχανισμός των προβολών για την εμφάνιση του λογιστικού σχεδίου. Στη παράγραφο 1 του παραρτήματος Β φαίνονται οι περιγραφές των φορμών της λογιστικής ενώ στη παράγραφο 4 του παραρτήματος Γ παρατίθεται και ο αντίστοιχος πηγαίος κώδικας των διεπαφών.

3.3.3.6 Συμπεράσματα του σταδίου της υλοποίησης

Ολοκληρώνοντας τη περιγραφή του σταδίου της υλοποίησης, η πιο σημαντική διαπίστωση είναι το γεγονός ότι λαμβάνοντας υπόψη τις απαιτήσεις, όσον αφορά τη λειτουργικότητα και όχι μόνο, η προσπάθεια που καταβλήθηκε στα πλαίσια της υλοποίησης ήταν σχετικά περιορισμένη. Σημαντικό ρόλο στο γεγονός έπαιξαν αφενός η επιλογή του μοντέλου ανάπτυξης, και αφετέρου η ύπαρξη του κατάλληλου εργαλείου για να το υποστηρίξει. Το μοντέλο ανάπτυξης μέσω συναρμολόγησης από τη μια μεριά βοήθησε στη μέγιστη δυνατή χρησιμοποίηση έτοιμων μερών, ενώ η ύπαρξη ενός περιβάλλοντος όπως η Delphi παρείχε τόσο το μεγαλύτερο μέρος των εξαρτημάτων, όσο και το πλαίσιο για την εύκολη υλοποίηση της σχεδίασης. Αν σε όλα αυτά προστεθεί και η προσπάθεια της υλοποίησης των περιορισμών σε επίπεδο γραφικών διεπαφών, αποφεύγοντας σε πολλές περιπτώσεις την υλοποίηση πολύπλοκων αλγορίθμων για το σκοπό αυτό, γίνεται αντιληπτός ο τρόπος με τον οποίο ελαχιστοποιήθηκε η προσπάθεια που καταβλήθηκε για την υλοποίηση. Το μόνο που απομένει πλέον για την ολοκλήρωση της προσπάθειας ανάπτυξης, είναι η σύγκριση του αποτελέσματος με τις αρχικά διατυπωμένες απαιτήσεις, ώστε να επιβεβαιωθεί ότι το σύστημα ανταποκρίνεται πραγματικά στις ανάγκες του χώρου όπου απευθύνεται.

3.4. Δοκιμή και επικύρωση

3.4.1. Μέθοδοι ελέγχου και επικύρωσης

Ο έλεγχος και η επικύρωση του λογισμικού, έχουν ως σκοπό να εξασφαλίσουν ότι το σύστημα ικανοποιεί τις απαιτήσεις που έχουν καθοριστεί και κατ'επέκταση τις ανάγκες των χρηστών του. Αν και έχει να κάνει με όλα τα στάδια της ανάπτυξης, το κύριο μέρος του ελέγχου γίνεται μετά την ολοκλήρωση και της υλοποίησης. Στα μεγάλης κλίμακας συστήματα που αποτελούνται από αρκετά υποσυστήματα, ο έλεγχος αυτός θα πρέπει πρώτα να γίνει και σε κάθε υποσύστημα ξεχωριστά και γενικότερα σε κάθε αυτοτελές συστατικό κομμάτι του συστήματος. Στη συνέχεια και αφού έχει εξασφαλισθεί η αξιοπιστία των επιμέρους συστατικών, ελέγχεται η απόδοση του συστήματος σαν σύνολο.

Η κλιμακωτή αυτή προσέγγιση για τον έλεγχο των μερών ενός συστήματος, μπορεί χωριστεί στα εξής πέντε επίπεδα ελέγχου [1]:

- έλεγχος εξαρτημάτων, όπου περιλαμβάνει ουσιαστικά τον έλεγχο κάθε εξαρτήματος που χρησιμοποιείται,
- έλεγχος αυτοτελών μονάδων, όπου αφορά τον έλεγχο ενός συνόλου εξαρτημάτων που συνεργάζονται δημιουργώντας μια αυτοτελή μονάδα,
- τον έλεγχο των υποσυστημάτων, που ασχολείται κυρίως με τον έλεγχο της ολοκλήρωσης των αυτοτελών μονάδων, ώστε να συγκροτήσουν τα υποσυστήματα,
- τον έλεγχο συνολικά του συστήματος, που είναι τελικά αυτός που ελέγχει τη συνέπεια του συστήματος ως προς τις απαιτήσεις και τέλος
- ο έλεγχος αποδοχής του συστήματος, όπου πλέον το σύστημα ελέγχεται με πραγματικά δεδομένα και σε πραγματικές συνθήκες λειτουργίας από τους χρήστες του.

Από τα παραπάνω στάδια, τα δύο πρώτα συνήθως πραγματοποιούνται από τους ίδιους τους προγραμματιστές, που αναλαμβάνουν την υλοποίηση και για το λόγο αυτό θεωρούνται περισσότερο κομμάτι της υλοποίησης και λιγότερο της επικύρωσης του συστήματος.

Τα επόμενα στάδια του ελέγχου απαιτούν ιδιαίτερη προσπάθεια και οργάνωση και είναι αυτά που ουσιαστικά αποτελούν το κύριο μέρος της επικύρωσης του

λογισμικού. Για τα στάδια αυτά απαιτείται η απασχόληση μιας ομάδας προγραμματιστών, που θα εφαρμόσουν στο σύστημα μια σειρά προκαθορισμένων σεναρίων ελέγχου, με σκοπό τη παρατήρηση των αποτελεσμάτων του συστήματος και τη σύγκριση με τα επιθυμητά. Ο έλεγχος αποδοχής, που ονομάζεται "*alpha testing*", είναι μια διαδικασία που συνεχίζεται μέχρι ο δημιουργός του συστήματος και ο πελάτης συμφωνήσουν ότι το υπάρχον σύστημα είναι μια αποδεκτή υλοποίηση των απαιτήσεων. Αφού ολοκληρωθεί και η διαδικασία αυτή, το σύστημα θεωρείται εμπορεύσιμο προϊόν και περνάει και στο τελευταίο στάδιο ελέγχου που ονομάζεται "*beta testing*" και στο οποίο το σύστημα ελέγχεται από ένα περιορισμένο αριθμό τελικών χρηστών, με σκοπό τον έλεγχο της συμπεριφοράς του σε πραγματικές συνθήκες χρήσης.

3.4.2. Έλεγχος και επικύρωση του προϊόντος της υλοποίησης

Το στάδιο αυτό της ανάπτυξης του συγκεκριμένου συστήματος ανήκει στα στάδια, που θα πρέπει να υποστούν αρκετές αλλαγές σε σχέση με τα όσα αναφέρθηκαν πιο πάνω για τις γενικές μεθόδους ελέγχου και επικύρωσης. Όπως και στη περίπτωση του καθορισμού του έργου, που οι μέθοδοι που ακολουθήθηκαν διέφεραν από τις συνήθεις πρακτικές, έτσι και στον έλεγχο του συστήματος δεν είναι δυνατό να ακολουθηθούν πιστά τα όσα περιγράφηκαν για τις μεθόδους. Ο λόγος αυτή τη φορά είναι το γεγονός ότι η όλη προσπάθεια δε γίνεται με τους όρους και τις συνθήκες που γίνεται η ανάπτυξη εμπορικού λογισμικού στην αγορά. Αυτό σημαίνει ότι δεν υπάρχει η δυνατότητα για εφαρμογή κυρίως των τελευταίων σταδίων της επικύρωσης, που αφορούν τον έλεγχο του συστήματος κάτω από συνθήκες πραγματικής λειτουργίας και από τους πραγματικούς χρήστες του. Σε κάθε περίπτωση βέβαια που δεν ήταν εφικτή η εφαρμογή των καθιερωμένων μεθόδων, έγινε προσπάθεια εξομοίωσής τους με όποιο τρόπο ήταν δυνατό.

Ένα μεγάλο μέρος του ελέγχου, όπως έχει ήδη αναφερθεί, ανήκει περισσότερο στο στάδιο της υλοποίησης παρά αποτελεί ανεξάρτητο στάδιο στα πλαίσια της ανάπτυξης. Το κομμάτι αυτό του ελέγχου έχει να κάνει πιο πολύ με την εξασφάλιση της συνέπειας του προϊόντος της υλοποίησης με αυτό που είχε αρχικά σχεδιαστεί. Ο έλεγχος αυτός έγινε τόσο σε επίπεδο εξαρτημάτων και σε επίπεδο ανεξάρτητων μηχανισμών, ώστε να επιβεβαιωθεί ότι κάθε επιμέρους κομμάτι του συστήματος είναι σε θέση να παρέχει αυτό που έχει σχεδιαστεί να παρέχει. Σημαντικό ρόλο στο

κομμάτι αυτό του ελέγχου, έπαιξε το γεγονός ότι χρησιμοποιήθηκε μεγάλος αριθμός εξαρτημάτων αλλά και ολοκληρωμένων λύσεων (π.χ. ολόκληρη η τεχνολογία πρόσβασης στη βάση δεδομένων) από το περιβάλλον της Delphi. Με τον τρόπο αυτό ένα μεγάλο κομμάτι της λειτουργικότητας βασίστηκε σε ελεγμένα και αξιόπιστα εξαρτήματα, ελαχιστοποιώντας τους πόρους που απαιτεί η διαδικασία του ελέγχου.

Το μέρος εκείνου της διαδικασίας ελέγχου και επικύρωσης του συστήματος, που αποτελεί ανεξάρτητο στάδιο είναι ο έλεγχος του αποτελέσματος της υλοποίησης τόσο ως προς τις διατυπωμένες στο καθορισμό του έργου απαιτήσεις, όσο και ως προς τις πραγματικές απαιτήσεις για το σύστημα. Το στάδιο αυτό απαιτεί τη συνεργασία των εμπλεκομένων στην ανάλυση και των πραγματικών χρηστών, που όπως αναφέρθηκε δεν ήταν διαθέσιμη στο βαθμό που θα ήταν στη περίπτωση που η ανάπτυξη θα γινόταν με τους όρους και τις συνθήκες ανάπτυξης ενός πραγματικού προϊόντος λογισμικού. Όσον αφορά το μέρος της συνέπειας ως προς την ανάλυση, σημαντικό ρόλο έπαιξε η ύπαρξη του παλαιού συστήματος ως πρότυπο αναφοράς. Με τον ίδιο τρόπο που το υπάρχον σύστημα αποτέλεσε το πρότυπο για την εξαγωγή των λειτουργικών απαιτήσεων, αποτέλεσε και το μέτρο σύγκρισης για τη λειτουργία του προϊόντος της υλοποίησης. Όσον αφορά την εκτίμηση του κατά πόσο το σύστημα μπορεί να καλύψει τις πραγματικές ανάγκες των τραπεζών, σημαντική ήταν η προσφορά του υπεύθυνου ανάπτυξης του υπάρχοντος συστήματος, που είχε το ρόλο του συμβούλου και καθοδηγητή καθ'όλη τη διάρκεια της ανάπτυξης. Με τον τρόπο αυτό έγινε μια προσπάθεια να καλυφθεί η αδυναμία ελέγχου του συστήματος από πραγματικούς χρήστες και να πιστοποιηθεί έτσι η συνέπεια του τελικού προϊόντος ως προς τους αρχικούς στόχους.

Συνοψίζοντας την όλη διαδικασία ελέγχου και επικύρωσης του συστήματος, αυτό που αξίζει να σημειωθεί είναι ο μικρός αριθμός των διορθώσεων που προτάθηκαν στα τελευταία στάδια της επικύρωσης. Το γεγονός αυτό βέβαια δεν είναι τυχαίο, αλλά αντίθετα αποδεικνύει με το καλύτερο τρόπο τη σημασία που είχε η ύπαρξη του υπάρχοντος συστήματος και η χρησιμοποίησή του σαν πρότυπο για το νέο. Πολλές ατέλειες που διαφορετικά θα φαινόταν κατά την επικύρωση του συστήματος, αποφεύχθηκαν μια και είχαν ήδη ληφθεί υπόψη από το πρότυπο που χρησιμοποιήθηκε. Σε κάθε περίπτωση αυτό που εξάγεται σαν συμπέρασμα μετά και τη διαδικασία της επικύρωσης, είναι ότι το υλοποιημένο προϊόν τηρεί τις προϋποθέσεις ως προς τη ποιότητα, όχι μόνο για να καλύψει τις ανάγκες του

λογιστηρίου, αλλά και για να αποτελέσει τη βάση της ανάπτυξης και των υπόλοιπων μερών του συστήματος των τραπεζών.

ΚΕΦΑΛΑΙΟ 4ο
ΣΥΜΠΕΡΑΣΜΑΤΑ, ΕΚΤΙΜΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΑΙ ΔΙΑΔΙΚΑΣΙΑΣ ΑΝΑΠΤΥΞΗΣ

Μετά και την επικύρωση του αποτελέσματος της υλοποίησης, έχει ολοκληρωθεί η διαδικασία ανάπτυξης του συστήματος του λογιστηρίου και έτσι είναι πλέον δυνατή η εκτίμηση τόσο του τελικού προϊόντος της προσπάθειας, όσο και όλης της διαδικασίας ανάπτυξης. Σκοπός στο σημείο αυτό δεν είναι η τεχνική εκτίμηση του αποτελέσματος, που άλλωστε έγινε κατά την επικύρωση, αλλά μια γενικότερη εκτίμηση του τι ακριβώς δημιουργήθηκε και πως μπορεί να αξιοποιηθεί. Όσον αφορά τη διαδικασία ανάπτυξης, είναι πλέον εφικτό μετά την ολοκλήρωση της προσπάθειας, να γίνει μια εκτίμηση όλων των αποφάσεων και επιλογών, που έγιναν σε κάθε στάδιο της ανάπτυξης, έτσι όπως εμφανίζονται μέσα από την επίδραση που είχαν τόσο στα επόμενα στάδια, όσο και στο τελικό αποτέλεσμα.

Αναφορικά με το τελικό προϊόν, μια πρώτη ματιά δείχνει ότι το αποτέλεσμα μπορεί και να παρεκκλίνει από τον αρχικό στόχο που είχε τεθεί για τη συγκεκριμένη προσπάθεια. Έχει σημασία να σημειωθεί ότι η ενδεχόμενη αυτή απόκλιση είναι σε σχέση με τον αρχικό στόχο και όχι με τις πραγματικές ανάγκες και για το λόγο αυτό δεν αποτελεί σε καμία περίπτωση απειλή για την επιτυχία της όλης προσπάθειας. Είναι γεγονός πως ο αρχικός στόχος για την ανάπτυξη του συστήματος ήταν η κάλυψη όλων των αναγκών του λογιστηρίου μιας τράπεζας. Βλέποντας το τελικό αποτέλεσμα, είναι φανερό το γεγονός ότι υπάρχουν κομμάτια λειτουργικότητας του υποσυστήματος του λογιστηρίου που δεν έχουν υλοποιηθεί, αν και υπάρχει το απαραίτητο πλαίσιο για την ανάπτυξή τους (π.χ. δεν έχουν δημιουργηθεί όλες οι προβολές που χρειάζονται στο λογιστήριο αλλά υπάρχει ο μηχανισμός για τη δημιουργία προβολών).

Η απόκλιση αυτή από τον αρχικό στόχο δεν είναι αποτέλεσμα κάποιας κακής εκτίμησης ή αδυναμίας κάλυψης μερών της λειτουργικότητας, αλλά συνειδητή επιλογή που ελήφθη από κοινού με το συντονιστή της προσπάθειας ανάπτυξης ολόκληρου του συστήματος. Αυτό που έγινε αντιληπτό στο σημείο εκείνο, είναι ο κίνδυνος να καλυφθεί μεν η λειτουργικότητα στο σύνολό της, αλλά σε τέτοιο επίπεδο ποιότητας και δυνατοτήτων, που ενδεχομένως ολόκληρα μέρη του συστήματος να χρειαζόταν να διορθωθούν ή και αναπτυχθούν από την αρχή κατά την επέκταση. Το γεγονός αυτό οδήγησε στην απόφαση να αναπτυχθεί τελικά ένα πλαίσιο με το επίπεδο

ποιότητας και λειτουργικότητας που κάνει δυνατή τη χρήση του από ολόκληρο το σύστημα της τράπεζας. Στη συνέχεια, χρησιμοποιώντας το πλαίσιο αυτό, θα καλυφθεί όσο το δυνατόν μεγαλύτερο μέρος της λειτουργικότητας του λογιστηρίου, δεδομένων πάντα των χρονικών ορίων για την ολοκλήρωση της προσπάθειας.

Χαρακτηριστικό παράδειγμα της φιλοσοφίας αυτής, είναι η περίπτωση των προβολών. Είναι φανερό ότι σχετικά εύκολα θα μπορούσαν να υλοποιηθούν στατικά όλες οι προβολές της λογιστικής, χωρίς φυσικά να είναι παραμετρικές και συντηρήσιμες χωρίς την επέμβαση στο πηγαίο κώδικα. Αντί αυτού, που θα ήταν εντελώς αδιάφορο λαμβάνοντας υπόψη τη συχνή ανάγκη για προσαρμογή των προβολών, δημιουργήθηκε ο μηχανισμός διαχείρισης των προβολών παραμετρικά και χωρίς καμία επέμβαση στο πηγαίο κώδικα, που μπορεί παράλληλα να χρησιμοποιηθεί και από τα υπόλοιπα μέρη του συστήματος, εκτός του λογιστηρίου. Η απόφαση αυτή είναι προφανές ότι αφαίρεσε πόρους από την κάλυψη των αναγκών του ίδιου του λογιστηρίου και τους μετέθεσε στην ανάπτυξη ενός πλαισίου που αφορά ολόκληρο το σύστημα της τράπεζας. Με τον τρόπο αυτό και μόνο ήταν δυνατό να έχει, η προσπάθεια που έγινε, αξιόλογη συνεισφορά στην ανάπτυξη ολόκληρου του συστήματος της τράπεζας.

Ένας ακόμα τομέας, στον οποίο είναι δυνατό να εξαχθούν συμπεράσματα μετά το τέλος της διαδικασίας ανάπτυξης, είναι οι επιλογές ως προς την ίδια τη διαδικασία. Από την επιλογή του μοντέλου ανάπτυξης, έως και την επιλογή των μεθόδων ελέγχου και επικύρωσης που θα χρησιμοποιηθούν, χρειάστηκε αρκετές φορές να γίνει επιλογή προσεγγίσεων, τεχνολογιών και προϊόντων ανάμεσα από ένα αριθμό εναλλακτικών λύσεων. Η πιο καθοριστική ίσως απόφαση για την έκβαση της προσπάθειας, ήταν η επιλογή του μοντέλου ανάπτυξης. Ο λόγος είναι μάλλον προφανής, αφού αφενός η επιλογή αυτή επηρέασε τη πορεία του έργου από το ξεκίνημά του και αφετέρου το μοντέλο ανάπτυξης είναι αυτό που καθορίζει τα ίδια τα στάδια της ανάπτυξης. Η επιλογή μοντέλου ανάπτυξης βασισμένου στη συναρμολόγηση και στα εξαρτήματα λογισμικού, είναι πέρα από κάθε αμφισβήτηση αυτό που έκανε δυνατή την ανάπτυξη ενός τέτοιου συστήματος, με τόσο περιορισμένους διαθέσιμους πόρους. Το γεγονός αυτό γίνεται εύκολα κατανοητό αν αντιληφθεί κανείς την επιπλέον προσπάθεια που θα χρειαζόταν για να καλυφθούν ζητήματα, όπως η αξιόπιστη πρόσβαση στη βάση δεδομένων ή η οπτικοποίηση των δεδομένων στην οθόνη, εάν για το σκοπό αυτό δεν είχαν χρησιμοποιηθεί έτοιμα εξαρτήματα λογισμικού. Λαμβάνοντας υπόψη τα παραπάνω, η απόφαση για την

εφαρμογή ενός τέτοιου μοντέλου χαρακτηρίζεται όχι μόνο θετική, αλλά απαραίτητη για τη τελική επιτυχία της προσπάθειας.

Μια ακόμα σημαντική απόφαση με αντίκτυπο στο τελικό αποτέλεσμα, ήταν η επιλογή του να βασιστεί το νέο σύστημα σε ένα ήδη υπάρχον. Είναι γεγονός, ότι το υπάρχον σύστημα καλύπτει από πλευράς λειτουργικότητας τις ανάγκες των σύγχρονων τραπεζών. Επίσης είναι σαφές, ότι αυτό έχει επιτευχθεί μετά από μακροχρόνια εργασία, μέχρις ότου φτάσει το σύστημα στη σημερινή του μορφή. Επομένως, εύκολα συμπεραίνει κανείς ότι θα ήταν μάλλον ανέφικτος ο πληρέστερος καθορισμός των απαιτήσεων με μια νέα προσπάθεια που θα ξεκινούσε από το μηδέν στα πλαίσια του συγκεκριμένου εγχειρήματος. Αυτή η αρχική σκέψη ενισχύθηκε σε μεγάλο βαθμό κατά τη διαδικασία της ανάπτυξης, εξαιτίας των πολύ καθοριστικών υπηρεσιών που προσέφερε ύπαρξη μιας λειτουργικής αναπαράστασης των απαιτήσεων, η οποία χρησιμοποιήθηκε σαν πρότυπο σε αρκετά στάδια της ανάπτυξης. Η επιβεβαίωση για την επιλογή αυτή ήταν τα αποτελέσματα της διαδικασίας επικύρωσης του συστήματος, που ουσιαστικά ήταν η αποδοχή του συστήματος όπως ακριβώς βγήκε από την υλοποίηση, κάτι που αποδεικνύει ότι η προσπάθεια καθορισμού του έργου είχε στηριχθεί στα σωστά θεμέλια.

Τέλος ολοκληρώνοντας την εκτίμηση των αποτελεσμάτων, κάτι που πρέπει να αναφερθεί, είναι η συνεισφορά της Delphi στη τελική επιτυχία της προσπάθειας. Όπως είχε τονιστεί από την περιγραφή ακόμα των μοντέλων ανάπτυξης, μια σημαντική προϋπόθεση για την επιτυχία της ανάπτυξης λογισμικού με συναρμολόγηση, είναι η ύπαρξη ενός σημαντικού αριθμού αξιόπιστων εξαρτημάτων λογισμικού. Αυτό που αποδείχτηκε κατά την ανάπτυξη είναι ότι δεν αρκεί μόνο το περιβάλλον προγραμματισμού να παρέχει τα εξαρτήματα αυτά, αλλά να υποστηρίζει στο σύνολο του μια προσέγγιση ανάπτυξης μέσω εξαρτημάτων. Αυτό σημαίνει, ότι θα πρέπει να παρέχει και όλους τους μηχανισμούς για εργασία με εξαρτήματα, όπως η δυνατότητα ανάπτυξης εξαρτημάτων κληρονομώντας από τα ήδη υπάρχοντα, η δυνατότητα προσθήκης εξαρτημάτων στο περιβάλλον και γενικότερα μια δομή που συνολικά λαμβάνει υπόψη της την ανάπτυξη με εξαρτήματα λογισμικού. Είναι σαφές ότι η ύπαρξη ενός εργαλείου τόσο εξειδικευμένου σε αυτή τη προσέγγιση ανάπτυξης λογισμικού όσο η Delphi, έπαιξε ιδιαίτερα σημαντικό ρόλο στην αβίαστη μεταφορά του προϊόντος της σχεδίασης σε εκτελέσιμο πρόγραμμα.

ΠΑΡΑΡΤΗΜΑ Α
ΚΑΘΟΡΙΣΜΟΣ ΟΝΤΟΤΗΤΩΝ ΤΗΣ ΛΟΓΙΣΤΙΚΗΣ

1.Λογαριασμοί λογιστικής

Περιγραφή: Η οντότητα που αναπαριστά ένα λογαριασμό λογιστικής
Ονομασία: GLM

Περιγραφή πεδίου	**Παρατηρήσεις**	**Ονομασία πεδίου**	**Τύπος δεδομένων**	**Μήκος**
Λογαριασμός	Ο κωδικός του λογαριασμού που έχει τη μορφή 00.00.00.00.00000 . Υπάρχουν λογαριασμοί 5 επιπέδων (οι 00. είναι πρώτου επιπέδου , οι 00.00. δεύτερου κ.τ.λ.). Για να υπάρχει ένας λογαριασμός θα πρέπει να υπάρχουν και οι προηγούμενοι του (π.χ. για να υπάρχει ο 30.00. θα πρέπει να υπάρχει και ο 30.).	Account	Κείμενο	20
Τίτλος λογαριασμού	Η περιγραφή του λογαριασμού	Title	Κείμενο	255
Λογαριασμός πελάτη	Εάν ο λογαριασμός αναφέρεται σε κάποιον πελάτη, εδώ συμπληρώνεται ο κωδικός του. Το πεδίο αυτό δεν το διαχειρίζεται το υποσύστημα της λογιστικής	Customer	Κείμενο	10
Δέχεται εγγραφές	Συμπληρώνεται με «Ν» όταν ο λογαριασμός δέχεται εγγραφές και με «Ο» όταν δεν δέχεται. Αν κάποιος λογαριασμός δέχεται εγγραφές τότε δεν μπορεί να έχει υπολογαριασμούς. Οι λογαριασμοί πρώτου επιπέδου δεν δέχονται ποτέ εγγραφές και οι λογαριασμοί πέμπτου επιπέδου δέχονται πάντα.	Accepts_Records	Κείμενο	1
Αναλυτικές / Συγκεντρωτικές	Το πεδίο αυτό συμπληρώνεται με «Α» όταν ο λογαριασμός δέχεται αναλυτικές εγγραφές και με «Σ» όταν δέχεται συγκεντρωτικές. Συμπληρώνεται μόνο όταν ο λογαριασμός δέχεται εγγραφές.	Account_Type	Κείμενο	1
Έλεγχος κινήσεων	Το πεδίο αυτό συμπληρώνεται με «Χ» όταν απαιτείται έλεγχος στις χορηγήσεις, με «Κ» όταν πρόκειται για καταθέσεις, με «Μ» για μερίδες και με «Ο» όταν δεν είναι επιθυμητός ο έλεγχος. Και το πεδίο αυτό συμπληρώνεται μόνο όταν ο λογαριασμός δέχεται εγγραφές.	Record_Check	Κείμενο	1
Ημερομηνία ανοίγματος	Η ημερομηνία δημιουργίας του λογαριασμού.	Creation_Date	Ημερομηνία	8
Προστασία	Συμπληρώνεται με «Π» για προστασία του λογαριασμού από πίστωση, με «Χ» για προστασία του λογαριασμού από χρέωση και με «Ο» για καμία προστασία. Έχει νόημα μόνο όταν ο λογαριασμός δέχεται εγγραφές.	Protection	Κείμενο	1
Σύνδεση	Συμπληρώνεται με «Ε» για σύνδεση με πελάτη, με «Π» για σύνδεση με προμηθευτή και με «Ο» για καμία σύνδεση. Και το πεδίο αυτό έχει νόημα όταν ο λογαριασμός δέχεται εγγραφές.	Link	Κείμενο	1
Λογαριασμός σύνδεσης	Ο λογαριασμός του πελάτη ή του προμηθευτή με τον οποίο συνδέεται ο λογαριασμός. Συμπληρώνεται μόνο όταν ο λογαριασμός έχει κάποια σύνδεση. Ο λογαριασμός σύνδεσης είναι	Link_Account	Κείμενο	5

	συνήθως τα πέντε τελευταία ψηφία του λογαριασμού και αυτό θα πρέπει να προτείνεται στο χρήστη.			
Βαθμός εκτύπωσης λογιστικού σχεδίου	Συμπληρώνεται μόνο στους λογαριασμούς πρώτου επιπέδου με ένα ακέραιο αριθμό από το 0 έως το 5 και δηλώνει το επίπεδο λογαριασμών στο οποίο θα φτάνει η εκτύπωση του λογαριασμού αυτού (π.χ. αν επιλεγεί το 2 τότε στις προβολές του λογαριασμού αυτού θα φαίνονται όλοι οι υπολογαριασμοί του μέχρι και το 2ο επίπεδο).	L_P_Print_Level	Ακέραιος	4
Βαθμός εκτύπωσης ισοζυγίου	Ο βαθμός εκτύπωσης ισοζυγίου για το συγκεκριμένο λογαριασμό που συμπληρώνεται από έναν ακέραιο αριθμό από 0 έως 5. Το πεδίο συμπληρώνεται από το υπό ανάπτυξη σύστημα αλλά η διαχείρισή του δεν γίνεται από αυτό και ισχύει μόνο για λογαριασμούς πρώτου επιπέδου.	Balance_Print_Level	Ακέραιος	4
Οι αξίες περιέχουν ΦΠΑ	Συμπληρώνεται με ένα ακέραιο αριθμό που δηλώνει το ποσοστό ΦΠΑ που θα περιέχουν οι αξίες που συμπληρώνονται στο λογαριασμό αυτό. Έχει νόημα μόνο για λογαριασμούς πρώτου επιπέδου.	Values_Include_Tax	Ακέραιος	4
Ημερομηνία τελευταίας κίνησης	Στο πεδίο αυτό συμπληρώνεται η ημερομηνία της τελευταίας κίνησης στους λογαριασμούς που δέχονται εγγραφές.	Last_Record_Date	Ημερομηνία	8
Νόμισμα	Στο πεδίο αυτό συμπληρώνεται το νόμισμα των αξιών του λογαριασμού. Το νόμισμα δηλώνεται με ένα κωδικό που ορίζεται στην οντότητα των νομισμάτων.	Currency	Κείμενο	3
Παρατηρήσεις 1ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 1ο επίπεδο. Ο κάθε χρήστης έχει δικαίωμα να διαχειρίζεται τις παρατηρήσεις που είναι μικρότερου ή ίσου επιπέδου με το δικό του.	Notes_Level_1	Κείμενο	255
Παρατηρήσεις 2ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 2ο επίπεδο.	Notes_Level_2	Κείμενο	255
Παρατηρήσεις 3ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 3ο επίπεδο.	Notes_Level_3	Κείμενο	255
Παρατηρήσεις 4ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 4ο επίπεδο.	Notes_Level_4	Κείμενο	255
Παρατηρήσεις 5ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 5ο επίπεδο.	Notes_Level_5	Κείμενο	255
Παρατηρήσεις 6ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 6ο επίπεδο.	Notes_Level_6	Κείμενο	255
Παρατηρήσεις 7ου επιπέδου	Οι παρατηρήσεις που συμπληρώνονται από χρήστες που ανήκουν στο 7ο επίπεδο.	Notes_Level_7	Κείμενο	255

2. Κινήσεις λογαριασμών

Περιγραφή: Η οντότητα που αναπαριστά την κίνηση ενός λογαριασμού
Ονομασία: GLK

Περιγραφή πεδίου	**Παρατηρήσεις**	**Ονομασία πεδίου**	**Τύπος δεδομένων**	**Μήκος**
Κατάστημα	Δηλώνει το υποκατάστημα της τράπεζας που πραγματοποιήθηκε η κίνηση. Το κατάστημα συμπληρώνεται ο αντίστοιχος κωδικός.	Branch	Κείμενο	3
Παραστατικό	Δηλώνει το παραστατικό της κίνησης	Voucher	Κείμενο	11
Πράξη	Δηλώνει τη πράξη που αντιστοιχεί στη κίνηση. Η πράξη συμπληρώνεται με κωδικό που ορίζεται από	GLK_Action	Κείμενο	3

	την οντότητα των πράξεων λογιστικής.			
Λογαριασμός	Είναι ο λογαριασμός στον οποίο γίνεται η χρέωση ή η πίστωση. Στο πεδίο αυτό μπορούν να χρησιμοποιηθούν μόνο λογαριασμοί που δέχονται εγγραφές και δεν έχουν σχετική προστασία.	Account	Κείμενο	20
Τίτλος λογαριασμού	Είναι ο τίτλος του λογαριασμού που συμπληρώνεται στο προηγούμενο πεδίο. Αρχικά προτείνεται ο τίτλος του λογαριασμού που υπάρχει ήδη στο GLM αλλά δίνεται στο χρήστη η δυνατότητα να το αλλάξει και γι'αυτό υπάρχει το πεδίου του τίτλου στην οντότητα αυτή.	Account_Title	Κείμενο	255
Αιτιολογία	Είναι η αιτιολογία της κίνησης. Αν και υπάρχει ανεξάρτητη οντότητα αιτιολογιών, η αιτιολογία δεν συμπληρώνεται μέσω κωδικού αλλά ολογράφως. Αυτό γιατί οι κωδικοποιημένες αιτιολογίες δεν είναι δεσμευτικές και μπορεί να τις επεξεργάζεται ο χρήστης κατά περίπτωση.	Reason	Κείμενο	255
Δραστηριότητα	Είναι η δραστηριότητα της κίνησης. Συμπληρώνεται με κωδικό που ορίζεται στην οντότητα των δραστηριοτήτων.	Activity	Κείμενο	3
Αύξων αριθμός	Είναι ένας ακέραιος που δηλώνει τον αύξοντα αριθμό μιας συγκεκριμένης εγγραφής μέσα σε μια κίνηση.	GLK_index	Ακέραιος	4
Ημερομηνία κίνησης	Είναι η ημερομηνία που πραγματοποιήθηκε η κίνηση.	GLK_Date	Ημερομηνία	9
Χρέωση	Δηλώνει το ποσό της χρέωσης στο λογαριασμό. Κάθε εγγραφή της κίνησης μπορεί να είναι μόνο χρεωστική ή μόνο πιστωτική. Το συνολικό πόσο χρέωσης και πίστωσης σε μια κίνηση θα πρέπει να είναι ίσο για να είναι η κίνηση αποδεκτή.	Debit	Πραγματικός αριθμός	9
Πίστωση	Δηλώνει το ποσό της πίστωσης στο λογαριασμό	Credit	Πραγματικός αριθμός	9
Χρήστης	Συμπληρώνεται με το κωδικό του χρήστη που εισάγει τη κίνηση.	GLK_User	Κείμενο	3
Νόμισμα	Είναι το νόμισμα των ποσών στη κίνηση. Προς το παρόν θα έρχεται από το αντίστοιχο πεδίο του λογαριασμού, αλλά θα δοθεί στο μέλλον η δυνατότητα αλλαγής.	Currency	Κείμενο	3
Ημερομηνία και ώρα κίνησης	Η ημερομηνία και η ώρα που εισάγεται η κίνηση	Record_Date	Ημερομηνία	8

3. Δραστηριότητες λογιστικής

Περιγραφή: Η οντότητα που αναπαριστά τις δραστηριότητες της λογιστικής
Ονομασία: Activities

Περιγραφή πεδίου	**Παρατηρήσεις**	**Ονομασία πεδίου**	**Τύπος δεδομένων**	**Μήκος**
Κωδικός δραστηριότητας	Ένας τριψήφιος κωδικός που δηλώνει μια δραστηριότητα.	ID	Κείμενο	3
Περιγραφή	Η αναλυτική περιγραφή της δραστηριότητας.	Description	Κείμενο	255

4. Αιτιολογίες

Περιγραφή: Η οντότητα που αναπαριστά τις αιτιολογίες
Ονομασία: Reasons

Περιγραφή πεδίου	**Παρατηρήσεις**	**Ονομασία πεδίου**	**Τύπος δεδομένων**	**Μήκος**
Κωδικός αιτιολογίας	Ένας τριψήφιος κωδικός που δηλώνει μια αιτιολογία. Οι αιτιολογίες δεν είναι δεσμευτικές όπως οι δραστηριότητες. Στις οντότητες που περιέχουν αιτιολογία δεν αποθηκεύεται ο κωδικός αλλά ολόκληρη η περιγραφή αφού οι αιτιολογίες είναι απλά προτάσεις στο χρήστη που έχουν κωδικοποιηθεί για επιτάχυνση της εργασίας των χρηστών.	ID	Κείμενο	3
Περιγραφή	Η αναλυτική περιγραφή της δραστηριότητας.	Description	Κείμενο	255

5. Πράξεις λογιστικής

Περιγραφή: Η οντότητα που αναπαριστά τις πράξεις της λογιστικής
Ονομασία: Actions

Περιγραφή πεδίου	**Παρατηρήσεις**	**Ονομασία πεδίου**	**Τύπος δεδομένων**	**Μήκος**
Κωδικός πράξης	Ένας τριψήφιος κωδικός που δηλώνει μια πράξη.	ID	Κείμενο	3
Περιγραφή	Η αναλυτική περιγραφή της δραστηριότητας.	Description	Κείμενο	255

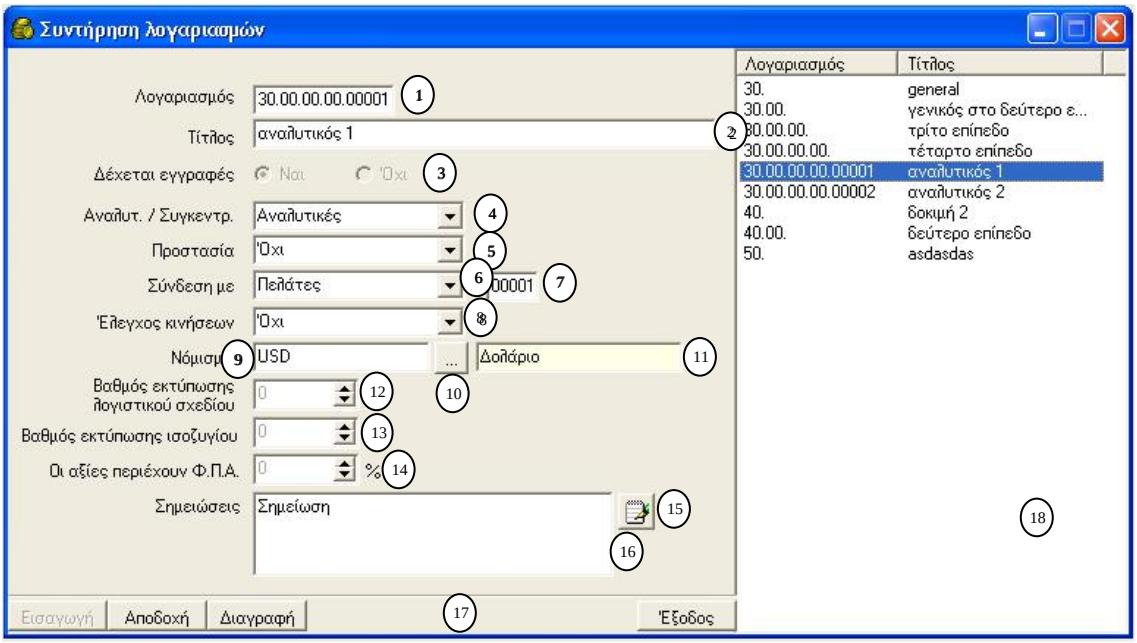
6. Νομίσματα

Περιγραφή: Η οντότητα που αναπαριστά τα νομίσματα
Ονομασία: Currency

Περιγραφή πεδίου	**Παρατηρήσεις**	**Ονομασία πεδίου**	**Τύπος δεδομένων**	**Μήκος**
Κωδικός νομίσματος	Ένας τριψήφιος κωδικός που δηλώνει ένα νόμισμα	ID	Κείμενο	3
Περιγραφή	Η αναλυτική περιγραφή της δραστηριότητας.	Description	Κείμενο	255
Ισοτιμία με το ευρώ	Η ισοτιμία του νομίσματος με το ευρώ.	Exchange_Rate	Πραγματικός αριθμός	9

ΠΑΡΑΡΤΗΜΑ Β
ΠΕΡΙΓΡΑΦΗ ΓΡΑΦΙΚΩΝ ΔΙΕΠΑΦΩΝ

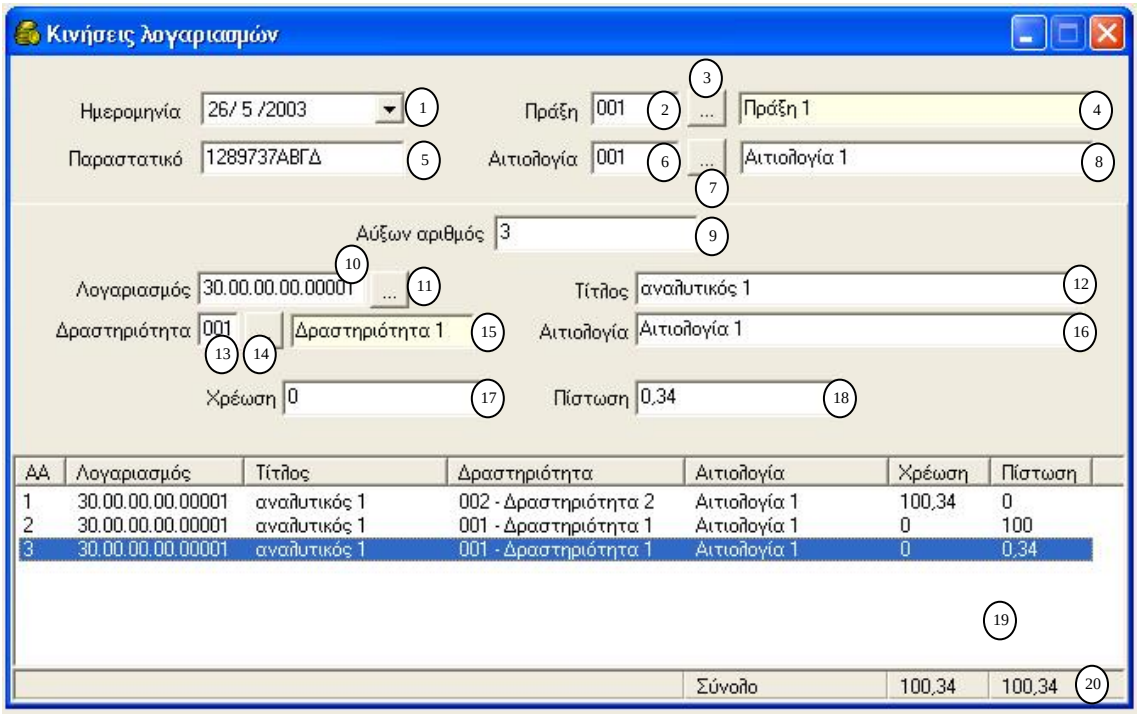
1. Γραφικές διεπαφές του υποσυστήματος της λογιστικής
1.1 Διαχείριση λογαριασμών



1. Αντικείμενο τύπου TAccountMaskEdit. Είναι ένα από τα δύο εξαρτήματα που αναπτύχθηκαν για τις ανάγκες του συστήματος και χρησιμοποιείται για τη διαχείριση των λογαριασμών.
2. Αντικείμενο τύπου TEdit που χρησιμοποιείται για τη διαχείριση του τίτλου του λογαριασμού
3. Δύο αντικείμενα τύπου TRadioButton για την επιλογή σχετικά με το αν ο λογαριασμός δέχεται εγγραφές
4. Αντικείμενο TCombobox με επιλογές τις πιθανές τιμές για το πεδίο Accepts_Records (αν δέχεται εγγραφές ή όχι) του πίνακα GLM
5. Ομοίως με το 4 αλλά για το πεδίο Protection (προστασία) του πίνακα GLM
6. Ομοίως με το 4 αλλά για το πεδίο Link (σύνδεση) του πίνακα GLM
7. Αντικείμενο τύπου TMaskEdit που δέχεται το κωδικό του πελάτη ή του προμηθευτή αν έχει επιλεγεί η αντίστοιχη σύνδεση
8. Ομοίως με το 4 αλλά για το πεδίο Record_Check (έλεγχος κινήσεων) του πίνακα GLM
9. Αντικείμενο τύπου TMaskEdit που δέχεται το κωδικό του νομίσματος
10. Αντικείμενο τύπου TSpeedButton που καλεί όταν πατηθεί το μηχανισμό των προβολών
11. Αντικείμενο τύπου TEdit για την εμφάνιση της περιγραφής του νομίσματος
12. Αντικείμενο τύπου TSpinEdit που επιτρέπει την εισαγωγή μόνο ακεραίων αριθμών. Χρησιμποιείται για την εισαγωγή του βαθμού εκτύπωσης λογιστικού σχεδίου
13. Ομοίως με το 12 για το βαθμό εκτύπωσης ισοζυγίου
14. Ομοίως με το 12 για το ποσοστό ΦΠΑ που περιέχουν οι αξίες

15. Αντικείμενο τύπου TMemo που διαχειρίζεται της σημειώσεις του επιπέδου που ανήκει ο χρήστης
16. Αντικείμενο τύπου TSpeedButton που εμφανίζει τη φόρμα TfrmNotes για τη διαχείριση όλων των σημειώσεων που επιτρέπεται να διαχειριστεί ο χρήστης (όλα τα επίπεδα που είναι μικρότερα από το δικό του).
17. Αντικείμενο τύπου TToolBar όπου ανήκουν τα αντίστοιχα TToolButtons για την εισαγωγή, αποδοχή, διαγραφή και έξοδο από τη φόρμα.
18. Αντικείμενο τύπου TListView για την εμφάνιση των λογαριασμών

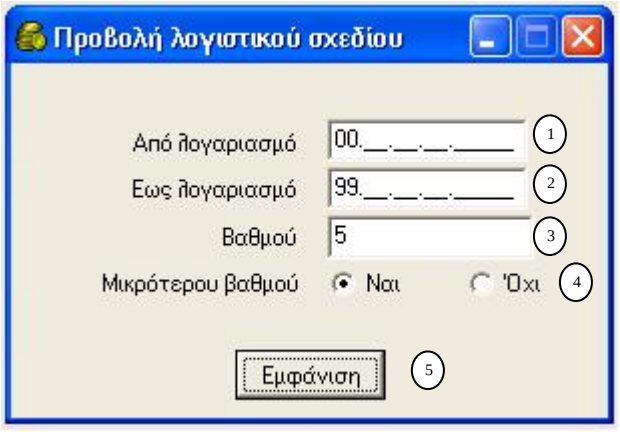
1.2 Κινήσεις λογαριασμών



1. Αντικείμενο τύπου TDateTimePicker που επιτρέπει την επιλογή της ημερομηνίας για τη κίνηση.
2. Αντικείμενο τύπου TMaskEdit για την εισαγωγή του κωδικού της πράξης
3. Αντικείμενο τύπου TSpeedButton για την κλήση του μηχανισμού των προβολών με σκοπό τη δημιουργία προβολής βοήθειας για το κωδικό της πράξης.
4. Αντικείμενο τύπου TEdit για τη προβολή της περιγραφής της πράξης.
5. Αντικείμενο τύπου TMaskEdit για την εισαγωγή του παραστατικού
6. Ομοίως με το 2 για το κωδικό της αιτιολογίας
7. Ομοίως με το 3 για τις αιτιολογίες
8. Ομοίως με το 2 για τη περιγραφή της αιτιολογίας
9. Αντικείμενο τύπου TSpinEdit για την εισαγωγή ακεραίου αριθμού που αντιστοιχεί στον αύξοντα αριθμό της εγγραφής
10. Αντικείμενο TAccountMaskEdit για την εισαγωγή του λογαριασμού
11. Αντικείμενο τύπου TSpeedButton τη δημιουργία προβολής με σκοπό την επιλογή λογαριασμού
12. Αντικείμενο τύπου TEdit για την εμφάνιση και αλλαγή του τίτλου του λογαριασμού

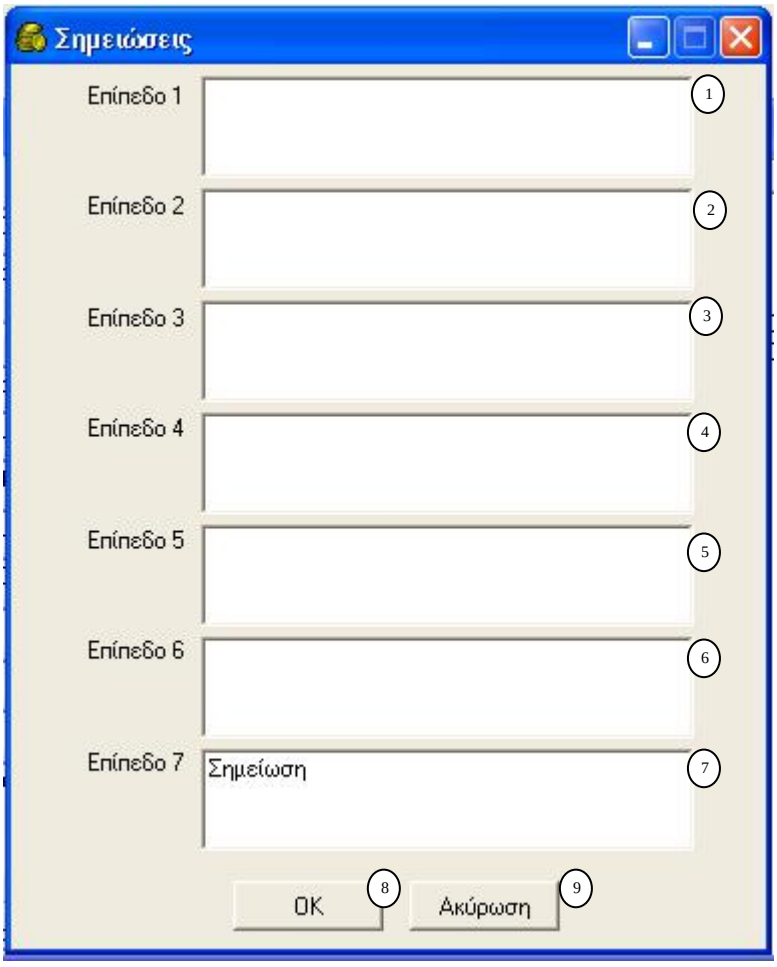
13. Ομοίως με το 2 για το κωδικό της δραστηριότητας
14. Ομοίως με το 2 για την επιλογή δραστηριότητας
15. Ομοίως με το 2 για την περιγραφή της δραστηριότητας
16. Αντικείμενο TEdit για αλλαγή της αιτιολογίας στη συγκεκριμένη εγγραφή
17. Αντικείμενο τύπου TLimitSpinEdit για την εισαγωγή του ποσού χρέωσης. Είναι το δεύτερο εξάρτημα που αναπτύχθηκε για τις ανάγκες του συγκεκριμένου συστήματος.
18. Αντικείμενο τύπου TLimitSpinEdit για εισαγωγή του ποσού πίστωσης.
19. Αντικείμενο τύπου TListView για τη προβολή των εγγραφών της παρούσας κίνησης.
20. Αντικείμενο τύπου TListView με μία μόνο γραμμή για τη προβολή των συνόλων χρέωσης και πίστωσης

1.3 Προβολή λογιστικού σχεδίου.



1. Αντικείμενο τύπου TAccountMaskEdit που χρησιμοποιείται για την εισαγωγή του λογαριασμού από όπου θα ξεκινάει η προβολή του λογιστικού σχεδίου.
2. Αντικείμενο τύπου TAccountMaskEdit που χρησιμοποιείται για την εισαγωγή του λογαριασμού όπου θα σταματάει η προβολή.
3. Αντικείμενο τύπου TSpinEdit για την εισαγωγή ενός ακεραίου έως το 5 που δηλώνει το βαθμό των λογαριασμών που θα περιλαμβάνει η προβολή.
4. Αντικείμενα τύπου TRadioButton για την επιλογή του αν θα προβάλλονται ή όχι οι λογαριασμοί με επίπεδο μικρότερο από αυτό που δηλώθηκε πιο πάνω.
5. Αντικείμενο τύπου TBitButton για τη δημιουργία των κριτηρίων και τη κλήση του μηχανισμού των προβολών

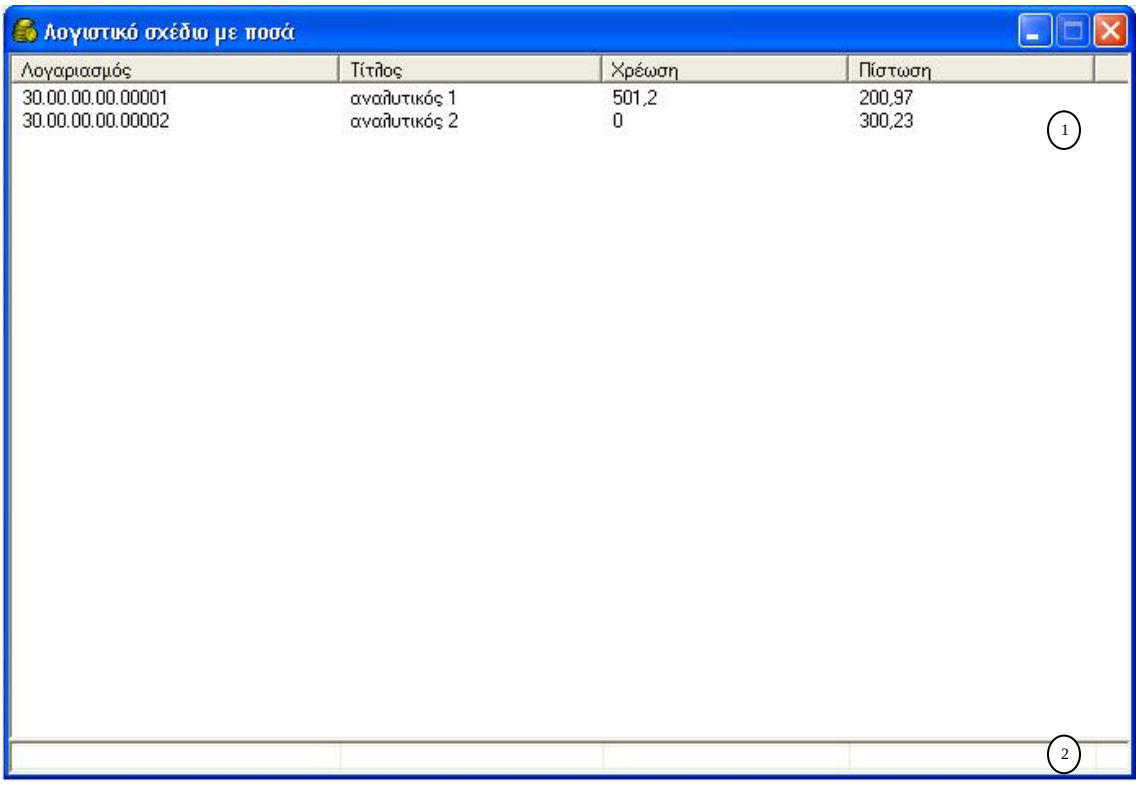
1.4 Διαχείριση σημειώσεων



1. Αντικείμενο τύπου TMemo για την εισαγωγή και προβολή των σημειώσεων του πρώτου επιπέδου χρηστών.
2. Ομοίως με το 1 για το δεύτερο επίπεδο.
3. Ομοίως με το 1 για το τρίτο επίπεδο.
4. Ομοίως με το 1 για το τέταρτο επίπεδο.
5. Ομοίως με το 1 για το πέμπτο επίπεδο.
6. Ομοίως με το 1 για το έκτο επίπεδο.
7. Ομοίως με το 1 για το έβδομο επίπεδο.
8. Αντικείμενο τύπου TBitButton για την αποδοχή των αλλαγών και επιστροφή στη φόρμα των λογαριασμών.
9. Αντικείμενο τύπου TBitButton για επιστροφή στην φόρμα των λογαριασμών χωρίς αποδοχή των αλλαγών.

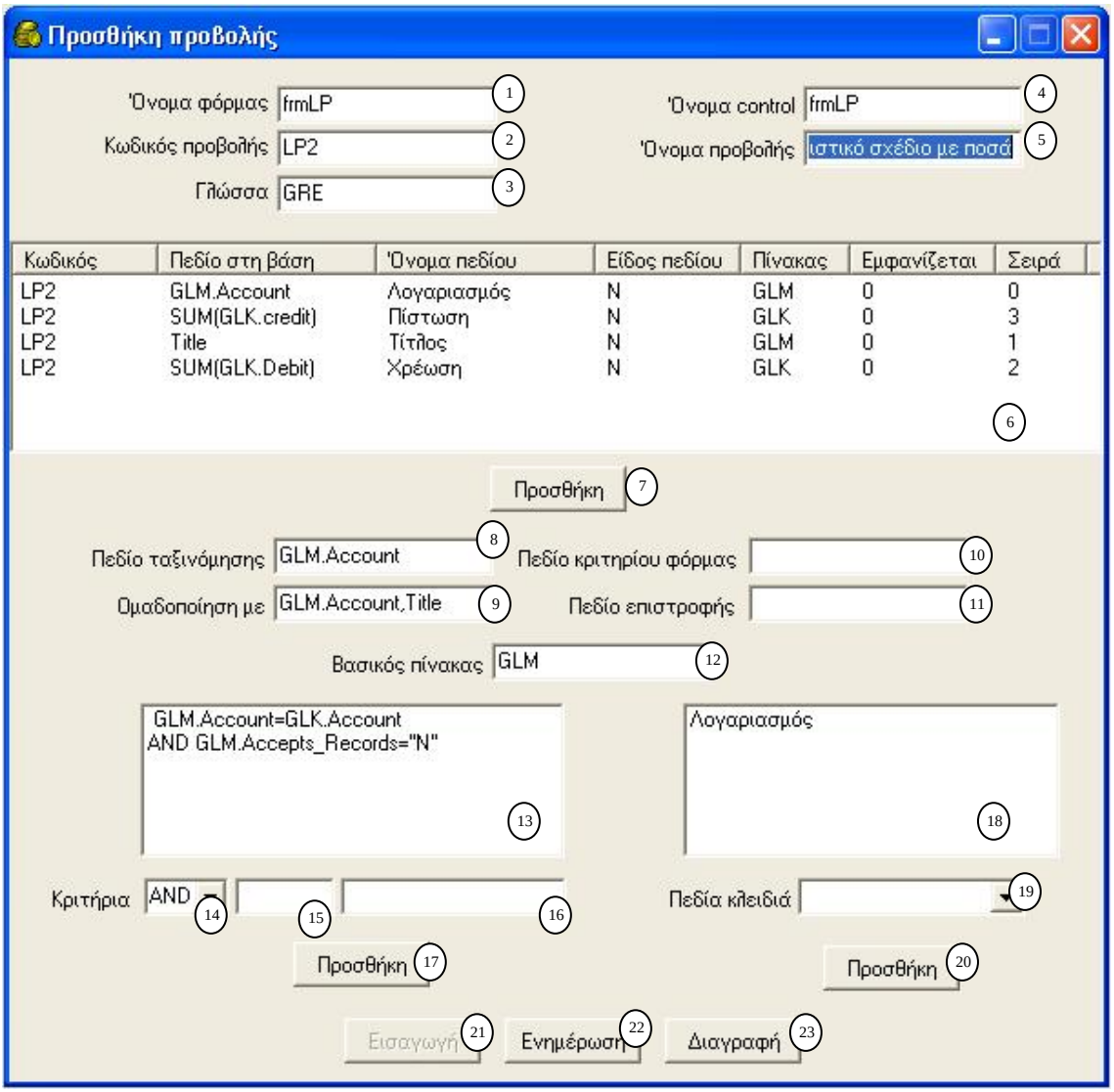
2. Γραφικές διεπαφές του υποσυστήματος των προβολών

2.1 Η φόρμα των προβολών



1. Αντικείμενο τύπου TListView για την εμφάνιση των δεδομένων της προβολής.
2. Αντικείμενο τύπου TListView με μία μόνο γραμμή και στήλες ίδιες με το 1 για τη προβολή αθροισμάτων ή γενικότερα πράξεων πάνω στις στήλες.

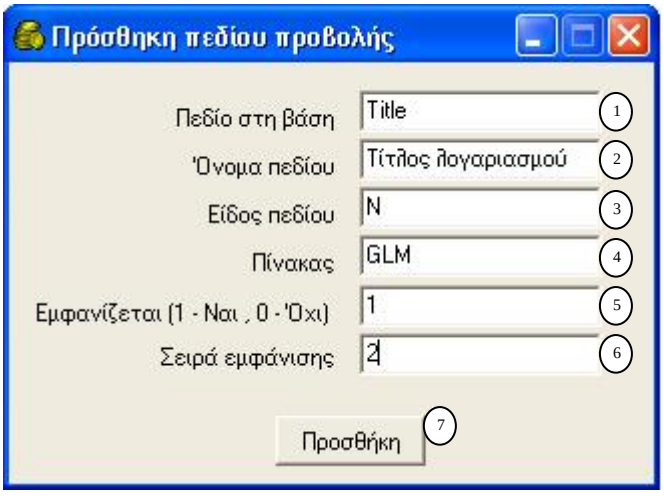
2.2 Διεπαφή συντήρησης των προβολών



1. Αντικείμενο τύπου TEdit για εισαγωγή του ονόματος της φόρμας στην οποία θα ανήκει η προβολή.
2. Ομοίως με το 1 για το αντικείμενο της φόρμας στο οποίο αναφέρεται η προβολή.
3. Ομοίως με το 1 για το κωδικό της προβολής.
4. Ομοίως με το 1 για το όνομα της προβολής.
5. Ομοίως με το 1 για τον κωδικό της γλώσσας.
6. Αντικείμενο τύπου TListView για την εμφάνιση των πεδίων που έχουν καταχωρηθεί στη προβολή.
7. Αντικείμενο τύπου TBitButton που εμφανίζει τη φόρμα προσθήκης πεδίων στη προβολή.
8. Ομοίως με το 1 για τα πεδία ταξινόμησης.
9. Ομοίως με το 1 για τα πεδία ομαδοποίησης.
10. Ομοίως με το 1 για το πεδίο της προβολής με το οποίο συγκρίνονται τα κριτήρια τα οποία εισάγονται σαν όρισμα στο μηχανισμό προβολών όταν η προβολή είναι βοήθεια για τη συμπλήρωση κάποιου κωδικού.

11. Ομοίως με το 1 για το πεδίο που επιστρέφεται όταν η προβολή είναι βοήθεια για τη συμπλήρωση κάποιου κωδικού.
12. Ομοίως με το 1 για το βασικό πίνακα της προβολής.
13. Αντικείμενο τύπου TListBox που εμφανίζονται τα κριτήρια της προβολής.
14. Αντικείμενο τύπου TComboBox που περιλαμβάνει σαν επιλογές τους τρόπους σύνδεσης μεταξύ των κριτηρίων (AND, OR κ.τ.λ.).
15. Αντικείμενο τύπου TEdit για την εισαγωγή του πρώτου μέρους του κριτηρίου (π.χ. GLM.Accepts_Records=).
16. Ομοίως με το 15 για το μέρος του κριτηρίου.
17. Αντικείμενο τύπου TBitButton για τη προσθήκη ενός κριτηρίου στο 13.
18. Ομοίως με το 13 για τα πεδία κλειδιά της προβολής.
19. Αντικείμενο τύπου TComboBox με επιλογές τα πεδία της προβολής, για την επιλογή ενός πεδίου ως κλειδί.
20. Αντικείμενο TBitButton για την προσθήκη στο 18 του επιλεγμένου στο 19 πεδίου.
21. Αντικείμενο τύπου TBitButton για την εισαγωγή της προβολής.
22. Ομοίως με το 21 για αποδοχή των αλλαγών.
23. Ομοίως με το 21 για διαγραφή της προβολής.

2.3 Προσθήκη πεδίου στη προβολή



1. Αντικείμενο τύπου TEdit για την εισαγωγή της ονομασίας του πεδίου στη βάση.
2. Ομοίως με το 1 για την ονομασία του πεδίου στη προβολή.
3. Ομοίως με το 1 για το είδος του πεδίου.
4. Ομοίως με το 1 για το πίνακα όπου ανήκει το πεδίο στη βάση.
5. Ομοίως με το 1 για το αν θα εμφανίζεται το πεδίο ή όχι (μη ορατά πεδία)
6. Ομοίως με το 1 για τη σειρά εμφάνισης του πεδίου.
7. Αντικείμενο τύπου TBitButton για επιστροφή στη φόρμα συντήρησης των προβολών και προσθήκη του πεδίου στη προβολή.

ΠΑΡΑΡΤΗΜΑ Γ
Ο ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

1. Εξαρτήματα λογισμικού
1.1 Το εξάρτημα διαχείρισης λογαριασμών (TAccountMaskEdit)

```
unit AccountMaskEdit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Mask;

type
  TAccountMaskEdit = class(TCustomMaskEdit)
  private
    { Private declarations }
    FExistParentAccount : bool;
    FAccount : string;
    FEditorEnabled : bool;
    FNotExistParentAccountMessage : string ;
    function GetAccount():string;
    procedure CMEnter(var Message: TCMEnter); message CM_ENTER;
    procedure WMLButtonDown(var Message: TWMLButtonDown); message
WM_LBUTTONDOWN;
  protected
    { Protected declarations }
    procedure KeyDown(var Key: Word; Shift: TShiftState); override;
    procedure KeyPress(var Key: Char); override;
  public
     constructor Create(AOwner: TComponent); override;
  published
    { Published declarations }
    property Account: string read GetAccount;
    property ExistParentAccount: bool read FExistParentAccount write
FExistParentAccount;
    property EditorEnabled: bool read FEditorEnabled write
FEditorEnabled;
    property NotExistParentAccountMessage :string read
FNotExistParentAccountMessage write FNotExistParentAccountMessage;
    property Anchors;
    property AutoSelect;
    property AutoSize;
    property BiDiMode;
    property BorderStyle;
    property CharCase;
    property Color;
    property Constraints;
    property Ctl3D;
    property DragCursor;
    property DragKind;
    property DragMode;
    property Enabled;
    property Font;
    property ImeMode;
    property ImeName;
    property MaxLength;
    property ParentBiDiMode;
```

```
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PasswordChar;
    property PopupMenu;
    property ReadOnly;
    property ShowHint;
    property TabOrder;
    property TabStop;
    property Text;
    property Visible;
    property OnChange;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDock;
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnStartDock;
    property OnStartDrag;
  end;

procedure Register;

implementation

var hlpKey46 :bool;

constructor TAccountMaskEdit.Create(AOwner: TComponent);
begin
  inherited;

  EditMask := '99.99.99.99.99999;1;_';
  Width := 100;
  FExistParentAccount := true;
  FEditorEnabled := true;
  FNotExistParentAccountMessage := 'Ο προηγούμενος λογαριασμός είτε
δεν υπάρχει είτε δεν δέχεται εγγραφές!';

end;

function TAccountMaskEdit.GetAccount():string;
var
  tmpStr:string;
begin
  tmpStr := Text;
  delete(tmpStr,pos(' ',tmpStr),length(tmpStr));
  FAccount := tmpStr;
  GetAccount := tmpStr;
end;
```

```
procedure TAccountMaskEdit.KeyDown(var Key: Word; Shift:
TShiftState);
var
  tmpText :string;
begin
  if EditorEnabled then
  begin
  if (key <> 37) then
  begin
    GetAccount;
    hlpKey46 := False;

    if (key = 46) then hlpKey46 := true;

    if (key = 8) then
    begin
      GetAccount;
      if (Length(FAccount) = SelStart) then inherited
      else if (Length(FAccount)-1 = SelStart) and ((SelStart = 2) or
(SelStart = 5) or (SelStart = 8) or (SelStart = 11) and
(Length(FAccount) = SelStart)) then inherited;
    end
    else inherited;
  end;

  if (hlpkey46) then
  begin
    if (Length(FAccount)  <> (SelStart)) then
      if (Text[SelStart+1] <> '.') or (Text[SelStart + 2] <> ' ')
then
      begin
        Text := FAccount;
        GetAccount;
        SelStart := Length(FAccount);
      end;
    hlpKey46 := False;
  end;
  end
  else
  begin
    tmpText := Text;
    inherited;
    Text := tmpText;
  end;
end;

procedure TAccountMaskEdit.KeyPress(var Key: Char);
var
  tmpText :string;
begin
  tmpText := Text;
  if EditorEnabled then
  begin
    if (key > chr(47)) and (key < chr(58)) then
    begin
      inherited;
      if (Length(GetAccount) < SelStart) then
      begin
        GetAccount;
        Text := tmpText;
        SelStart := Length(FAccount);
```

```
      end
      else
      if (SelStart = 4) or (SelStart = 7) or (SelStart = 10) or
(SelStart = 13) then
      begin
        if (not FExistParentAccount) then
        begin

Application.MessageBox(PChar(FNotExistParentAccountMessage),'',MB_ICO
NWARNING);
          GetAccount;
          Text := tmpText;
          SelStart := Length(FAccount)-1;
        end;
      end;
    end
      else inherited;
  end
  else
  begin
    inherited;
    Text := tmpText;
  end;
end;

procedure TAccountMaskEdit.CMEnter(var Message: TCMEnter);
begin
  inherited;
  SelStart := Length(GetAccount);
end;

procedure TAccountMaskEdit.WMLButtonDown(var Message:
TWMLButtonDown);
begin
  inherited;
  SelStart := Length(GetAccount);
end;


procedure Register;
begin
  RegisterComponents('Samples', [TAccountMaskEdit]);
end;

end.
```

1.2 Το εξάρτημα διαχείρισης πραγματικών αριθμών (TLimitSpinEdit)

```
unit LimitSpinEdit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Spin;

type
  TLimitSpinEdit = class(TSpinEdit)
  private
```

```
    FPrecision:Word;
    FMinCurrValue: Currency;
    FMaxCurrValue: Currency;
    procedure CMExit(var Message: TCMExit);   message CM_EXIT;
    function GetButtonVisible():boolean;
    procedure SetButtonVisible(btnVisible:boolean);
    function GetCurrencyValue():Currency;
    procedure SetCurrencyValue(CurValue:Currency);
  protected
    procedure KeyDown(var Key: Word; Shift: TShiftState); override;
    procedure KeyPress(var Key: Char); override;
  public
    constructor Create(AOwner: TComponent);override;
    function GetStringForSQL():string;
    destructor Destroy; override;
  published
    property ButtonVisible: Boolean read GetButtonVisible write
SetButtonVisible default True;
    property CurrencyValue: Currency read GetCurrencyValue write
SetCurrencyValue ;
    property Precision: Word read FPrecision write FPrecision ;
    property MaxCurrValue: Currency read FMaxCurrValue write
FMaxCurrValue;
    property MinCurrValue: Currency read FMinCurrValue write
FMinCurrValue;

  end;

procedure Register;

implementation

constructor TLimitSpinEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Precision := 0;
  CurrencyValue := 0;
  MinCurrValue := 0;
  MaxCurrValue := 0;
end;

procedure TLimitSpinEdit.KeyDown(var Key: Word; Shift: TShiftState);
begin
  if not ReadOnly then
    if Precision > 0 then
      if (Key = 38) then
      begin
        SetCurrencyValue(StrToCurr(Text)+Increment );
        Key := 35;
      end
      else if (Key = 40) then
      begin
        SetCurrencyValue(StrToCurr(Text)-Increment);
        Key := 35;
      end;

  inherited KeyDown(Key,Shift);
end;

procedure TLimitSpinEdit.KeyPress(var Key: Char);
begin
```

```
  if (Key = DecimalSeparator) then
  begin
    if (pos(DecimalSeparator,Text) <> 0) or (Fprecision = 0) then Key
:= chr(35);
  end
  else if (Key = '-') then
  begin
    if GetSelStart <> 0 then Key := chr(35);
  end
  else if ((key < '0') or (Key > '9')) and (Key > #32) then  Key :=
chr(35)
  else if ((key >= '0') and (Key <= '9')) then
    if (pos(DecimalSeparator,Text) <> 0) and
       (Fprecision <> 0) and
       (Length(Text) = pos(DecimalSeparator,Text) + Fprecision) and
       (SelStart >= pos(DecimalSeparator,Text)) then  Key := chr(35);

  inherited KeyPress(Key);
end;

procedure TLimitSpinEdit.CMExit(var Message: TCMExit);
var
  tmpText:string;
  flag:boolean;
begin
  flag := false;
  if (pos(decimalseparator,Text) <> 0) and (Self.Text <>
decimalseparator) then
  begin
    flag := true;
    tmpText := Text;
  end;

  if (Self.Text = '') or (Self.Text = '-') or (Self.Text =
decimalseparator) or (pos(decimalseparator,Text) <> 0)then
  begin
    Value := 0;
    if (pos(decimalseparator,Text) = 0) or (Self.Text =
decimalseparator) then CurrencyValue := 0;
  end;

  inherited;


  if flag then
    CurrencyValue  := StrToCurr(tmpText);
end;

destructor TLimitSpinEdit.Destroy;
begin
  inherited Destroy;
end;

function TLimitSpinEdit.GetButtonVisible():boolean;
begin
  Result := Button.Visible ;
end;

procedure TLimitSpinEdit.SetButtonVisible(btnVisible:boolean);
begin
  Button.Visible := btnVisible;
```

```
end;

function TLimitSpinEdit.GetCurrencyValue():Currency;
begin
  try
    Result := StrToCurr(Text);
  except
    Result := 0;
  end;
end;

procedure TLimitSpinEdit.SetCurrencyValue(CurValue:Currency);
begin
  if (FMinCurrValue <> FMaxCurrValue) then
    if (CurValue < FMinCurrValue) then CurValue := FMinCurrValue
    else if (CurValue > FMaxCurrValue) then CurValue :=
FMaxCurrValue;

  Text := CurrToStr(CurValue)
end;

function TLimitSpinEdit.GetStringForSQL():string;
var
  ReturnStr:string;
begin
  ReturnStr := Text;
  if decimalseparator = ',' then
    if pos(',',Text) <> 0 then
      ReturnStr[pos(',',Text)] := '.';
  Result := ReturnStr;
end;

procedure Register;
begin
  RegisterComponents('Samples', [TLimitSpinEdit]);
end;

end.
```

2. Γενικές διεπαφές του συστήματος

2.1 Η διεπαφή-πρότυπο (TfrmTemplate)

```
unit TemplateForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  Db, DBTables, ComCtrls, ExtCtrls, StdCtrls,Buttons,Variants;

type
  TfrmTemplate = class(TForm)
    myDatabase: TDatabase;
    myQuery: TQuery;
    CaptionQuery: TQuery;
    ErrorsQuery: TQuery;
    ComboQuery: TQuery;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public


  protected
    function GetData(var iQuery:TQuery;strSQL:widestring):bool;
    function ExecQuery(var iQuery:TQuery;strSQL:widestring):bool;
    procedure SwitchControls(Sender: TObject);
    function CheckIfExists(var
iQuery:TQuery;Table:string;Code:string;Language:string):boolean;
    function GetError(ErrorID:string):string;
    procedure FillComboBox(ListID:string;Combobox:TComboBox);
    function GetListValue(ListID:string;ListValueID:string):string;
    procedure FillFormCaption(Form:TForm);
  end;

var
  frmTemplate: TfrmTemplate;

implementation

uses MainForm;

{$R *.DFM}

//Η συνάρτηση πρόσβασης στη βάση για ανάκτηση δεδομένων
function TfrmTemplate.GetData(var
iQuery:TQuery;strSQL:widestring):bool;
begin
  try
    iQuery.Close;
    iQuery.SQL.Clear;
    iQuery.SQL.Add(strSQL);
    iQuery.Prepare;
    iQuery.Open;
    Getdata := true;
  except
    Getdata := false;
  end;
end;

//Η συνάρτηση πρόσβασης στη βάση δεδομένων για εκτέλεση εντολών SQL
χωρίς επιστροφή δεδομένων
function TfrmTemplate.ExecQuery(var
iQuery:TQuery;strSQL:widestring):bool;
begin
  try
    iQuery.Close;
    iQuery.SQL.Clear;
    iQuery.SQL.Add(strSQL);
    iQuery.Prepare;
    iQuery.ExecSQL;
    ExecQuery := true;
  except
    ExecQuery := false;
  end;
end;

//Απλά προχωράει στο επόμενο control της φόρμας
```

```
procedure TfrmTemplate.SwitchControls(Sender: TObject);
begin
  Self.FocusControl(FindNextControl((Sender as
TWinControl),true,true,false));
end;

//Αυτή η συνάρτηση ελέγχει αν υπάρχει μια εγγραφή στον πίνακα Table
εγγραφή αυτή προσδιορίζεται
//από code που αντιστοιχεί στο ID του πίνακα Table
function TfrmTemplate.CheckIfExists(var
iQuery:TQuery;Table:string;Code:string;Language:string):boolean;
begin
  Result := false;

  if GetData(iQuery,'SELECT ID,Description FROM '+ Table +' WHERE
ID=''' + Code + ''' AND LanguageID=''' + Language + '''') then
  begin
    if myQuery.RecordCount > 0 then Result := true;
  end
  else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
end;

//Ελευθερώνει τη μνήμη στο κλείσιμο
procedure TfrmTemplate.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
end;

//Η συνάρτηση αυτή γεμιζεί ένα combobox με τισ ανάλογες τιμές από τη
βάση.Κάθε λίστα τιμών
//έχει ένα ListID και από αυτό ξεχωρίζουμε ποιές τιμές του πίνακα
ListValues αντιστοιχούν στο combobox
procedure
TfrmTemplate.FillComboBox(ListID:string;Combobox:TCombobox);
var i:integer;
begin
  if GetData(ComboQuery,'SELECT * FROM LISTVALUES WHERE LISTID = '''
+ ListID
                      + ''' AND LANGUAGEID = ''' +
frmMain.UserQuery.FieldByName('LanguageID').AsString
                      + ''' ORDER BY ListValueID') then
  begin
    ComboQuery.First ;
    Combobox.Items.Clear ;
    for i:= 0 to ComboQuery.RecordCount - 1 do
    begin

Combobox.Items.Add(ComboQuery.FieldbyName('LISTVALUE').AsString);
      ComboQuery.Next;
    end;
  end
  else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
end;

//Επιστρέφει το ListValue όταν του δοθεί το ListValueID
(χρησιμοποιείται για τη γλώσσα.Εσωτερικά δουλεύουμε
//με ListValueID για να μην εξαρτώμαστε από τη γλώσσα.Όταν
χρειαζόμαστε κάποιο ListValue το ζητάμε με τη
```

```
//συνάρτηση αυτή)
function
TfrmTemplate.GetListValue(ListID:string;ListValueID:string):string;
begin
  if GetData(ComboQuery,'SELECT * FROM LISTVALUES WHERE LISTID = '''
+ ListID
                      + ''' AND LISTVALUEID ='''' + ListValueID
                      + ''' AND LANGUAGEID = ''' +
frmMain.UserQuery.FieldByName('LanguageID').AsString
                      + '''') then
    Result := ComboQuery.FieldByName('LISTVALUE').AsString
  else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
end;

//Αυτή η συνάρτηση γεμίζει τα caption της φόρμας (τα labels,το τίτλο
της φόρμας,τα caption των RadioButtons,
//τα caption των ListView καθώς και τα caption των Buttons).Τις
πληροφορίες για τη γλώσσα που έχει επιλέξει ο χρήστης
//τισ παίρνει από το UserQuery.Το query αυτό γεμίζει στο login και
περιέχει την εγγραφή του χρήστη στο σύστημα
//(επομένως και τη γλώσσα)
procedure TfrmTemplate.FillFormCaption(Form:TForm);
var i,j:integer;
begin
  if GetData(CaptionQuery,'SELECT * FROM CAPTIONS WHERE FORMNAME ='''
+ Form.Name + ''''') then
  begin
    //Ο τίτλος της φόρμας
    if
CaptionQuery.Locate('ControlName;LanguageID',VarArrayOf([Form.Name,fr
mMain.UserQuery.FieldByName('LanguageID').AsString]),[]) then
Form.Caption := CaptionQuery.FieldByName('CAPTIONTEXT').AsString;

    for i:=0 to form.ControlCount - 1 do
    begin
      //Τα labels
      if (Form.Controls[i] is TLabel) then
      begin
        if
CaptionQuery.Locate('ControlName;LanguageID',VarArrayOf([(Form.Contro
ls[i] as
TLabel).Name,frmMain.UserQuery.FieldByName('LanguageID').AsString]),[
]) then
          (Form.Controls[i] as TLabel).Caption :=
CaptionQuery.FieldByName('CAPTIONTEXT').AsString;
      end
      //Τα radiobuttons
      else if (Form.Controls[i] is TRadioButton) then
      begin
        if
CaptionQuery.Locate('ControlName;LanguageID',VarArrayOf([(Form.Contro
ls[i] as
TRadioButton).Name,frmMain.UserQuery.FieldByName('LanguageID').AsStri
ng]),[]) then
          (Form.Controls[i] as TRadioButton).Caption :=
CaptionQuery.FieldByName('CAPTIONTEXT').AsString;
      end
      //Τα ListView
      else if (Form.Controls[i] is TListView) then
      begin
```

```
        for j:=0 to (Form.Controls[i] as TListView).Columns.Count - 1
do
        begin
          if
CaptionQuery.Locate('ControlName;LanguageID',VarArrayOf([(Form.Contro
ls[i] as TListView).Name + '-'
+IntToStr(j),frmMain.UserQuery.FieldByName('LanguageID').AsString]),[
]) then
            (Form.Controls[i] as TListView).Columns[j].Caption :=
CaptionQuery.FieldByName('CAPTIONTEXT').AsString;
        end;
      end
      //Τα Comboboxes
      else if (Form.Controls[i] is TComboBox) then
FillComboBox((Form.Controls[i] as TComboBox).Name,(Form.Controls[i]
as TComboBox))
      //Τα buttons
      else if (Form.Controls[i] is TBitBtn) then
      begin
        if
CaptionQuery.Locate('ControlName;LanguageID',VarArrayOf([(Form.Contro
ls[i] as
TBitBtn).Name,frmMain.UserQuery.FieldByName('LanguageID').AsString]),
[]) then
          (Form.Controls[i] as TBitBtn).Caption :=
CaptionQuery.FieldByName('CAPTIONTEXT').AsString;
      end
      else if (Form.Controls[i] is TToolBar) then
      begin
        for j:=0 to (Form.Controls[i] as TToolBar).ButtonCount -1 do
          if
CaptionQuery.Locate('ControlName;LanguageID',VarArrayOf([(Form.Contro
ls[i] as
TToolBar).Buttons[j].Name,frmMain.UserQuery.FieldByName('LanguageID')
.AsString]),[]) then
            (Form.Controls[i] as TToolBar).Buttons[j].Caption :=
CaptionQuery.FieldByName('CAPTIONTEXT').AsString;
      end;
    end;
  end
  else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
end;

//Αυτή η συνάρτηση φέρνει το text του λάθους ανάλογα με τη γλώσσα
function TfrmTemplate.GetError(ErrorID:string):string;
begin
  Result := '';
  if GetData(ErrorsQuery,'SELECT * FROM ERRORS WHERE ERRORID =''' +
ErrorID + ''' AND LANGUAGEID=''' +
frmMain.UserQuery.FieldByName('LanguageID').AsString + '''') then
  begin
    if ErrorsQuery.RecordCount > 0 then Result :=
ErrorsQuery.FieldByName('ErrorText').AsString ;
  end
end;

//Στο event onShow ανακτά από τη βάση τις στατικές περιγραφές που
βρίσκονται πάνω στη φόρμα
procedure TfrmTemplate.FormShow(Sender: TObject);
begin
```

```
  FillFormCaption(Self);
end;

end.
```

2.2 Η κεντρική διεπαφή του συστήματος (TfrmMain)

```
unit MainForm;

interface

uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
     TemplateForm, StdCtrls, ExtCtrls, Db, DBTables,
ComCtrls,Accounts,AccountRecords,
     LPForm,Menus,formAddViews,formLogin,Variants;

type
  TfrmMain = class(TForm)
    myStatusBar: TStatusBar;
    myTimer: TTimer;
    MainMenu: TMainMenu;
    Logistiki: TMenuItem;
    SyntLogar: TMenuItem;
    KatKin: TMenuItem;
    Proboles: TMenuItem;
    LogistikoSxedio: TMenuItem;
    Syntirisi: TMenuItem;
    PinakesBoith: TMenuItem;
    UserQuery: TQuery;
    myDatabase: TDatabase;
    Arxeio: TMenuItem;
    Syndesi: TMenuItem;
    MenuCaptionQuery: TQuery;
    Language_DataSource: TDataSource;
    Languages: TTable;
    procedure myTimerTimer(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure SyntLogarClick(Sender: TObject);
    procedure KatKinClick(Sender: TObject);
    procedure LogistikoSxedioClick(Sender: TObject);
    procedure PinakesBoithClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SyndesiClick(Sender: TObject);
    procedure FormResize(Sender: TObject);

  private
    { Private declarations }
    procedure Login;
  public
    { Public declarations }
  end;

var
  frmMain: TfrmMain;

implementation

{$R *.DFM}
```

```
procedure TfrmMain.myTimerTimer(Sender: TObject);
begin
  myStatusBar.Panels[2].Text := FormatDateTime('hh:mm',Now);
  myStatusBar.Panels[0].Text := FormatDateTime('dd/mm/yyyy',Date);
end;

procedure TfrmMain.FormShow(Sender: TObject);
var i:integer;
begin
  myStatusBar.Panels[0].Width := 80;
  myStatusBar.Panels[2].Width := 70;
  myStatusBar.Panels[1].Width := myStatusBar.Width -
myStatusBar.Panels[0].Width - myStatusBar.Panels[2].Width ;

  if UserQuery.RecordCount > 0 then
    try
      //Στο σημείο αυτό φέρνονται από τη βάση τα captions του μενού
      with MenuCaptionQuery do
      begin
        Close;
        Sql.Text := 'SELECT * FROM CAPTIONS WHERE FORMNAME =
''frmMain'' ';
        Open;

        if
Locate('ControlName;LanguageID',VarArrayOf(['frmMain',UserQuery.field
ByName('LanguageID').AsString]),[]) then Self.Caption :=
FieldByName('CAPTIONTEXT').AsString;

        for i:= 0 to MainMenu.Items.Count - 1 do
          if
Locate('ControlName;LanguageID',VarArrayOf([MainMenu.Items[i].Name,Us
erQuery.fieldByName('LanguageID').AsString]),[]) then
            MainMenu.Items[i].Caption :=
FieldByName('CAPTIONTEXT').AsString;
      end
    except
      for i:= 0 to MainMenu.Items.Count - 1 do
MainMenu.Items[i].Caption := '??';
    end;
end;

procedure TfrmMain.SyntLogarClick(Sender: TObject);
begin
  frmAccounts := TfrmAccounts.Create(Self);
  frmAccounts.Show ;
end;

procedure TfrmMain.KatKinClick(Sender: TObject);
begin
  frmAccountRecords := TfrmAccountRecords.Create(Self);
  frmAccountRecords.Show ;
end;

procedure TfrmMain.LogistikoSxedioClick(Sender: TObject);
begin
  frmLP := TfrmLP.Create(Self);
  frmLP.Show ;
end;

procedure TfrmMain.PinakesBoithClick(Sender: TObject);
```

```
begin
  frmAddViews := TfrmAddViews.Create(Self);
  frmAddViews.Show ;
end;

procedure TfrmMain.FormCreate(Sender: TObject);
begin
  Login;
end;

//Η συνάρτηση αυτή εκτελεί τη διαδικασία του login.Εκτελείται κατά τη
δημιουργία της mainform
//και όταν ο χρήστης επιλέξει από το μενού "σύνδεση"
procedure TfrmMain.Login;
var i :integer;
begin

  //Εμφανίζει τη φόρμα του login
  frmLogin := TfrmLogin.Create(self);

  frmLogin.cmbLanguages.ListSource := Language_DataSource ;
  frmLogin.cmbLanguages.ListField := 'Description';
  frmLogin.cmbLanguages.KeyField := 'ID';

  frmLogin.ShowModal ;

  //Αρχικά απενεργοποιεί το μενού εκτός από τη "σύνδεση"
  for i:=0 to MainMenu.Items.Count - 1 do
    if (MainMenu.Items[i].Name <> 'Arxeio') and
(MainMenu.Items[i].Name <> 'Syndesi') then MainMenu.Items[i].Enabled
:= false;

  try
    //Εκτελεί το ερώτημα για να βρει το χρήστη
    with UserQuery do
    begin
      Close;

      SQL.Text := 'SELECT * FROM USERS WHERE USERNAME =''' +
frmLogin.edUsername.Text
                                  + ''' AND PASSWORD =''' +
frmLogin.pedPassword.Password
                                  + ''' AND LANGUAGEID =''' +
Languages.FieldByName('ID').AsString + '''';
      Prepare;
      Open;

      //Αν τον βρει ξαναενεργοποιεί το μενού
      if Recordcount > 0 then
      begin
        myStatusBar.Panels[1].Text :=
FieldByName('FIRSTNAME').AsString + '   ' +
FieldByName('LASTNAME').AsString;
        for i:=0 to MainMenu.Items.Count - 1 do
MainMenu.Items[i].Enabled := true;
      end
      else
      begin
        Application.MessageBox('Incorrect username or password!Login
failed','',MB_ICONERROR);
        myStatusBar.Panels[1].Text := 'Bad login';
```

```
      end;
    end;
  except
    begin
      Application.MessageBox('Error while connecting to the
database!','',MB_ICONERROR);
      myStatusBar.Panels[1].Text := 'Error!';
    end;
  end;

  FreeAndNil(frmLogin);
end;

procedure TfrmMain.SyndesiClick(Sender: TObject);
begin
  Login;
end;

procedure TfrmMain.FormResize(Sender: TObject);
begin
  myStatusBar.Panels[0].Width := 80;
  myStatusBar.Panels[2].Width := 70;
  myStatusBar.Panels[1].Width := myStatusBar.Width -
myStatusBar.Panels[0].Width - myStatusBar.Panels[2].Width ;
end;

end.
```

2.3 Η διεπαφή εισόδου στο σύστημα (TfrmLogin)

```
unit formLogin;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Buttons, PasswordEdit, DBCtrls;

type
  TfrmLogin = class(TForm)
    edUsername: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    btnLogin: TBitBtn;
    pedPassword: TPasswordEdit;
    cmbLanguages: TDBLookupComboBox;
    procedure btnLoginClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;


var
  frmLogin: TfrmLogin;

implementation
```

```
{$R *.DFM}

procedure TfrmLogin.btnLoginClick(Sender: TObject);
begin
  Close;
end;

end.
```

3. Το υποσύστημα των προβολών

3.1 Η διεπαφή προβολής των δεδομένων (TfrmViews)

3.1.1 Ο πηγαίος κώδικας της διεπαφής

```
unit formViews;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  TemplateForm, ComCtrls, ExtCtrls, Db,
DBTables,formSelView,formSearchOnView,
  StdCtrls;

type
  TfrmViews = class(TfrmTemplate)
    LV: TListView;
    FieldsQuery: TQuery;
    dataQuery: TQuery;
    TablesQuery: TQuery;
    sumLV: TListView;
    KeyFieldsQuery: TQuery;
    procedure LVKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure LVDblClick(Sender: TObject);
    procedure LVKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
    SelIndex:integer;
    LastSearchString:string;
    LastSearchType:TSearchType;
    LastColumnIndex:integer;
    CurrentViewID:string;

    procedure FillLV(Crit:string);  //Γεμίζει το ListView
    function SelViewID(FormName:string;CtrlName:string):string;
//Βρίσκει το ViewID από τη βάση
    function GetFields(ViewID:string):string; //Φέρνει τα πεδία του
επιλεγμένου ερωτήματος
    function GetTables(ViewID:string):string; //Φέρνει τους πίνακες
του επιλεγμένου ερωτήματος
    procedure MoveToTable();  //Η συνάρτηση αυτή εκτελεί τη μετάβαση
στο βασικό πίνακα όταν ζητηθεί

  public
    { Public declarations }
```

```
    //Η συνάρτηση που καλείται από τις φόρμες για εμφάνιση προβολών
    procedure ShowView(FormName:string;Criteria:string);

    //Η συνάρτιση που καλείται από τα controls για εμφάνιση πίνακα
βοήθειας
    function
ShowViewForFields(FormName:string;CtrlName:string;Criteria:string):st
ring;
  end;

var
  frmViews: TfrmViews;

implementation

uses Accounts;

{$R *.DFM}

procedure TfrmViews.ShowView(FormName:string;Criteria:string);
var
  ViewID:string;
  SQLQuery : widestring;
  StrCrit : string;
begin
  FormStyle := fsMDIChild;

  //Αρχικά επιλέγεται το ViewID και αποθηκεύεται στη μεταβλητή
CurrentViewID
  ViewID := SelViewID(FormName,FormName); //Η συνάρτηση SelViewID
εμφανίζει τη φόρμα επιλογής
                                          //προβολής και
επιστρέφει το VIEWID της προβολής που
                                          //επέλεξε ο χρήστης
  CurrentViewID := ViewID;

  if ViewID <> '' then
  begin
    //Εδώ φέρνουμε στο αντικείμενο myQuery την εγγραφή του πίνακα
VIEWS για το επιλεγμένο VIEWID
    if GetData(myQuery,'SELECT * FROM VIEWS WHERE FORMNAME=''' +
FormName + ''' AND VIEWID=''' + ViewID + '''') then
    begin
      //Τίτλος της φόρμας γίνεται το όνομα της προβολής
      Caption := myQuery.fieldByName('VIEWDESC').AsString ;

      //Από εδώ ξεκινάει η κατασκευή του ερωτήματος της προβολής
      SQLQuery := 'SELECT ' + GetFields(ViewID) + ' FROM ' +
GetTables(ViewID) ; //Αρχικά κατασκευάζεται το

            //το πρώτο κομμάτι του ερωτήματος

            //με τη βοήθεια των GetFields και

            //GetTables που επιστρέφουν τα πεδία

            //και τους πίνακες του ερωτήματος

            //αντίστοιχα
```

```
      StrCrit := '';  //Αρχικοποιείται η StrCrit όπου θα αποθηκευτεί
το string των κριτηρίων
      if myQuery.fieldByName('CRITERIA').AsString <> '' then
//Ελέγχεται αν υπάρχουν στο πεδίο CRITERIA του

//πίνακα VIEWS κριτήρια για αυτήν τη προβολή
      begin
        StrCrit := ' WHERE ' +
myQuery.fieldByName('CRITERIA').AsString;  //Αν ναι τότε προστίθενται
στο string
        if Criteria <> '' then StrCrit := StrCrit + ' AND ' +
Criteria;   //και στη συνέχεια προστίθενται και τα κριτήρια

     //που έχει περάσει σαν παράμετρο η φόρμα που

     //ζητάει τη προβολή
      end
      else if Criteria <> '' then StrCrit := ' WHERE ' + Criteria;
//Αλλιώς προστίθενται μόνο τα κριτήρια της φόρμας

      SQLQuery := SQLQuery + StrCrit; //Το string των κριτηρίων
προστίθεται στο υπόλοιπο ερώτημα

      //Επίσης προστίθενται τα πεδία ομαδοποίησης
      if myQuery.fieldByName('GROUPBY').AsString <> '' then SQLQuery
:= SQLQuery + ' GROUP BY ' + myQuery.fieldByName('GROUPBY').AsString;

      //και ταξινόμησης που παίρνονται από τα αντίστοιχα πεδία του
πίνακα VIEWS
      if myQuery.fieldByName('ORDERBY').AsString <> '' then SQLQuery
:= SQLQuery + ' ORDER BY ' + myQuery.fieldByName('ORDERBY').AsString;

      //Τέλος εκτελείται το ερώτημα , γεμίζει το ListView  με τα
δεδομένα
      if GetData(dataQuery,SQLQuery) then FillLV(StrCrit);

      //και εμφανίζεται η φόρμα
      Show;
    end;
  end;
end;

procedure TfrmViews.LVKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if Key = 13 then LVDblClick(Sender)
  else if Key = 27 then Close;
end;

procedure TfrmViews.LVDblClick(Sender: TObject);
begin
  inherited;
  if LV.Selected <> nil then
  begin
    SelIndex := LV.Selected.Index ;
    Close;
  end;
end;
```

```
//Η συνάρτηση αυτή είναι κατά το μεγαλύτερο μέρος της ίδια με την
ShowView και για το λόγο αυτό εξηγούνται
//μόνο οι διαφορές τους
function
TfrmViews.ShowViewForFields(FormName:string;CtrlName:string;Criteria:
string):string;
var
  ViewID,ResultField,SQLQuery,StrCrit : string;
begin
  Result := '';
  ViewID := SelViewID(FormName,CtrlName);
  CurrentViewID := ViewID;

  if ViewID <> '' then
  begin
    if GetData(myQuery,'SELECT * FROM VIEWS WHERE FORMNAME=''' +
FormName + ''' AND FORMNAME=''' + FormName + ''' AND VIEWID=''' +
ViewID + '''') then
    begin
      //Τίτλος της φόρμας γίνεται το όνομα της προβολής
      Caption := myQuery.fieldByName('VIEWDESC').AsString ;

      //Βασική διαφορά είναι η ύπαρξη του RESULTFIELD που είναι το
πεδίο της επιλεγμένης εγγραφής που επιστρέφεται
      //όταν ο χρήστης επιλέξει μια εγγραφή
      ResultField := myQuery.fieldByName('RESULTFIELD').AsString;
//Εδώ αποθηκεύεται το όνομα του RESULTFIELD που

//είναι αποθηκευμένο στο αντίστοιχο πεδίο του VIEWS

//στη μεταβλητή ResultField

      //Επειδή το ResultField μπορεί να είναι μέσα σε " αφαιρούνται
τα " από το string
      if pos('"',ResultField) <> 0 then
      begin
        delete(ResultField,1,1);
        delete(ResultField,Length(ResultField),1);
      end;

      SQLQuery := 'SELECT ' + GetFields(ViewID) +
                  ' FROM ' + GetTables(ViewID) ;

      StrCrit := '';
      if myQuery.fieldByName('CRITERIA').AsString <> '' then
      begin
        StrCrit :=  ' WHERE ' +
myQuery.fieldByName('CRITERIA').AsString;

        //Μια ακόμη διαφορά είναι ότι στη συνάρτηση αυτή το control
που τη καλεί περνάει στη παράμετρο
        //Criteria μόνο το string που έχει γράψει ο χρήστης και όχι
ολόκληρη την έκφραση των κριτηρίων όπως
        //επιβάλεται από την SQL.Η έκφραση δημιουργείται εδώ με τη
βοήθεια του πεδίου FORMCRITFIELD του πίνακα
        //VIEWS που περιέχει το πεδίο στο οποίο αναφέρεται το string
που έχει γράψει ο χρήστης
        if Criteria <> '' then StrCrit := StrCrit + ' AND '  +
myQuery.fieldByName('FORMCRITFIELD').AsString + ' LIKE''' + Criteria
+ '%''';
      end
```

```
      else if Criteria <> '' then StrCrit := ' WHERE ' +
myQuery.fieldByName('FORMCRITFIELD').AsString + ' LIKE''' + Criteria
+ '%''';
      SQLQuery := SQLQuery + StrCrit ;

      if myQuery.fieldByName('GROUPBY').AsString <> '' then SQLQuery
:= SQLQuery + ' GROUP BY ' + myQuery.fieldByName('GROUPBY').AsString;

      if myQuery.fieldByName('ORDERBY').AsString <> '' then SQLQuery
:= SQLQuery + ' ORDER BY ' + myQuery.fieldByName('ORDERBY').AsString;

      if GetData(dataQuery,SQLQuery) then
        FillLV(StrCrit);

      //Αρχικοποιείται η μεταβλητή SelIndex όπου θα αποθηκευτεί το
index της εγγραφής που θα επιλέξει ο χρήστης
      SelIndex := -1;

      Result := '';

      ShowModal;

      //Όταν κλείσει η φόρμα (επιστροφή από το ShowModal) το SelIndex
περιέχει το Index της επιλεγμένης εγγραφής
      if SelIndex > -1 then
      begin
        dataQuery.First ;
        dataQuery.MoveBy(SelIndex); //Ίδιο με την εγγραφή είναι και
το Index του συγκεκριμένου record
                                    //μέσα στο dataQuery και έτσι
εντοπίζουμε την εγγραφή που επέλεξε ο χρήστης
        Result := dataQuery.FieldByName(ResultField).AsString ; //και
επιστρέφουμε το πεδίο ResultField της

//εγγραφής αυτής
      end;
    end;
  end;
end;

//Η συνάρτηση αυτή γεμίζει το ListView από το dataQuery που ήδη
περιέχει τα δεδομένα
procedure TfrmViews.FillLV(Crit:string);
var
  i,j:integer;
  LVColumnsWidth:integer;
  StrFldName:string;
begin
  LV.Columns.Add ;
  LV.Columns[0].Width := 0;
  LVColumnsWidth := 0;

  sumLV.Columns.Add ;
  sumLV.Columns[0].Width := 0;
  sumLV.Items.Add ;
  sumLV.Items[0].Caption := '';

  //Φτιάχνουμε για κάθε πεδίο του ερωτήματος και μια στήλη στο LV
αλλά και στο sumLV
  for j:=1 to dataQuery.FieldCount do
  begin
```

```
    LV.Columns.Add ;
    sumLV.Columns.Add ;
    sumLV.Items[0].SubItems.Add('');
  end;

  //Διαβάζουμε ένα ένα τα πεδία και φτιάχνουμε τια στήλες (μέγεθος ,
επικεφαλίδα)
  FieldsQuery.first;
  for j:=1 to FieldsQuery.RecordCount do
  begin

    //Παίρνουμε το όνομα και αφαιρούμε τα " αν υπάρχουν (τα "
χρειάζονται για την SQL)
    StrFldName := FieldsQuery.FieldByName('FIELDNAME').AsString;
    if pos('"',StrFldName) <> 0 then
    begin
      delete(StrFldName,1,1);
      delete(StrFldName,Length(StrFldName),1);
    end;

    //Βάζουμε το όνομα στη στήλη που δείχνει που η σειρά του πεδίου
(το πεδίο "TURN")
    LV.Columns[FieldsQuery.FieldByName('TURN').AsInteger + 1].Caption
:= StrFldName ;
    LVColumnsWidth := LVColumnsWidth + (Length(StrFldName)*10);

    FieldsQuery.Next ;
  end;

  //Αρχικοποιούμε το μέγεθος των στηλών
  if (Width - 27) >  LVColumnsWidth then LVColumnsWidth := ((Width -
27) - LVColumnsWidth) div dataQuery.FieldCount;

  //Ρυθμίζουμε το μέθος κάθε στήλης ανάλογα με το μέγεθος τις
επικεφαλίδας
  for j:=1 to dataQuery.FieldCount do
  begin
    LV.Columns[j].Width := (Length(dataQuery.Fields[j-
1].FieldName)*10) + LVColumnsWidth;
    sumLV.Columns[j].Width := LV.Columns[j].Width;
  end;

  //Βάζουμε τα δεδομένα στο LV
  for i:=0 to dataQuery.RecordCount - 1 do
  begin
    LV.Items.Add;
    for j:=1 to dataQuery.FieldCount do

LV.Items[i].SubItems.Add(dataQuery.FieldByName(LV.Columns[j].Caption)
.AsString);
    dataQuery.Next ;
  end;

  //Εδώ γίνονται οι έλεγχοι για το τύπο του πεδίου (π.χ. αθροίσματα)
  FieldsQuery.First ;
  for j:=1 to dataQuery.FieldCount do
  begin
    if FieldsQuery.FieldByName('TYPE').AsString <> 'N' then
    begin
```

```
      //Αν ο τύπος είναι "S" τότε άθροισε τη στήλη και βάλε το
αποτέλεσμα στο sumLV στην TURN στήλη
      if FieldsQuery.FieldByName('TYPE').AsString = 'S' then
      begin
        if GetData(myQuery,'SELECT SUM(' +
FieldsQuery.FieldByName('DBFIELD').AsString
                           + ') as ' +
FieldsQuery.FieldByName('FIELDNAME').AsString
                           + ' FROM ' +
FieldsQuery.FieldByName('DBTABLE').AsString
                           + Crit) then
          begin

sumLV.items[0].SubItems[FieldsQuery.FieldByName('TURN').AsInteger] :=
myQuery.FieldByName(FieldsQuery.FieldByName('FIELDNAME').AsString).As
String;
          end;
      end
      else

      //Αν ο τύπος της στήλης είναι "L" τότε βάλε την τιμή της
τελευταίας εγγραφής στην αντίστοιχη στήλη του sumLV
      if FieldsQuery.FieldByName('TYPE').AsString = 'L' then
      begin

sumLV.Items[0].SubItems[FieldsQuery.FieldByName('TURN').AsInteger] :=
LV.Items[LV.Items.count-
1].SubItems[FieldsQuery.FieldByName('TURN').AsInteger];
      end;

    end;

    FieldsQuery.Next ;
  end;
end;

//Είναι μια συνάρτηση που βρίσκει τις προβολές που υπάρχουν για ένα
συγκεκριμένο control και εμφανίζει τη
//φόρμα απιλογής προβολής
function TfrmViews.SelViewID(FormName:string;CtrlName:string):string;
var
  i:integer;
begin
  Result := '';
  //Φέρνει όλες τις προβολές για το συγκεκριμένο control
  if GetData(myQuery,'SELECT * FROM VIEWS WHERE FORMNAME=''' +
FormName + ''' AND CTRLNAME=''' + CtrlName + '''') then
  begin
    if myQuery.RecordCount > 0 then
    begin
      if myQuery.RecordCount > 1 then
      begin
        //Εμφανίζεται η φόρμα επιλογής
        frmSelView := TfrmSelView.Create(Self) ;

        for i:=0 to myQuery.RecordCount - 1 do
        begin
          frmSelView.LV.Items.Add ;
          frmSelView.LV.Items[i].Caption :=
myQuery.fieldByName('VIEWID').AsString ;
```

```
frmSelView.LV.Items[i].SubItems.Add(myQuery.fieldByName('VIEWDESC').A
sString);
          myQuery.Next ;
        end;

        Result := frmSelView.ShowSelView ;
        FreeAndNil(frmSelView);
      end
      //Επιστρέφει το ViewID που επιλέχθηκε
      else Result := myQuery.fieldByName('VIEWID').AsString;
    end;
  end;
end;

//Φέρνει τα πεδία μιας προβολής κσι τα βάζει στο FieldsQuery
function TfrmViews.GetFields(ViewID:string):string;
var
  i:integer;
begin
  Result := '';
  if GetData(FieldsQuery,'SELECT * FROM VIEWFIELDS WHERE VIEWID=''' +
ViewID + '''') then
  begin
    if FieldsQuery.RecordCount > 0 then
    begin
      Result := Result + FieldsQuery.fieldByName('DBFIELD').AsString
+ ' as "' + FieldsQuery.fieldByName('FIELDNAME').AsString + '"';
      FieldsQuery.Next ;
      for i:= 1 to  FieldsQuery.RecordCount - 1 do
      begin
        Result := Result + ',' +
FieldsQuery.fieldByName('DBFIELD').AsString + ' as "' +
FieldsQuery.fieldByName('FIELDNAME').AsString + '"';
        FieldsQuery.Next ;
      end;
    end;
  end;
end;

//Φέρνει τουε πίνακες που εμπλέκονται στο ερώτημα της προβολής και
φτιάχνει το string των πινάκων
//όπως ακριβώς χρειάζεται στο ερώτημα (π.χ. επιστρέφει το string
"GLM,GLK,GLP")
function TfrmViews.GetTables(ViewID:string):string;
var
  i:integer;
  TableStr,Tables:string;
begin
  Result := '';
  if GetData(TablesQuery,'SELECT DISTINCT(DBTABLE) FROM VIEWFIELDS
WHERE VIEWID=''' + ViewID + '''') then
  begin
    if TablesQuery.RecordCount > 0 then
    begin
      Result := Result + TablesQuery.fieldByName('DBTABLE').AsString;
      TablesQuery.Next ;
      for i:= 1 to  TablesQuery.RecordCount - 1 do
      begin
        //μπορεί κάποιο calculated πεδίο να χρειάζεται 2 πίνακες
οπότε θα πρέπει να τυος ξεχωρίσουμε και
```

```
        //να ελέγξουμε αν υπάρχει ήδη κάποιος από αυτούς στο string
        Tables := TablesQuery.fieldByName('DBTABLE').AsString;
        while pos(',',Tables) > 0 do
        begin
          TableStr := copy(Tables,1,pos(',',Tables)-1);
          delete(Tables,1,pos(',',Tables));
          if pos(TableStr,Result) = 0 then Result := Result + ',' +
TableStr;
        end;
        if pos(Tables,Result) = 0 then Result := Result + ',' +
Tables;
        TablesQuery.Next ;
      end;
    end;
  end;
end;

//Το event onKeyPress
procedure TfrmViews.LVKeyPress(Sender: TObject; var Key: Char);
begin
  inherited;
  //Αν πατήσει το "F" εμφανίζεται η φόρμα εύρεσης
  if (Key = 'f') or (Key = 'F') then
  begin
    frmSearchOnView := TfrmSearchOnView.Create(Self);
    LastColumnIndex :=
frmSearchOnView.Find(LV,LastSearchType,LastSearchString);
    FreeAndNil(frmSearchOnView);
  end
  else

  //Αν πατήσει το "N" κάνει find next
  if (Key = 'n') or (Key = 'N') then
  begin
    frmSearchOnView := TfrmSearchOnView.Create(Self);

frmSearchOnView.FindNext(LV,LastSearchType,LastColumnIndex,LastSearch
String);
    FreeAndNil(frmSearchOnView);
  end
  else
  //Αν πατήσει "P" μεταφέρεται στην αντίστοιχη φόρμα
  if (Key = 'p') or (Key = 'P') then MoveToTable;
end;

//Εμφανίζει την αντίστοιχη φόρμα όταν ο χρήστης πατήσει το "P"
procedure TfrmViews.MoveToTable();
var
  KeysString,KeysQueryString,QueryString:string;
  i:integer;
begin
  if LV.Selected <> nil then
  begin
    //Φέρνει την εγγραφή της προβολής
    if GetData(myQuery,'SELECT * FROM VIEWS WHERE VIEWID=''' +
CurrentViewID + '''') then
    begin
      //Ελέγχει αν είναι συμπληρωμένα τα πεδία "KEYS" και "MAINTABLE"
που είναι απαραίτητα για την εμφάνιση
      if (myQuery.FieldByName('KEYS').AsString <> '') and
(myQuery.FieldByName('MAINTABLE').AsString <> '') then
```

```
      begin
        //Φτιάχνει το query για να φέρει τα πεδία κλειδιά στο
KeyFieldsQuery
        KeysQueryString := 'SELECT * FROM VIEWFIELDS WHERE VIEWID='''
+ CurrentViewID + ''' AND (';

        KeysString := myQuery.FieldByName('KEYS').AsString;
        if pos(',',KeysString) > 0 then
        begin
          KeysQueryString := KeysQueryString + 'FIELDNAME=''' +
copy(KeysString,1,pos(',',KeysString));
          delete(KeysString,1,pos(',',KeysString)+1);
          while pos(',',KeysString) > 0 do
          begin
            KeysQueryString := KeysQueryString + ' OR ' +
'FIELDNAME=''' + copy(KeysString,1,pos(',',KeysString));
            delete(KeysString,1,pos(',',KeysString)+1);
          end;
          KeysQueryString := KeysQueryString + ' OR ' +
'FIELDNAME=''' +  KeysString + ''')';
        end
        else KeysQueryString := KeysQueryString + 'FIELDNAME=''' +
KeysString + ''')';

        //Φτιάχνει το query για να φέρει την εγγραφή του MainTable
που έχει επιλέξει ο χρήστης
        if GetData(KeyFieldsQuery,KeysQueryString) then
        begin
          QueryString := 'SELECT * FROM ' +
myQuery.FieldByName('MAINTABLE').AsString + ' WHERE';

          for i:=0 to KeyFieldsQuery.RecordCount - 1 do
          begin
            if i>1 then QueryString := QueryString + ' AND' ;
            QueryString := QueryString + ' ' +
KeyFieldsQuery.FieldByName('DBFIELD').AsString + '=''' +
LV.Selected.SubItems[KeyFieldsQuery.FieldByName('TURN').AsInteger] +
'''' ;
            KeyFieldsQuery.Next ;
          end;

          //Εδώ αρχίζει η άτυπη case

          if myQuery.FieldByName('MAINTABLE').AsString = 'GLM' then
          begin
            //Αν ο mainTable είναι ο GLM τότε εκτέλεσε το ερώτημα που
φτιάχτηκε (QueryString) στο CurrRecQuery
            //της φόρμας 110 (είναι το query όπου αποθηκεύεται η
παρούσα εγγραφή) και γέμισε τη φόρμα από το query
            frmAccounts := TfrmAccounts.Create(Self);
            if GetData(frmAccounts.CurrRecQuery,QueryString) then
frmAccounts.FillFormFromQuery ;
            frmAccounts.Show;
          end;
        end;
      end;
    end;
  end;
end;

end.
```

3.1.2 Η διαδικασία εμφάνισης των προβολών

Η δημιουργία των προβολών γίνεται από τη φόρμα frmViews στο unit formViews. Για τη κλήση αυτη΄ς της φόρμας έχουν δημιουργηθεί δύο public συναρτήσεις :

- η procedure *ShowView(FormName:string;Criteria:string);*
- και η function *ShowViewForFields (FormName:string; CtrlName:string; Criteria:string) :string;*

Η διαφορά των δύο συναρτήσεων είναι ότι η πρώτη χρησιμοποιείται για προβολές από φόρμα όπως για παράδειγμα η προβολή λογιστικού σχεδίου ενώ η δεύτερη για προβολές με τη μορφή πίνακα βοήθειας ώστε ο χρήστης να συμπληρώσει κάποιο πεδίο.Και οι δύο εμφανίζουν την ίδια φόρμα έχουν όμως διαφορετικές παραμέτρους και ελαφρώς διαφορετική λειτουργία .

Ξεκινώντας με την πρώτη περίπτωση ας εξηγήσουμε τις παραμέτρους της ***ShowView***:

1. ***FormName*** είναι το όνομα της φόρμας που καλεί την προβολή
2. ***Criteria*** είναι τα κριτήρια που έχει εισάγει ο χρήστης σε μορφή SQL .Η σύνθεση του string της SQL έχει ήδη γίνει και εδώ απλά περνάει το string σαν παράμετρος για να προστεθεί στο query.

Η διαδικασία εμφάνισης μιας προβολής έχει ως εξής

- Όταν καλείται η ShowView το πρώτο που γίνεται είναι η κλήση της φόρμας επιλογής της προβολής με τη συνάρτηση *SelViewID(FormName,'frm' + FormName)* .Η συνάρτηση αυτή παίρνει σαν παράμετρο το όνομα της φόρμας και το όνομα του control που στη περίπτωση αυτή είναι η ίδια η φόρμα (*'frm' + FormName)* και βρίσκει με ένα query όλα τα Views που αναφέρονται σε αυτή τη φόρμα. Στη συνέχεια τα παρουσιάζει στο χρήστη που επιλέγει κάποιο. Μόλις επιλεχθεί κάποιο view κλείνει η φόρμα και επιστρέφεται το VIEWID της προβολής που επιλέχθηκε. Το VIEWID αποθηκεύεται στη τοπική μεταβλητή ViewID και στη συνέχεια σε μια global μεταβλητή την CurrentViewID.
- Στη συνέχεια με ένα SQL query φέρνουμε την εγγραφή του πίνακα Views που αντιστοιχεί στο ViewID που επέλεξε ο χρήστης στο αντικείμενο myQuery
- Επόμενο βήμα είναι η δημιουργία του SQL query της προβολής με βάση τα χαρακτηριστικά που έχουμε πάρει για τη συγκεκριμένη προβολή με το αντικείμενο myQuery
    - Στη μεταβλητή SQLQuery αποθηκεύεται το string του ερωτήματος SQL.Αρχικά προστίθενται τα πεδία που θα επιλεχθούν και οι πίνακες όπου εμπλέκονται στο ερώτημα μέσω των συναρτήσεων *GetFields(ViewID)* και *GetTables(ViewID)* αντίστοιχα .Η συνάρτηση *GetFields(ViewID)* φέρνει στο αντικείμενο FieldsQuery τις εγγραφές του πίνακα VIEWFIELDS που αντιστοιχούν στο VIEWID που παίρνει σαν είσοδο δηλαδή φέρνει τα πεδία που νατιστοιχούν στη προβολή που επέλεξε ο χρήστης. Κάθε πεδίο έχει και ιδιότητες που θα εξεταστούν αργότερα κατά την τοποθέτηση των δεδομένων στο ListView. Τελικά διαμορφώνει τα πεδία σε ένα string μορφής SQL (πεδίο1,πεδίο2,...)

το οποίο είναι και η επιστροφή της συνάρτησης.Κάτι αντίστοιχο κάνει και η συνάρτηση *GetTables(ViewID)* για τους πίνακες του ερωτήματος. Βρίσκει όλους τους πίνακες στους οποίους ανήκουν τα πεδία του ερωτήματος και τους επιστρέφει με τη μορφή string για να προστεθεί στη μεταβλητή SQLQuery .Η συνάρτηση αυτή ελέγχει επιπλέον ώστε να μην υπάρχει στο string ο ίδιος πίνακας δύο φορές (π.χ. GLM,GLK, GLM) κάτι που θα ήταν δυνατό αφού ο ίδιος πίνακας μπορεί να αναφέρεται σε περισσότερα από ένα πεδία (π.χ. αν δύο πεδία της προβολής ανήκουν στον ίδιο πίνακα της βάσης).

- Επόμενο βήμα είναι η προσθήκη των κριτηρίων στο string του ερωτήματος. Πρώτα προστίθενται τα κριτήρια που είναι αποθηκευμένα στο πεδίο CRITERIA του πίνακα VIEWS (που μπορούμε να το προσπελάσουμε μέσω του myQuery όπου ήδη περιέχει την εγγραφή του VIEWS που επέλεξε ο χρήστης) και στη συνέχεια τα κριτήρια που έχει περάσει σαν παράμετρο η φόρμα που ζητάει την προβολή (μεταβλητή Criteria)
- Μετά τα κριτήρια προστίθενται τα πεδία ομαδοποίησης (GROUP BY) και ταξινόμησης (ORDER BY) του ερωτήματος SQL αν υπάρχουν.

- Στο επόμενο βήμα και αφού το string του ερωτήματος έχει ολοκληρωθεί, εκτελείται το ερώτημα μέσω του αντικειμένου dataQuery το οποίο θα περιέχει και τα δεδομένα μετά την εκτέλεση.Εαν το ερώτημα εκτελεστεί σωστά τότε καλείται η συνάρτηση *FillLV(StrCrit)* όπου γεμίζει το ListView με τα δεδομένα που πήραμε από το ερώτημα και είναι αποθηκευμένα στο dataQuery .Παίρνει σαν παράμετρο τα κριτήρια του ερωτήματος τα οποία στη συγκεκριμένη περίπτωση έχουμε κρατήσει στη μεταβλητή StrCrit για αυτόν αριβώς το σκοπό.Τα κριτήρια θα χρησιμοποιηθούν σε περίπτωση που κατά την εξέταση των ιδιοτήτων των πεδίων προκύψει η ανάγκη για δημιουργία νέου ερωτήματος με σκοπό τον υπολογισμό κάποιου μεγέθους (π.χ. για τον υπολογισμό ενός αθροίσματος όταν κάποιο πεδίου είναι τέτοιου τύπου χρειάζεται ένα ερώτημα SQL με τη συνάρτηση SUM(πεδίο) το οποίο ερώτημα θα πρέπει να περιλαμβάνει τα κριτήρια με τα οποία επιλέχθηκαν τα υπόλοιπα δεδομένα εκτός του αθροίσματος).Στη φόρμα των προβολών υπάρχουν δύο ListViews: το ένα είναι το LV όπου και τοποθετούνται τα κύρια δεδομένα και το άλλο είναι το sumLV στο οποίο τοποθετούνται τα αθροίσματα και ουσιαστικά έχει μία μόνο γραμμή.Μέσα στην συνάρτηση FillLV αφού αρχικοποιηθούν τα δύο ListViews (δημιουργία στηλών με βάση τη σειρά των πεδίων (TURN), ρύθμιση πλάτους στηλών) τοποθετούνται τα δεδομένα του dataQuery (κύρια δεδομένα) στο LV. Μετά την τοποθέτηση των δεδομένων ελέγχεται ο τύπος (ο έλεγχος ουσιαστικά είναι μια case αλλά η case δουλεύει μόνο με integer μεταβλητές γι'αυτό βάζουμε πολλές if..then…else) κάθε πεδίου της προβολής (δηλαδή κάθε στήλης).Αν ο τύπος είναι διαφορετικός από «N» (Το «N» αντιστοιχεί στο κανονικό πεδίο) τότε ανάλογα με το τύπο του οδηγείται στην ανάλογη ενέργεια (π.χ. αν ο τύπος είναι "S" το δημιουργείται ένα ερώτημα SQL που αθροίζει τα δεδομένα αυτού του πεδίου και επιστρέφει το άθροισμα.Εδώ χρησιμοποιούνται τα κριτήρια που εισέρχονται ως παράμετρος για να

προσδιοριστούν τα δεδομένα που πρέπει να αθροιστούν).Στο σημείο αυτό μπορούμε να φτιάξουμε όσους τύπους αρχείων θέλουμε αρκεί να γράψουμε το κώδικα για το τι θέλουμε να μας παρουσιάζει κάθε τύπος (να προσθέσουμε δηλαδή μια if που θα εκτελεί αυτό που θέλουμε).

- Τέλος το μόνο που μένει να γίνει είναι η εμφάνιση της φόρμας με το *ShowModal;*

Όσον αφορά τη δεύτερη περίπτωση , δηλαδή την κλήση της προβολής σαν πίνακα βοήθειας, οι διαφορές από την ShowView είναι μικρές.Μια από αυτές είναι και οι παράμετροι:

1. ***FormName*** είναι το όνομα της φόρμας που καλεί την προβολή
2. ***CtrlName*** είναι το όνομα του control που καλεί την ShowViewForFields
3. ***Criteria*** είναι το string που ενδεχομένως έχει εισάγει ο χρήστης για να διευκολύνει την αναζήτηση (π.χ. εισάγει το πρώτο γράμμα αυτού που ψάχνει με σκοπό να κάνει πιο συγκεκριμένη την αναζήτηση βρίσκοντας μόνο εκείνα που ξεκινούν από αυτό το γράμμα)
4. Η επιστροφή της συνάρτησης είναι το string που επιστρέφεται στην καλούσα φόρμα για να τοποθετηθεί στο control στο οποίο αναφέρεται ο πίνακας βοήθειας

Οι διαφορές στη διαδικασία σε σχέση με τη ShowView είναι βασικά η δημιουργία των κριτηρίων και ο επιπλέον κώδικας για την επιστροφή του αποτελέσματος.

Η διαφορά προέρχεται από το γεγονός ότι τα controls στη παράμετρο Criteria δεν εισάγουν τα κριτήρια έτοιμα σε μορφή SQL αλλά περνάνε μόνο στο string στο οποίο θα εφαρμοστεί το LIKE της SQL. Το πεδίο στο οποίο αναφέρεται το LIKE (δηλαδή σε πεδίο του πίνακα θα ψάξει το ερώτημα να ταιριάξει το string που εισήγαγε ο χρήστης) υπάρχει στο πεδίο FORMCRITFIELD του πίνακα VIEWS και λαμβάνεται από το αντικείμενο myQuery που περιέχει τα δεδομένα του πίνακα VIEWS .Έτσι δημιουργείται το string των κριτηρίων αφού προστεθεί και η σύνταξη του κώδικα SQL ως εξής:

StrCrit := ' WHERE ' + myQuery.fieldByName('FORMCRITFIELD').AsString + ' LIKE'" + Criteria + '%"; .

Όσον αφορά τώρα την επιστροφή της συνάρτησης το πρώτο που γίνεται είναι ο προσδιορισμός του πεδίου της επιλεγμένης εγγραφής που θα επιστραφεί.Η πληροφορία αυτή είναι αποθηκευμένη στο πεδίο RESULTFIELD του πίνακα VIEWS και προσπελάζεται και αυτή μέσω του myQuery που έχει φέρει τη σχετική εγγραφή του πίνακα VIEWS.Για να μπορούμε να προσδιορίσουμε ποια εγγραφή επέλεξε ο χρήστης στο LV στο event του ListView για το διπλό κλικ (LVDblClick) στην μεταβλητή SelIndex αποθηκεύεται ο αριθμός της εγγραφής του LV που είναι επιλεγμένη και στη συνέχεια κλείνει η φόρμα .Η συνάρτηση LVDblClick εκτός από το event του διπλού κλικ καλείται και όταν ο χρήστης πατήσει enter με αποτέλεσμα όταν γίνει διπλό κλικ ή πατηθεί το enter και υπάρχει επιλεγμένη εγγραφή , αποθηκεύεται το index της εγγραφής στο SelIndex και κλείνει η φόρμα .Κλείνοντας η φόρμα βρισκόμαστε στη συνάρτηση ShowViewForFields ακριβώς μετά το ShowModal.Στο σημείο αυτό ελέγχεται το SelIndex και επειδή οι εγγραφές στο LV έχουν την ίδια σειρά με τις στο dataQuery επιστρέφουμε από την εγγραφή του dataQuery με αριθμό SelIndex το πεδίο που ορίζεται που ορίζεται από το ResultField.

3.1.3 Η συνάρτηση *MoveToTable*

Ένα ακόμα σημείο που πρέπει να εξηγηθεί είναι και η συνάρτηση MoveToTable. Η συνάρτηση αυτή καλείται όταν ο χρήστης πληκτρολογήσει τους χαρακτήρες «Π» ή «π».Η κλήση της βρίσκεται στο event KeyDown όταν οι χαρακτήρες είναι οι προαναφερθέντες.Το πρώτο πράγμα που ελέγχει είναι αν υπάρχει επλεγμένη εγγραφή στο LV.Στη συνέχεια αφού φορτωθεί (ξανά για είμαστε σίγουροι) στο myQuery η σχετική εγγραφή του πίνακα VIEWS (με κριτήριο το CurrViewID που περιέχει το VIEWID της προβολής που έχει επιλέξει ο χρήστης) δημιουργείται με βάση τα περιεχόμενα του πεδίου KEYS ένα ερώτημα στο πίνακα VIEWFIELDS που θα φέρει τις εγγραφές που αντιστοιχούν στα πεδία που περιέχονται στο string του πεδίου KEYS.Χρειαζόμαστε αυτές τις εγγραφές του VIEWFIELDS για να μπορούμε να αντιστοιχήσουμε το FIELDNAME (που είναι και το όνομα της στήλης στο LV) με το πραγματικό όνομα του πεδίου στη βάση , δηλαδή το DBFIELD (στο πεδίο KEYS τα κλειδιά είναι με το FIELDNAME τους όμως εμείς θα χρειαστούμε το πραγματικό DBFIELD).Αφού φέρουμε τα πεδία κλειδιά στο αντικείμενο KeyFieldsQuery μπορούμε να κατακευάσουμε το string του ερωτήματος με το οποίο θα φέρουμε ολόκληρη την εγγραφή που έχει επιλέξει ο χρήστης, από το βασικό πίνακα. Το ποιά στήλη του LV αντιστοιχεί σε ποιο πεδίο κλειδί μπορεί να αντιμετωπιστεί χάρη στο πεδίο TURN αφού η σειρά των στηλών στο LV έχει φτιαχτεί με βάση το TURN των πεδίων. Αφού φτιαχτεί το ερώτημα SQL ελέγχεται το ποιός είναι ο βασικός πίνακας (MAINTABLE) και ανάλογα θα πρέπει να κληθεί και η σχετική με αυτόν φόρμα.Ουσιαστικά είναι μια case του τύπου "αν το MAINTABLE είναι xxx τότε κάλεσε τη φόρμα yyy" αλλά επειδή η case ελέγχει μόνο integer μεταβλητές χρησιμοποιούμε if…then…else . Στη συνέχεια το ερώτημα SQL εκτελείται στο αντικείμενο τύπου TQuery της φόρμας η οποία θα κληθεί και στη συνέχεια καλείται η συνάρτηση που γεμίζει τη φόρμα από το σχετικό αντικείμενο (π.χ. για τη φόρμα 110 το αντικείμενο που παριστάνει τη τρέχουσα εγγραφή είναι το CurrRecQuery και η συνάρτηση που γεμίζει τη φόρμα από αυτό είναι η FillFormFromQuery).Τέλος το μόνο που μένει να γίνει είναι η εμφάνιση της φόρμας που είναι ήδη γεμάτη με τα δεδομένα που θέλουμε.

3.2 Η διεπαφή επιλογής της προβολής (TfrmSelView)

```
unit formSelView;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, ComCtrls, Buttons,TemplateForm;

type
  TfrmSelView = class(TfrmTemplate)
    LV: TListView;
    btnOK: TBitBtn;
    procedure LVKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure LVDblClick(Sender: TObject);
```

```
    procedure btnOKClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    function ShowSelView():String;
  end;

var
  frmSelView: TfrmSelView;

implementation

{$R *.DFM}

function TfrmSelView.ShowSelView():String;
begin
  Result := '';
  FormStyle := fsNormal;
  Visible := false;
  if ShowModal = mrok then
    if (LV.Selected <> nil) then Result := LV.Selected.Caption ;
end;

procedure TfrmSelView.LVKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = 13 then
  begin
    if LV.Selected = nil then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONWARNING)
    else
    begin
      Close;
      self.ModalResult := mrOk;
    end;
  end
  else if Key = 27 then Close;
end;

procedure TfrmSelView.LVDblClick(Sender: TObject);
begin
  if LV.Selected = nil then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONWARNING)
  else
  begin
    Close;
    self.ModalResult := mrOk;
  end;
end;

procedure TfrmSelView.btnOKClick(Sender: TObject);
begin
  if LV.Selected = nil then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONWARNING)
  else
  begin
    Close;
    self.ModalResult := mrOk;
  end;
end;
```

```
end.
```

3.3 Η διεπαφή συντήρησης των προβολών (TfrmAddViews)

```
unit formAddViews;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  TemplateForm, ExtCtrls, Db, DBTables, ComCtrls,
StdCtrls,formAddFieldsToView,
  Buttons;

type
  TfrmAddViews = class(TfrmTemplate)

    edFormName: TEdit;
    edCtrlName: TEdit;
    edViewID: TEdit;
    edViewDesc: TEdit;
    edCriteriaField: TEdit;
    edCriteria: TEdit;
    cmbKeys: TComboBox;
    edResultField: TEdit;
    edOrderBy: TEdit;
    lbxCriteria: TListBox;
    lbxKeys: TListBox;
    lbFormName: TLabel;
    lbControlName: TLabel;
    lbViewID: TLabel;
    lbViewDesc: TLabel;
    lbOrderBy: TLabel;
    lbResultField: TLabel;
    lbKeys: TLabel;
    lbCriteria: TLabel;
    edMainTable: TEdit;
    lbMainTable: TLabel;
    cmbConnectors: TComboBox;
    lbFormCritField: TLabel;
    edFormCritField: TEdit;
    LV: TListView;
    lbGroupBy: TLabel;
    edGroupBy: TEdit;
    btnAddField: TBitBtn;
    btnAddCriteria: TBitBtn;
    btnAddKey: TBitBtn;
    btnInsert: TBitBtn;
    btnUpdate: TBitBtn;
    btnDelete: TBitBtn;
    lbLanguageID: TLabel;
    edLanguage: TEdit;
    procedure KeyFieldsExit(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure GlobalKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
```

```
    procedure cmbKeysEnter(Sender: TObject);
    procedure LVKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure btnAddFieldClick(Sender: TObject);
    procedure btnAddCriteriaClick(Sender: TObject);
    procedure btnAddKeyClick(Sender: TObject);
    procedure btnInsertClick(Sender: TObject);
    procedure btnUpdateClick(Sender: TObject);
    procedure btnDeleteClick(Sender: TObject);
  private
    { Private declarations }
    function CheckView():boolean;
    procedure
FillLB(lb:TListBox;FieldName:string;DBSeparator:string);
    procedure FillCriteriaLB();
    procedure FillForm();
    procedure ClearForm();
    function GetLBText(lb:TListBox):string;
    procedure AddViewFields;
    procedure DeleteViewFields;
    procedure FillLV;
  public
    { Public declarations }
  end;

var
  frmAddViews: TfrmAddViews;

implementation

{$R *.DFM}

function TfrmAddViews.CheckView():boolean;
begin
  Result := false;
  if GetData(myQuery,'SELECT * FROM VIEWS WHERE FORMNAME='''
              + edFormName.Text + ''' AND CTRLNAME='''
              + edCtrlName.Text + ''' AND VIEWID='''
              + edViewID.Text + '''') then

      if myQuery.RecordCount > 0 then Result := true;
end;

procedure
TfrmAddViews.FillLB(lb:TListBox;FieldName:string;DBSeparator:string);
var
  FieldValue,tmpValue:string;
begin
  lb.Clear ;
  if myQuery.FieldByName(FieldName).AsString <> '' then
  begin
    FieldValue := myQuery.FieldByName(FieldName).AsString;
    while pos(DBSeparator,FieldValue) <> 0 do
    begin
      tmpValue := copy(FieldValue,0,pos(DBSeparator,FieldValue)-1);
      delete(FieldValue,1,pos(DBSeparator,FieldValue));
      lb.Items.Add(tmpValue);
    end;
    lb.Items.Add(FieldValue);
  end;
end;
```

```
procedure TfrmAddViews.FillForm();
begin
  edFormName.Text := myQuery.FieldByName('FORMNAME').AsString;
  edCtrlName.Text := myQuery.FieldByName('CTRLNAME').AsString;
  edViewID.Text := myQuery.FieldByName('VIEWID').AsString;
  edViewDesc.Text := myQuery.FieldByName('VIEWDESC').AsString;
  edLanguage.Text := myQuery.FieldByName('LANGUAGEID').AsString;
  edOrderBy.Text := myQuery.FieldByName('ORDERBY').AsString;
  edGroupBy.Text := myQuery.FieldByName('GROUPBY').AsString;
  edFormCritField.Text :=
myQuery.FieldByName('FORMCRITFIELD').AsString;
  edMainTable.Text := myQuery.FieldByName('MAINTABLE').AsString;
  edResultField.Text := myQuery.FieldByName('RESULTFIELD').AsString;
  FillLB(lbxKeys,'KEYS',',');
  FillCriteriaLB;
  FillLV;
end;

procedure TfrmAddViews.FillCriteriaLB();
var
  tmpSeparator,FieldValue,tmpValue,QuotedtmpValue:string;
begin
  lbxCriteria.Clear ;
  if myQuery.FieldByName('CRITERIA').AsString <> '' then
  begin
    FieldValue := myQuery.FieldByName('CRITERIA').AsString;
    while (pos(' AND ',FieldValue) <> 0) or (pos(' OR ',FieldValue)
<> 0) do
    begin
      if (pos(' OR ',FieldValue) <> 0) then
      begin
        if (pos(' AND ',FieldValue) < pos(' OR ',FieldValue)) and
(pos(' AND ',FieldValue) <> 0) then tmpSeparator := ' AND '
        else if (pos(' AND ',FieldValue) > pos(' OR ',FieldValue))
and (pos(' OR ',FieldValue) <> 0) then tmpSeparator := ' OR ';
      end
      else if (pos(' AND ',FieldValue)<>0) then tmpSeparator := ' AND
';

      tmpValue := copy(FieldValue,0,pos(tmpSeparator,FieldValue));
      delete(FieldValue,1,pos(tmpSeparator,FieldValue));

      QuotedtmpValue := '';
      if pos('''',tmpValue) > 0 then
      begin
        QuotedtmpValue := tmpValue;
        delete(QuotedtmpValue,1,pos('''',QuotedtmpValue)-1);
        delete(tmpValue,pos('''',tmpValue),length(tmpValue)-
pos('''',tmpValue));
        QuotedtmpValue := '''' + QuotedtmpValue + '''' ;
      end;
      lbxCriteria.Items.Add(tmpValue + QuotedtmpValue);
    end;

    QuotedtmpValue := '';
    if pos('''',FieldValue) > 0 then
    begin
      QuotedtmpValue := FieldValue;
      delete(QuotedtmpValue,1,pos('''',QuotedtmpValue));
```

```
      delete(FieldValue,pos('''',FieldValue)+1,length(FieldValue)-
pos('''',FieldValue));
      QuotedtmpValue := '''' + QuotedtmpValue + '''' ;
    end;
    lbxCriteria.Items.Add(FieldValue+QuotedtmpValue);
  end;
end;

procedure TfrmAddViews.KeyFieldsExit(Sender: TObject);
begin
  if CheckView then
  begin
    FillForm;
    btnUpdate.Enabled := true;
    btnDelete.Enabled := true;
    btnInsert.Enabled := false;
  end
  else if (edFormName.Text <> '') and (edCtrlName.Text <> '') and
(edViewID.Text <> '') then
  begin
    btnUpdate.Enabled := false;
    btnDelete.Enabled := false;
    btnInsert.Enabled := true;
  end;
end;

procedure TfrmAddViews.ClearForm();
begin
  edFormName.Text := '';
  edCtrlName.Text := '';
  edViewID.Text := '';
  edViewDesc.Text := '';
  edLanguage.Text := '';
  edOrderBy.Text := '';
  edGroupBy.Text := '';
  edFormCritField.Text := '';
  edMainTable.Text := '';
  edResultField.Text := '';
  lbxKeys.Clear ;
  cmbKeys.ItemIndex := -1;
  lbxCriteria.Clear ;
  cmbConnectors.ItemIndex := 0;
  edCriteriaField.Text := '';
  edCriteria.Text := '';
  LV.Items.Clear ;
end;

procedure TfrmAddViews.FormShow(Sender: TObject);
begin
  inherited;
  btnUpdate.Enabled := false;
  btnDelete.Enabled := false;
  btnInsert.Enabled := false;

  cmbConnectors.Clear ;
  cmbConnectors.Items.Add('AND');
  cmbConnectors.Items.Add('OR');
  cmbConnectors.ItemIndex := 0;
end;

procedure TfrmAddViews.GlobalKeyDown(Sender: TObject; var Key: Word;
```

```
      Shift: TShiftState);
begin
  if Key = 27 then
  begin
    if (Sender as TWinControl) = edFormName then Close
    else
    begin
      ClearForm ;
      edFormName.SetFocus ;
    end;
  end
  else if Key = 46 then
    if Sender is TListBox then (Sender as TListBox).Clear ;
end;

function TfrmAddViews.GetLBText(lb:TListBox):string;
var
  i:integer;
begin
  for i:=0 to lb.items.count-1 do Result := Result + lb.items[i];
end;

procedure TfrmAddViews.cmbKeysEnter(Sender: TObject);
var
  i:integer;
begin
  inherited;
  cmbKeys.Clear ;
  if LV.items.count > 0 then
    for i:=0 to LV.items.count - 1 do
cmbKeys.Items.Add(LV.items[i].SubItems[1]);
end;

procedure TfrmAddViews.AddViewFields;
var
  i:integer;

begin
  for i:=0 to LV.Items.Count - 1 do
    ExecQuery(myquery,'INSERT INTO VIEWFIELDS VALUES(''' +
                      LV.Items[i].Caption + ''',''' +
                      LV.Items[i].SubItems[0] + ''',''' +
                      LV.Items[i].SubItems[1] + ''',''' +
                      LV.Items[i].SubItems[2] + ''',''' +
                      LV.Items[i].SubItems[3] + ''',''' +
                      LV.Items[i].SubItems[4] + ''',''' +
                      LV.Items[i].SubItems[5] + ''')');
end;


procedure TfrmAddViews.DeleteViewFields;
begin
  ExecQuery(myQuery,'DELETE FROM VIEWFIELDS WHERE VIEWID=''' +
edViewID.Text + '''');
end;

procedure TfrmAddViews.LVKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if (Key = VK_DELETE) and (LV.Selected <> nil) then
```

```
  begin
    LV.Items[LV.Selected.Index].Delete ;
    LV.Selected := LV.items[0];
    LV.Items[0].Focused := true;
  end;
end;

procedure TfrmAddViews.FillLV;
var
  i:integer;
begin
  LV.Items.Clear ;
  GetData(myQuery,'SELECT * FROM VIEWFIELDS WHERE VIEWID =''' +
edViewID.Text + '''');
  if myQuery.RecordCount > 0 then
    for i:=0 to myQuery.RecordCount  -1 do
    begin
      LV.Items.Add ;
      LV.Items[i].caption := edViewID.Text ;

LV.Items[i].SubItems.Add(myQuery.FieldByName('DBFIELD').AsString) ;

LV.Items[i].SubItems.Add(myQuery.FieldByName('FIELDNAME').AsString);
      LV.Items[i].SubItems.Add(myQuery.FieldByName('TYPE').AsString);

LV.Items[i].SubItems.Add(myQuery.FieldByName('DBTABLE').AsString);
      if myQuery.FieldByName('VISIBLE').AsString = 'true' then
LV.Items[i].SubItems.Add('1')
      else LV.Items[i].SubItems.Add('0');
      LV.Items[i].SubItems.Add(myQuery.FieldByName('TURN').AsString);
      myQuery.Next ;
    end;
end;

procedure TfrmAddViews.btnAddFieldClick(Sender: TObject);
begin
  inherited;

  frmAddFieldsToView := TfrmAddFieldsToView.Create(Self);

  if frmAddFieldsToView.ShowModal  = mrOk then
  begin
    LV.Items.Add ;
    LV.Items[LV.Items.Count - 1].Caption := edViewID.Text ;
    LV.Items[LV.Items.Count -
1].SubItems.Add(frmAddFieldsToView.DBField);
    LV.Items[LV.Items.Count -
1].SubItems.Add(frmAddFieldsToView.FieldName);
    LV.Items[LV.Items.Count -
1].SubItems.Add(frmAddFieldsToView.FieldType);
    LV.Items[LV.Items.Count -
1].SubItems.Add(frmAddFieldsToView.Table);
    LV.Items[LV.Items.Count -
1].SubItems.Add(frmAddFieldsToView.Visibility);
    LV.Items[LV.Items.Count -
1].SubItems.Add(frmAddFieldsToView.Turn);
  end;
  FreeAndNil(frmAddFieldsToView);
end;

procedure TfrmAddViews.btnAddCriteriaClick(Sender: TObject);
```

```
begin
  inherited;
  if lbxCriteria.Items.Count > 0 then
    lbxCriteria.Items.Add(' ' + cmbConnectors.Text + ' ' +
edCriteriaField.Text + edCriteria.Text)
  else lbxCriteria.Items.Add(' ' + edCriteriaField.Text +
edCriteria.Text);

  edCriteriaField.Text := '';
  edCriteria.Text := '';
  cmbConnectors.ItemIndex := 0;
  cmbConnectors.SetFocus ;
end;

procedure TfrmAddViews.btnAddKeyClick(Sender: TObject);
begin
  inherited;
  if lbxKeys.Items.Count > 0 then lbxKeys.Items.Add(',' +
cmbKeys.Text)
  else lbxKeys.Items.Add(cmbKeys.Text);

  cmbKeys.ItemIndex := -1;
  cmbKeys.SetFocus ;
end;

procedure TfrmAddViews.btnInsertClick(Sender: TObject);
begin
  inherited;
  if not ExecQuery(myQuery,'INSERT INTO VIEWS
(FORMNAME,CTRLNAME,VIEWID,VIEWDESC,ORDERBY,GROUPBY,MAINTABLE,KEYS,RES
ULTFIELD,CRITERIA,FORMCRITFIELD,LANGUAGEID) VALUES(''' +
                           edFormName.Text + ''',''' +
                           edCtrlName.Text + ''',''' +
                           edViewID.Text + ''',''' +
                           edViewDesc.Text + ''',''' +
                           edOrderBy.Text + ''',''' +
                           edGroupBy.Text + ''',''' +
                           edMainTable.Text + ''',''' +
                           GetLBText(lbxKeys) + ''',''' +
                           edResultField.Text + ''',''' +
                           GetLBText(lbxCriteria) + ''',''' +
                           edFormCritField.Text + ''',''' +
                           edLanguage.Text + ''')') then


Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR)
  else
  begin
    AddViewFields;
    btnInsert.Enabled := false;
    btnUpdate.Enabled := false;
    btnDelete.Enabled := false;
    ClearForm ;
    edFormName.SetFocus ;
  end;
end;

procedure TfrmAddViews.btnUpdateClick(Sender: TObject);
begin
  inherited;
  if not ExecQuery(myQuery,'UPDATE VIEWS SET FORMNAME=''' +
```

```
                              edFormName.Text + ''',CTRLNAME=''' +
                              edCtrlName.Text + ''',VIEWID=''' +
                              edViewID.Text + ''',VIEWDESC=''' +
                              edViewDesc.Text + ''',LANGUAGEID=''' +
                              edLanguage.Text + ''',ORDERBY=''' +
                              edOrderBy.Text  + ''',GROUPBY=''' +
                              edGroupBy.Text + ''',FORMCRITFIELD=''' +
                              edFormCritField.Text + ''',MAINTABLE='''
+
                              edMainTable.Text + ''',KEYS=''' +
                              GetLBText(lbxKeys) + ''',RESULTFIELD='''
+
                              edResultField.Text + ''',CRITERIA=''' +
                              GetLBText(lbxCriteria) + ''' WHERE
FORMNAME=''' +

edFormName.Text + ''' AND CTRLNAME=''' +

edCtrlName.Text + ''' AND VIEWID=''' +
                                                          edViewID.Text
+ ''''') then


Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR)
  else
  begin
    DeleteViewFields ;
    AddViewFields ;
    btnInsert.Enabled := false;
    btnUpdate.Enabled := false;
    btnDelete.Enabled := false;
    ClearForm ;
    edFormName.SetFocus ;
  end;
end;

procedure TfrmAddViews.btnDeleteClick(Sender: TObject);
begin
  inherited;
  if not ExecQuery(myQuery,'DELETE FROM VIEWS WHERE FORMNAME=''' +
                              edFormName.Text + ''' AND CTRLNAME=''' +
                              edCtrlName.Text + ''' AND VIEWID=''' +
                              edViewID.Text + ''''') then


Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR)
  else
  begin
    DeleteViewFields ;
    btnInsert.Enabled := false;
    btnUpdate.Enabled := false;
    btnDelete.Enabled := false;
    ClearForm ;
    edFormName.SetFocus ;
  end;
end;

end.
```

3.4 Η διεπαφή προσθήκης πεδίου σε προβολή (TfrmAddFieldsToView)

```
unit formAddFieldsToView;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, TemplateForm, StdCtrls, Buttons, DB, DBTables;

type
  TfrmAddFieldsToView = class(TfrmTemplate)
    DBTable: TLabel;
    lbFieldType: TLabel;
    lbFieldName: TLabel;
    lbDBField: TLabel;
    lbVisible: TLabel;
    lbTurn: TLabel;
    edDBField: TEdit;
    edFieldName: TEdit;
    EdType: TEdit;
    edTable: TEdit;
    edVisible: TEdit;
    edTurn: TEdit;

    btnAddField: TBitBtn;
    procedure btnAddFieldClick(Sender: TObject);
    procedure GlobalKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
    DBField,FieldName,FieldType,Table,Visibility,Turn : string;
  end;

var
  frmAddFieldsToView: TfrmAddFieldsToView;

implementation

{$R *.dfm}

procedure TfrmAddFieldsToView.GlobalKeyDown(Sender: TObject; var Key:
Word;
      Shift: TShiftState);
begin
  if Key = 27 then
  begin
    if (Sender as TWinControl) = edDBField then Close
    else
    begin
      edDBField.Text := '';
      edFieldName.Text := '';
      EdType.Text := '';
      edTable.Text := '';
      edVisible.Text := '';
      edTurn.Text := '';
```

```
      edDBField.SetFocus ;
    end;
  end;
end;

procedure TfrmAddFieldsToView.btnAddFieldClick(Sender: TObject);
begin
  inherited;
  DBField := edDBField.Text ;
  FieldName := edFieldName.Text ;
  FieldType := edType.Text;
  Table := edTable.Text;
  Visibility := edVisible.Text ;
  Turn := edTurn.Text;
end;

end.
```

4. Το υποσύστημα της λογιστικής

4.1 Η διεπαφή διαχείρισης των λογαριασμών (TfrmAccounts)

```
unit Accounts;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  TemplateForm, Db, DBTables, StdCtrls, ExtCtrls, Mask, ComCtrls,
  AccountMaskEdit, Buttons, Spin, LimitSpinEdit,formViews,NotesForm,
  ToolWin;

type
  TfrmAccounts = class(TfrmTemplate)
    edTitle: TEdit;
    lbAccepts_Records: TLabel;
    lbAccount_Type: TLabel;
    lbProtection: TLabel;
    lbRecord_Check: TLabel;
    lbLink: TLabel;
    pnlAccepts_Records: TPanel;
    rbNaiDE: TRadioButton;
    rbOxiDE: TRadioButton;
    LV: TListView;
    cmbProtection: TComboBox;
    cmbLink: TComboBox;
    cmbRecord_Check: TComboBox;
    lbCurrency: TLabel;
    medAccount: TAccountMaskEdit;
    cmbAccount_Type: TComboBox;
    sbtnCurrency: TSpeedButton;
    edPCurrency: TEdit;
    medCurrency: TMaskEdit;
    medCustomer: TMaskEdit;
    CurrRecQuery: TQuery;
    lbL_P_Print_Level: TLabel;
    lbBalance_Print_Level: TLabel;
```

```
    lbValues_Include_Tax: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    sedL_P_Print_Level: TLimitSpinEdit;
    sedBalance_Print_Level: TLimitSpinEdit;
    sedValues_Include_Tax: TLimitSpinEdit;
    lbTitle: TLabel;
    lbAccount: TLabel;
    memNotes: TMemo;
    lbNotes: TLabel;
    sbtnNotes: TSpeedButton;
    ToolBar: TToolBar;
    tbtnInsert: TToolButton;
    tbtnAccept: TToolButton;
    tbtnDelete: TToolButton;
    tbtnSeparator: TToolButton;
    tbtnExit: TToolButton;
    procedure rbNaiDEEnter(Sender: TObject);
    procedure rbOxiDEEnter(Sender: TObject);
    procedure medAccountExit(Sender: TObject);
    procedure medAccountKeyDown(Sender: TObject; var Key: Word;Shift:
TShiftState);
    procedure medAccountKeyUp(Sender: TObject; var Key: Word;Shift:
TShiftState);
    procedure GlobalKeyDown(Sender: TObject; var Key: Word;Shift:
TShiftState);
    procedure Enter(Sender: TObject);
    procedure cmbLinkChange(Sender: TObject);
    procedure edTitleKeyUp(Sender: TObject; var Key: Word;Shift:
TShiftState);
    procedure edTitleExit(Sender: TObject);
    procedure LVSelectItem(Sender: TObject; Item: TListItem;Selected:
Boolean);
    procedure sbtnCurrencyClick(Sender: TObject);
    procedure edPCurrencyKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
    procedure FormShow(Sender: TObject);
    procedure btnEisagwgiClick(Sender: TObject);
    procedure btnApodoxiClick(Sender: TObject);
    procedure btnDiagrafiClick(Sender: TObject);
    procedure btnExodosClick(Sender: TObject);
    procedure sbtnNotesClick(Sender: TObject);
    procedure memNotesEnter(Sender: TObject);
    procedure tbtnInsertClick(Sender: TObject);
    procedure tbtnAcceptClick(Sender: TObject);
    procedure tbtnDeleteClick(Sender: TObject);
    procedure tbtnExitClick(Sender: TObject);
    procedure LVKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure medAccountEnter(Sender: TObject);
  private

varAccepts_Records,varAccount_Type,varRecord_Check,varProtection,varL
ink:widestring;
    varNotes: array[1..7] of string;
    varL_P_Print_Level,varBalance_Print_Level,varValues_Include_Tax :
String;
    Level:integer;
    Changes:boolean;
    prev_account:string;
    procedure FillLV();
```

```
    procedure InitiallizeControls();
    procedure UpdateVariablesFromForm();
    function  CheckForParentAccount():bool;
    function  CheckCurrency():bool;
    procedure ControlsEnabled(state : bool);
    procedure SearchFieldExit();
  public
    { Public declarations }
    procedure FillFormFromQuery();
  end;

var
  frmAccounts: TfrmAccounts;

implementation

{$R *.DFM}

//Γεμίζει το ListView από το myQuery
procedure TfrmAccounts.FillLV();
var
  i:integer;
begin
  myQuery.First;
  LV.Items.Clear;
  for i:=0 to myQuery.RecordCount-1 do
  begin
    LV.Items.Add;
    LV.items.Item[i].Caption :=
myQuery.FieldByName('Account').AsString;
    LV.Items.Item[i].SubItems.Add
(myQuery.FieldByName('Title').AsString);
    myQuery.Next;
  end;
end;

//Το event onEnter του radiobutton που αντιστοιχει στο "Ναι"
procedure TfrmAccounts.rbNaiDEEnter(Sender: TObject);
begin
  inherited;
  Enter(Sender);
  //Ενεργοποιεί τα controls που σχετίζονται με το αν δέχεται εγγραφές
  ControlsEnabled(true);
end;

//Το event onEnter του radiobutton που αντιστοιχει στο "Όχι"
procedure TfrmAccounts.rbOxiDEEnter(Sender: TObject);
begin
  inherited;
  Enter(Sender);
  //Απενεργοποιεί τα controls που σχετίζονται με το αν δέχεται
εγγραφές
  ControlsEnabled(false);
end;

//Εκτελεί τους ελέγχους κατά την έξοδο από το πεδίο του λογαριασμού
procedure TfrmAccounts.medAccountExit(Sender: TObject);
begin
  inherited;
  //Ελέγχει αν ο λογαριασμός είναι στο σωστό format (ελέγχει το μήκος
του)
```

```
  if (Length(medAccount.Account) in [3,6,9,12,17]) then
  begin
    if GetData(CurrRecQuery,'SELECT * FROM GLM WHERE Account=''' +
medAccount.Account + '''') then
    begin
      SearchFieldExit();
      if (prev_account <> medAccount.Account) then Changes:=false;
    end
    else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

    //Αν ο λογαριασμός είναι πέμπτου βαθμού τότε απενεργοποιεί τα
radiobuttons
    if Length(medAccount.Account) = 17 then
    begin
      ControlsEnabled(true);
      rbNaiDE.Enabled := false;
      rbOxiDE.Enabled := false;
    end;
  end
  else InitiallizeControls ;
end;

//Γεμίζει τη φόρμα από το CurrRecQuery
procedure TfrmAccounts.FillFormFromQuery();
begin
   InitiallizeControls ;
   medAccount.Text := CurrRecQuery.FieldByName('Account').AsString ;

   edTitle.Text := CurrRecQuery.FieldByName('Title').AsString ;

   if CurrRecQuery.FieldByName('Account_Type').AsString = 'A' then
      cmbAccount_Type.ItemIndex :=
cmbAccount_Type.Items.IndexOf(GetListValue('cmbAccount_Type','Analyti
kes'))
   else if CurrRecQuery.FieldByName('Account_Type').AsString = 'S'
then
      cmbAccount_Type.ItemIndex :=
cmbAccount_Type.Items.IndexOf(GetListValue('cmbAccount_Type','Sygentr
otikes'))
   else cmbAccount_Type.ItemIndex := -1;

   if CurrRecQuery.FieldByName('Record_Check').AsString = 'O' then
     cmbRecord_Check.ItemIndex :=
cmbRecord_Check.Items.IndexOf(GetListValue('cmbRecord_Check','Oxi'))
   else if CurrRecQuery.FieldByName('Record_Check').AsString = 'K'
then
     cmbRecord_Check.ItemIndex :=
cmbRecord_Check.Items.IndexOf(GetListValue('cmbRecord_Check','Katathe
seis'))
   else if CurrRecQuery.FieldByName('Record_Check').AsString = 'X'
then
     cmbRecord_Check.ItemIndex :=
cmbRecord_Check.Items.IndexOf(GetListValue('cmbRecord_Check','Xorigis
eis'))
   else if CurrRecQuery.FieldByName('Record_Check').AsString = 'M'
then
     cmbRecord_Check.ItemIndex :=
cmbRecord_Check.Items.IndexOf(GetListValue('cmbRecord_Check','Merides
'))
   else cmbRecord_Check.ItemIndex := -1;
```

```
   if CurrRecQuery.FieldByName('Protection').AsString = 'X' then
      cmbProtection.ItemIndex :=
cmbProtection.Items.IndexOf(GetListValue('cmbProtection','Xreosi'))
   else if CurrRecQuery.FieldByName('Protection').AsString = 'P' then
      cmbProtection.ItemIndex :=
cmbProtection.Items.IndexOf(GetListValue('cmbProtection','Pistosi'))
   else if CurrRecQuery.FieldByName('Protection').AsString = 'O' then
      cmbProtection.ItemIndex :=
cmbProtection.Items.IndexOf(GetListValue('cmbProtection','Oxi'))
   else cmbProtection.ItemIndex := -1;

   if CurrRecQuery.FieldByName('Link').AsString = 'O' then
     cmbLink.ItemIndex :=
cmbLink.Items.IndexOf(GetListValue('cmbLink','Oxi'))
   else if CurrRecQuery.FieldByName('Link').AsString = 'E' then
     cmbLink.ItemIndex :=
cmbLink.Items.IndexOf(GetListValue('cmbLink','Pelates'))
   else if CurrRecQuery.FieldByName('Link').AsString = 'P' then
     cmbLink.ItemIndex :=
cmbLink.Items.IndexOf(GetListValue('cmbLink','Promitheytes'))
   else cmbLink.ItemIndex := -1;

   medCustomer.Text := CurrRecQuery.FieldByName('Customer').AsString;
   if Length(medAccount.Account) = 3 then
   begin
     sedL_P_Print_Level.Text :=
CurrRecQuery.FieldByName('L_P_Print_Level').AsString;
     sedBalance_Print_Level.Text :=
CurrRecQuery.FieldByName('Balance_Print_Level').AsString;
     sedValues_Include_Tax.Text :=
CurrRecQuery.FieldByName('Values_Include_Tax').AsString;
   end;

   medCurrency.Text := CurrRecQuery.FieldByName('Currency').AsString;
   CheckCurrency;

   VarNotes[1] := CurrRecQuery.FieldByName('Notes_Level_1').AsString;
   VarNotes[2] := CurrRecQuery.FieldByName('Notes_Level_2').AsString;
   VarNotes[3] := CurrRecQuery.FieldByName('Notes_Level_3').AsString;
   VarNotes[4] := CurrRecQuery.FieldByName('Notes_Level_4').AsString;
   VarNotes[5] := CurrRecQuery.FieldByName('Notes_Level_5').AsString;
   VarNotes[6] := CurrRecQuery.FieldByName('Notes_Level_6').AsString;
   VarNotes[7] := CurrRecQuery.FieldByName('Notes_Level_7').AsString;
   memNotes.Text := VarNotes[level];

   //Αυτό μπαίνει τελευταίο για να ελέγξει τα κουμπιά ανάλογα με τα
δεδομένα

   rbNaiDE.Enabled := true;
   rbOxiDE.Enabled := true;

   //Αν δέχεται εγγραφές ενεργοποιεί τα comboboxes
   if CurrRecQuery.FieldByName('Accepts_Records').AsString = 'N' then
ControlsEnabled(true)
   else ControlsEnabled(false);

   //Αν είναι πέμπτου βαθμού απενεργοποιεί ta radiobuttons
   if Length(medAccount.account) = 17 then
   begin
     rbNaiDE.Enabled := false;
```

```
      rbOxiDE.Enabled := false;
    end;

    //Με το τρόπο αυτό ελέγχει αν έχει υπολογαριασμούς και αν έχει
απενεργοποιεί τα radiobuttons
    //(αν υπάρχουν λογαριασμοί που εχουν πρόθεμα το λογαριασμό που
εξετάζουμε τότε εχει υπολογαριασμούς)
    if GetData(myQuery,'SELECT Account,Title FROM GLM WHERE Account
LIKE''' + medAccount.Account  + '%'' ORDER BY Account') then
    begin
      if myQuery.RecordCount > 1 then
      begin
        rbNaiDE.Enabled := false;
        rbOxiDE.Enabled := false;
      end;
    end;

end;

//Αρχικοποιεί τα controls
procedure TfrmAccounts.InitiallizeControls();
begin
  edTitle.Text := '';
  if rbOxiDE.Enabled then rbOxiDE.Checked := true;
  memNotes.Text := '';
  sedL_P_Print_Level.Value := 0;
  sedBalance_Print_Level.Value := 0;
  sedValues_Include_Tax.Value := 0;
  ControlsEnabled(false);
end;

//Επειδή στη βάση κάποια πεδία αποθηκεύονται με char(1) πρέπει να
κρατάμε σε εσωτερικές μεταβλητές
//το character που αντιστοιχεί σε συτό που είναι επιλεγμένο.Τις
μεταβλητές αυτές θα περάσουμε στη βάση.
//Η συνάρτηση αυτή απλά ενημερώνει αυτές τις μεταβλητές ανάλογα με το
τι είναι επιλεγμένο τη στιγμή που καλείται
procedure TfrmAccounts.UpdateVariablesFromForm();
begin
  if rbNaiDE.Checked = true then
  begin
     varAccepts_Records := 'N';

     if cmbAccount_Type.Text =
GetListValue('cmbAccount_Type','Analytikes')  then varAccount_Type :=
'A'
     else if cmbAccount_Type.Text =
GetListValue('cmbAccount_Type','Sygentrotikes') then varAccount_Type
:= 'S';

     if
cmbRecord_Check.ItemIndex=cmbRecord_Check.Items.IndexOf(GetListValue(
'cmbRecord_Check','Katatheseis')) then varRecord_Check := 'K'
     else if
cmbRecord_Check.ItemIndex=cmbRecord_Check.Items.IndexOf(GetListValue(
'cmbRecord_Check','Xorigiseis')) then varRecord_Check := 'X'
     else if
cmbRecord_Check.ItemIndex=cmbRecord_Check.Items.IndexOf(GetListValue(
'cmbRecord_Check','Merides')) then varRecord_Check := 'M'
```

```
     else if
cmbRecord_Check.ItemIndex=cmbRecord_Check.Items.IndexOf(GetListValue(
'cmbRecord_Check','Oxi')) then varRecord_Check := 'O';

     if cmbProtection.ItemIndex =
cmbProtection.Items.IndexOf(GetListValue('cmbProtection','Xreosi'))
then varProtection := 'X'
     else if cmbProtection.ItemIndex =
cmbProtection.Items.IndexOf(GetListValue('cmbProtection','Pistosi'))
then varProtection := 'P'
     else if cmbProtection.ItemIndex =
cmbProtection.Items.IndexOf(GetListValue('cmbProtection','Oxi')) then
varProtection := 'O';

     if
cmbLink.ItemIndex=cmbLink.Items.IndexOf(GetListValue('cmbLink','Promi
theytes')) then varLink := 'P'
     else if
cmbLink.ItemIndex=cmbLink.Items.IndexOf(GetListValue('cmbLink','Pelat
es')) then varLink := 'E'
     else if
cmbLink.ItemIndex=cmbLink.Items.IndexOf(GetListValue('cmbLink','Oxi')
) then varLink := 'O';

  end
  else
  begin
     varAccepts_Records := 'O';
     varAccount_Type := '';
     varRecord_Check := '';
     varProtection := '';
     varLink := '';
  end;

  if sedL_P_Print_Level.Enabled then
  begin
    varL_P_Print_Level := sedL_P_Print_Level.Text;
    varBalance_Print_Level := sedBalance_Print_Level.Text;
    varValues_Include_Tax := sedValues_Include_Tax.Text;
  end
  else
  begin
    varL_P_Print_Level := 'NULL';
    varBalance_Print_Level := 'NULL';
    varValues_Include_Tax := 'NULL';
  end;

  varNotes[level] := trim(memNotes.Text) ;
end;

//Η συνάρτηση αυτή ελέγχει για το αν υπάρχει ο λογαριασμός και αν
δέχεται εγγραφές.
//Χρησιμοποιείται όταν χρήστης συμπληρώνει το πεδίο του
λογαριασμού.Καλείται όταν ο χρήστης
//προσπαθεί να βάλει λογαριασμό μεγαλύτερου βαθμού και ελέγχει αν
υπάρχει αυτό που έχει ήδη εισάγει κα αν δέχεται εγγραφές
function TfrmAccounts.CheckForParentAccount():bool;
begin
  if GetData(myQuery,'SELECT Account,Accepts_Records FROM GLM WHERE
Account=''' + medAccount.Account + '''') then
    if myQuery.RecordCount >0 then
```

```
    begin
      if myQuery.FieldByName('Accepts_Records').AsString = 'O' then
        CheckForParentAccount := true
      else
        CheckForParentAccount := false;
    end
    else
    begin
      CheckForParentAccount := false;
    end
  else CheckForParentAccount := false;
end;

//Το event αυτό εκτελείται όταν γράφει στο πεδίο του λογαριασμού
procedure TfrmAccounts.medAccountKeyDown(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
begin
  inherited;
  //Εδώ εκχωρούμε το αποτέλεσμα του ελέγχου στο property του conrtol
του λογαριασμού.Όταν αυτό το property είναι
  //false δεν τον αφήνει να συνεχίσει
  medAccount.ExistParentAccount := CheckForParentAccount ;
  GlobalKeyDown(Sender,Key,Shift);
end;

//Και αυτή η συνάρτηση εκτελείται όταν γράφει στο control του
λογαριασμού αλλά μετά την προηγούμενη
//Γεμίζει το ListView με όλους τους λογαριασμός που έχουν πρόθεμα
αυτό που έχει γράψει ήδη
procedure TfrmAccounts.medAccountKeyUp(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
begin
  inherited;
  if GetData(myQuery,'SELECT Account,Title FROM GLM WHERE Account
LIKE''' + medAccount.Account  + '%'' ORDER BY Account') then
  begin
    if (Length(medAccount.Account) in [3,6,9,12,17]) then FillLV
    else
    begin
      FillLV;
      tbtnInsert.Enabled := false;
    end;
  end;
end;

//Αυτή η συνάρτηση καλείται από όλα τα onKeyDown events όλων των
controls
procedure TfrmAccounts.GlobalKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if key = 13 then SwitchControls(Sender) ;  //Αν πατήθηκε το enter
πήγαινε στο επόμενο control
  if key = 27 then  //Αν πατήθηκε το backspace τότε
  begin
    if medAccount.Focused then Close   //Αν είναι στο πρώτο πεδίο
κλείνει η φόρμα
    else                              //αλλιώς καθαρίζει τη φόρμα
    begin
      InitiallizeControls ;
```

```
      medAccount.SetFocus ;
      medAccount.text := '';
    end;
  end;
  //Αν πατηθεί το επάνω ή κάτω βέλος τότε στέλνει το focus στο LV και
επιλέγει το πρώτο στη λίστα
  if (Key in [38,40]) and (((Sender as TWinControl).Name =
'medAccount') or ((Sender as TWinControl).Name = 'edTitle')) then
  begin
    LV.SetFocus ;
    LV.Selected := LV.Items[0];
    Lv.Items[0].Focused := true;
  end;
end;

//Με αυτή τη συνάρτηση είναι συνδεδεμένα όλα τα onEnter events των
Controls της φόρμας
procedure TfrmAccounts.Enter(Sender: TObject);
begin
  inherited;

  //Ελέγχει μέσω της CheckCurrency αν το νόμισμα έχει συμπληρωθεί
σωστά εκτός αν το πεδίο του
  //νομίσματος είναι απενεργοποιημένο (όταν δηλαδή δεν χρειάζεται να
συμπληρωθεί το νόμισμα)
  if (not CheckCurrency) and (medCurrency.Enabled = true) then
  begin
    if (Sender as TWinControl).Name <> 'edPCurrency' then
      if medCurrency.TabOrder < (Sender as TWinControl).TabOrder then
medCurrency.SetFocus ;
    medCurrency.Text := '';
  end;

  //Ελέγχει αν ο λογαριασμός που είναι γραμμένος αυτή τη στιγμή είναι
σωστός
  if not (Length(medAccount.Account) in [0,3,6,9,12,17]) then
medAccount.SetFocus ;
end;

//Ελέγχει αν υπάρχει στον GLP αυτό που έχει γράψει ο χρήστης στο
πεδίο του νομίσματος
function TfrmAccounts.CheckCurrency():bool;
begin
  Result :=
CheckIfExists(myQuery,'CURRENCY',medCurrency.Text,(Application.MainFo
rm.FindComponent('UserQuery') as
TQuery).fieldByName('LanguageID').AsString);

  if Result then edPCurrency.Text :=
myQuery.fieldByName('Description').AsString
  else
  begin
    medCurrency.Text := '';
    edPCurrency.Text := '';
  end;
end;

//Το event onChange του combobox της σύνδεσης
procedure TfrmAccounts.cmbLinkChange(Sender: TObject);
var
  tmpAccountText :string;
```

```
begin
  inherited;
  //Αν δεν έχει επιλογή ή είναι επιλεγμένο το "Όχι" απενεργοποιεί το
πεδίο που συμπληρώνει
  //το λογαριασμό σύνδεσης
  if (cmbLink.ItemIndex =
cmbLink.Items.IndexOf(GetListValue('cmbLink','Oxi')) ) or
(cmbLink.ItemIndex = -1) then
  begin
    medCustomer.Enabled := false;
    medCustomer.Text := '';
  end
  else  //Αλλίώς του γεμίζει το πεδίο σύνδεσης με τα 5 τελευταία
νούμερα του λογαριασμού
    medCustomer.Enabled := true;
    if Length(medAccount.Account) = 17 then
    begin
      tmpAccountText := medAccount.Text;
      delete(tmpAccountText,1,12);
      medCustomer.Text := tmpAccountText;
    end;
end;

//Η συνάρτηση αυτή ενεργοποεί και απενεργοποιεί τα controls που
σχετίζονται με το αν δέχεται εγγραφές
//ή όχι
procedure TfrmAccounts.ControlsEnabled(state : bool);
begin
  rbNaiDE.Checked := state;
  rbOxiDE.Checked := not state;
  cmbAccount_Type.Enabled := state;
  cmbRecord_Check.Enabled := state;
  cmbProtection.Enabled := state;
  cmbLink.Enabled := state;
  medCurrency.Enabled := state;
  edPCurrency.Enabled := state;
  sbtnCurrency.Enabled := state;

  if state then
  begin

    if cmbAccount_Type.ItemIndex = -1 then cmbAccount_Type.ItemIndex
:= 0;
    if cmbRecord_Check.ItemIndex = -1 then cmbRecord_Check.ItemIndex
:= 3;
    if cmbProtection.ItemIndex = -1 then cmbProtection.ItemIndex :=
2;
    if cmbLink.ItemIndex = -1 then cmbLink.ItemIndex := 2;
  end
  else
  begin
    cmbAccount_Type.ItemIndex := -1;
    cmbRecord_Check.ItemIndex := -1;
    cmbProtection.ItemIndex := -1;
    cmbLink.ItemIndex := -1;
  end;

  if (cmbLink.ItemIndex = -1) or (cmbLink.ItemIndex = 2) then
  begin
    medCustomer.Text := '';
    medCustomer.Enabled := false;
```

```
  end
  else medCustomer.Enabled := true;

  if Length(medAccount.Account) = 3 then
  begin
     sedL_P_Print_Level.Enabled := true;
     sedBalance_Print_Level.Enabled := true;
     sedValues_Include_Tax.Enabled := true;
   end
   else
   begin
     sedL_P_Print_Level.Enabled := false;
     sedBalance_Print_Level.Enabled := false;
     sedValues_Include_Tax.Enabled := false;
   end;
end;

//Αυτό εκτελείται όταν γράφει στο πεδίο του τίτλου και γεμίζει το
ListView με τους λογαριασμούς που
//στο τίτλο τους έχουν πρόθεμα αυτό που γράφει ο χρήστης
procedure TfrmAccounts.edTitleKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if medAccount.Account = '' then
    if GetData(myQuery,'SELECT Account,Title FROM GLM WHERE Title
LIKE''' + edTitle.Text  + '%'' ORDER BY Title') then FillLV;
end;

//Το onExit event του πεδίου του τίτλου
procedure TfrmAccounts.edTitleExit(Sender: TObject);
begin
  inherited;
  //Αν ο λογαριασμός είναι κενός τότε ψάχνει να βρει λογαριασμό με το
τίτλο που έχει γράψει
  if (medAccount.Account = '') then
  begin
    if GetData(CurrRecQuery,'SELECT * FROM GLM WHERE Title=''' +
edTitle.Text + '''') then SearchFieldExit()
    else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

    tbtnInsert.Enabled := false;   //Σε αυτή τη περίπτωση δε μπορούμε
να κάνουμε εισαγωγή
  end
  //Αν ο λογαριασμός δεν είναι κενός (περίπτωση εισαγωγής) και το
"Όχι" είναι επιλεγμένο για το αν δέχεται
  //εγγραφές τότε απενεργοποιεί τα controls που σχετίζονται με το αν
δέχεται εγγραφές
  else if rbOxiDE.Checked then ControlsEnabled(false);
end;

//Η συνάρτηση αυτή καλείται στα OnExit events των πεδίων του τίτλου
και του λογαριασμού αφού έχει
//εκτελέστεί το ερώτημα στη βάση για να φέρει το λογαριασμό στο
CurrRecQuery (το ερώτημα θα έχει γίνει
//βάσει λογαριασμού αν έχει εκτελεστεί το onExit του λογαριασμού πριν
ενώ βάσει του τίτλου θα είναι
//αν κληθεί από το onExit του τίτλου
procedure TfrmAccounts.SearchFieldExit();
begin
```

```
  if CurrRecQuery.RecordCount <> 0 then
  begin
    //Αν ήρθε εγγραφή από το ερώτημα τότε γεμίζει τη φόρμα
    FillFormFromQuery;
    //Και φτιάχνει ανάλογα τα κουμπιά
    tbtnInsert.Enabled := false;
    tbtnAccept.Enabled := true;
    tbtnDelete.Enabled := true;
  end
  else
  begin
    //αν όχι αρχικοποιεί τη φόρμα
    if not tbtnInsert.Enabled then InitiallizeControls ;
    tbtnInsert.Enabled := true;
    tbtnAccept.Enabled := false;
    tbtnDelete.Enabled := false;
  end;
end;

//Αυτό το event του ListView εκτελείται όταν επιλέγεται κάποια
εγγραφή του ListView.
//Στη περίπτωση αυτή εμείς θέλουμε να γεμίζει τη φόρμα με τα στοιχεία
της ανάλογης εγγραφής
procedure TfrmAccounts.LVSelectItem(Sender: TObject; Item: TListItem;
  Selected: Boolean);
begin
  inherited;
  if selected then
    if GetData(CurrRecQuery,'SELECT * FROM GLM WHERE Account=''' +
Item.Caption + '''') then
    begin
      Changes:=false;
      FillFormFromQuery ;
      tbtnInsert.Enabled := false;
      tbtnAccept.Enabled := true;
      tbtnDelete.Enabled := true;
    end;
end;

//Το event onClick του κουμπιού για την εμφάνιση βοήθειας στο νόμισμα
procedure TfrmAccounts.sbtnCurrencyClick(Sender: TObject);
var
  Criteria:string;
begin
  inherited;

  //Φτιάχνει τα κριτήρια με βάσει το τι έχει γράψει ο χρήστης στο
πεδίο της περιγραφής
  criteria := edPCurrency.Text;
  if CheckCurrency then criteria := '';   //Τα βάζω ανάποδα γιατί η
CheckCurrency σβήνει τη περιγραφή
                                         //και δεν μπορώ να τη βάλω
μετά στο Like

  //Καλεί τη φόρμα των προβολών περνώντας σαν όρισμα και τα κριτήρια
  frmViews := TfrmViews.Create(Self);
  medCurrency.Text :=
frmViews.ShowViewForFields('frmAccounts','medCurrency',Criteria);

  //Ελέγχει για επιβεβαίωση το νόμισμα και στέλνει το focus στο πεδίο
του κωδικού του νομίσματος
```

```
  CheckCurrency ;
  medCurrency.SetFocus ;
end;

//Το event onKeyDown του editbox της ΠΕΡΙΓΡΑΦΗΣ του νομίσματος
procedure TfrmAccounts.edPCurrencyKeyDown(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
begin
  inherited;

  //Αν πατηθεί το enter και δεν είναι σωστά συμπληρωμένο ο κωδικός
του νομίσματος (medCurrency.Text)
  //τότε εμφάνισε τη βοήθεια (το sbtnCurrencyClick(Sender) καλεί τη
συνάρτηση που καλείται στο onClick του
  //κουμπιού sbtnCurrency το οποίο καλεί την εμφάνιση της βοήθειας.Αν
ο κωδικός έχει συμπληρωθεί τότε καλεί
  //την GlobalKeyDown που καλείται στο KeyDown όλων των controls της
φόρμας
  if (Key = 13) and (not
CheckIfExists(myQuery,'CURRENCY',medCurrency.Text,(Application.MainFo
rm.FindComponent('UserQuery') as
TQuery).fieldByName('LanguageID').AsString)) then
sbtnCurrencyClick(Sender)
  else GlobalKeyDown(Sender,Key,Shift);
end;

//Το event onshow της φόρμας
procedure TfrmAccounts.FormShow(Sender: TObject);
begin
  inherited;
  //Φερνουμε αρχικά όλους τους λογαριασμούς και γεμίζουμε το LV
  if GetData(myQuery,'SELECT Account,Title FROM GLM ORDER BY
Account') then FillLV
  else
Application.messageBox(pchar(GetError('DBError')),'',MB_ICONERROR);

  //Η δικαιοδοσία του χρήστη
  Level := (Application.MainForm.FindComponent('UserQuery') as
TQuery).fieldByName('AUTHORITYLEVEL').AsInteger;

  //Η συνάρτηση αυτή φέρνει από τη βάση τα captions των labels,των
κουμπιών και των στηλών του LV
  tbtnSeparator.Width := ToolBar.Width - 4*tbtnInsert.Width ;

  //Βάζουμε το μήνυμα που θέλουμε να εμφανίζει το medAccount όταν δεν
υπάρχει προηγούμενος λογαριασμός
  medAccount.NotExistParentAccountMessage := GetError('No_Par_Acc');

  //Ρυθμίζουμε τα κουμπιά
  tbtnInsert.Enabled := false;
  tbtnAccept.Enabled := false;
  tbtnDelete.Enabled := false;

  //Καθαρίζουμε τη φόρμα
  InitiallizeControls ;
end;

//Το event onclick του κουμιού της εισαγωγής
procedure TfrmAccounts.btnEisagwgiClick(Sender: TObject);
begin
```

```
  inherited;
  //Ενημερώνουμε της μεταβλητές από τη φόρμα
  UpdateVariablesFromForm;

  //Εκτελούμε την εισαγωγή με τα περιεχόμενα των μεταβλητών ή των
control όπου είναι εφικτό
  if not ExecQuery(myQuery,'INSERT INTO GLM
(Account,Title,Customer,Accepts_Records,Account_Type,Record_Check,Cre
ation_Date,Protection,Link,Currency,' +

'L_P_Print_Level,Balance_Print_Level,Values_Include_Tax,Notes_Level_1
,Notes_Level_2,Notes_Level_3,Notes_Level_4,Notes_Level_5,Notes_Level_
6,Notes_Level_7) VALUES('''
                    + medAccount.Account + ''','''
                    + edTitle.Text + ''','''
                    + medCustomer.Text + ''','''
                    + varAccepts_Records + ''','''
                    + varAccount_Type + ''','''
                    + varRecord_Check + ''',GetDate(),'''
                    + varProtection + ''','''
                    + varLink + ''','''
                    + medCurrency.Text + ''','
                    + varL_P_Print_Level + ','
                    + varBalance_Print_Level + ','
                    + varValues_Include_Tax + ','''
                    + varNotes[1] + ''','''
                    + varNotes[2] + ''','''
                    + varNotes[3] + ''','''
                    + varNotes[4] + ''','''
                    + varNotes[5] + ''','''
                    + varNotes[6] + ''','''
                    + varNotes[7] +''')') then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

  //Ρυθμίζουμε τα κουμπιά (ενεργοποίηση/απενεργοποίηση)
  tbtnInsert.Enabled := false;
  tbtnAccept.Enabled := true;
  tbtnDelete.Enabled := true;

  InitiallizeControls ;  //Καθαρισμός φόρμας
  medAccount.SetFocus ;
  medAccount.text := '';
end;

//Το event onclick του κουμιού της αποδοχής αλλαγών
procedure TfrmAccounts.btnApodoxiClick(Sender: TObject);
begin
  inherited;
  //Ενημερώνουμε της μεταβλητές από τη φόρμα
  UpdateVariablesFromForm;

  //Εκτελούμε την ενημέρωση με τα περιεχόμενα των μεταβλητών ή των
control όπου είναι εφικτό
  if not ExecQuery(myQuery,'UPDATE GLM SET Account='''
                    + medAccount.Account + ''',Title='''
                    + edTitle.Text  + ''',Customer='''
                    + medCustomer.Text  + ''',Accepts_Records='''
                    + varAccepts_Records + ''',Account_Type='''
                    + varAccount_Type + ''',Record_Check='''
                    + varRecord_Check + ''',Protection='''
                    + varProtection + ''',Currency='''
```

```
                    + medCurrency.Text + ''',Link='''
                    + varLink + ''',L_P_Print_Level='
                    + varL_P_Print_Level + ',Balance_Print_Level='
                    + varBalance_Print_Level + ',Values_Include_Tax='
                    + varValues_Include_Tax + ',Notes_Level_1='''
                    + varNotes[1] + ''',Notes_Level_2='''
                    + varNotes[2] + ''',Notes_Level_3='''
                    + varNotes[3] + ''',Notes_Level_4='''
                    + varNotes[4] + ''',Notes_Level_5='''
                    + varNotes[5] + ''',Notes_Level_6='''
                    + varNotes[6] + ''',Notes_Level_7='''
                    + varNotes[7] + ''' WHERE Account=''' +
medAccount.Account + '''') then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

  InitiallizeControls ;   //Καθαρίζουμε τη φόρμα
  medAccount.SetFocus ;
  medAccount.text := '';
end;

//Το event onclick του κουμιού της διαγραφής
procedure TfrmAccounts.btnDiagrafiClick(Sender: TObject);
begin
  inherited;

  //Φέρνουμε με μια LIKE όλους τους λογαριασμούς που ξεκινούν με το
string που έχει εισάγει ο χρήστης
  if GetData(myQuery,'SELECT Account FROM GLM WHERE Account LIKE '''
+ medAccount.Account + '%''') then
  begin
    //Αν φέρει μόνο έναν σημαίνει πως δεν υπάρχει άλλος λογαριασμός
που να ξεκινάει με αυτό το string
    //δηλαδή ο λογαριασμός δεν έχει υπολογαριασμούς
    if myQuery.RecordCount = 1 then
    begin
      //Στη περίπτωση αυτή σβήνουμε κανονικά το λοριασμό από τη βάση
      if ExecQuery(myQuery,'DELETE FROM GLM WHERE Account=''' +
medAccount.Account + '''') then
      begin
          InitiallizeControls ;  //καθαρίζουμε τα controls
          medAccount.SetFocus ;
          medAccount.text := '';
          if GetData(myQuery,'SELECT Account,Title FROM GLM') then
FillLV;  //Ξαναγεμίζουμε το LV
      end
      else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
    end
    else
Application.messageBox(PChar(GetError('DELETE_GLM')),'',MB_ICONERROR)
;
  end
  else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
end;

//Το event onclick του κουμιού της Εξόδου
procedure TfrmAccounts.btnExodosClick(Sender: TObject);
begin
  inherited;
  Close;
```

```
end;

//Το onclick του κουμπιού που εμφανίζει τη φόραμ των σημειώσεων
procedure TfrmAccounts.sbtnNotesClick(Sender: TObject);
var i:integer;
begin
  inherited;
  frmNotes := TfrmNotes.Create(Self);
  if frmNotes.ShowAllNotes(varNotes)=mrok then
    for i:=1 to 7 do varNotes[i]:=frmNotes.Notes[i];
  FreeAndNil(frmNotes);

  memNotes.Text := VarNotes[level];
end;

//Απλά στέλνει τι κέρσορα στην αρχή κατά την εισαγωγή στο Memo των
σημειώσεων
procedure TfrmAccounts.memNotesEnter(Sender: TObject);
begin
  inherited;
  Enter(Sender);
  memNotes.SelStart:=length(memNotes.Text);
  memNotes.SelLength := 0;
end;

//Το κουμπί της εισαγωγής
procedure TfrmAccounts.tbtnInsertClick(Sender: TObject);
begin
  inherited;
  if
Application.MessageBox(PChar(GetError('Insert_Record')),'',MB_YESNO+M
B_ICONWARNING)=IDYES then
  begin
    //Ενημερώνουμε της μεταβλητές από τη φόρμα
    UpdateVariablesFromForm;

    //Εκτελούμε την εισαγωγή με τα περιεχόμενα των μεταβλητών ή των
control όπου είναι εφικτό
    if not ExecQuery(myQuery,'INSERT INTO GLM
(Account,Title,Customer,Accepts_Records,Account_Type,Record_Check,Cre
ation_Date,Protection,Link,Currency,' +

'L_P_Print_Level,Balance_Print_Level,Values_Include_Tax,Notes_Level_1
,Notes_Level_2,Notes_Level_3,Notes_Level_4,Notes_Level_5,Notes_Level_
6,Notes_Level_7) VALUES('''
                    + medAccount.Account + ''','''
                    + edTitle.Text + ''','''
                    + medCustomer.Text + ''','''
                    + varAccepts_Records + ''','''
                    + varAccount_Type + ''','''
                    + varRecord_Check + ''',GetDate(),'''
                    + varProtection + ''','''
                    + varLink + ''','''
                    + medCurrency.Text + ''','
                    + varL_P_Print_Level + ','
                    + varBalance_Print_Level + ','
                    + varValues_Include_Tax + ','''
                    + varNotes[1] + ''','''
                    + varNotes[2] + ''','''
                    + varNotes[3] + ''','''
                    + varNotes[4] + ''','''
```

```
                    + varNotes[5] + ''','''
                    + varNotes[6] + ''','''
                    + varNotes[7] +''')') then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

    //Ρυθμίζουμε τα κουμπιά (ενεργοποίηση/απενεργοποίηση)
    tbtnInsert.Enabled := false;
    tbtnAccept.Enabled := true;
    tbtnDelete.Enabled := true;

    if GetData(myQuery,'SELECT Account,Title FROM GLM ORDER BY
Account') then FillLV
    else
Application.messageBox(pchar(GetError('DBError')),'',MB_ICONERROR);
    InitiallizeControls ;  //Καθαρισμός φόρμας
    medAccount.SetFocus ;
    medAccount.text := '';
  end;
end;

//Το κουμπί της αποδοχής
procedure TfrmAccounts.tbtnAcceptClick(Sender: TObject);
begin
  inherited;
  if
Application.MessageBox(PChar(GetError('Modify_Record')),'',MB_YESNO+M
B_ICONWARNING)=IDYES then
  begin
    //Ενημερώνουμε της μεταβλητές από τη φόρμα
    UpdateVariablesFromForm;

    //Εκτελούμε την ενημέρωση με τα περιεχόμενα των μεταβλητών ή των
control όπου είναι εφικτό
    if not ExecQuery(myQuery,'UPDATE GLM SET Account='''
                    + medAccount.Account + ''',Title='''
                    + edTitle.Text  + ''',Customer='''
                    + medCustomer.Text  + ''',Accepts_Records='''
                    + varAccepts_Records + ''',Account_Type='''
                    + varAccount_Type + ''',Record_Check='''
                    + varRecord_Check + ''',Protection='''
                    + varProtection + ''',Currency='''
                    + medCurrency.Text + ''',Link='''
                    + varLink + ''',L_P_Print_Level='
                    + varL_P_Print_Level + ',Balance_Print_Level='
                    + varBalance_Print_Level +
',Values_Include_Tax='
                    + varValues_Include_Tax + ',Notes_Level_1='''
                    + varNotes[1] + ''',Notes_Level_2='''
                    + varNotes[2] + ''',Notes_Level_3='''
                    + varNotes[3] + ''',Notes_Level_4='''
                    + varNotes[4] + ''',Notes_Level_5='''
                    + varNotes[5] + ''',Notes_Level_6='''
                    + varNotes[6] + ''',Notes_Level_7='''
                    + varNotes[7] + ''' WHERE Account=''' +
medAccount.Account + '''') then
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

    if GetData(myQuery,'SELECT Account,Title FROM GLM ORDER BY
Account') then FillLV
    else
Application.messageBox(pchar(GetError('DBError')),'',MB_ICONERROR);
```

```
    InitiallizeControls ;   //Καθαρίζουμε τη φόρμα
    medAccount.SetFocus ;
    medAccount.text := '';
  end;
end;

//Το κουμπί της διαγραφής
procedure TfrmAccounts.tbtnDeleteClick(Sender: TObject);
begin
  inherited;

  if
Application.MessageBox(PChar(GetError('Delete_Record')),'',MB_YESNO+M
B_ICONWARNING)=IDYES then
  begin
    //Φέρνουμε με μια LIKE όλους τους λογαριασμούς που ξεκινούν με το
string που έχει εισάγει ο χρήστης
    if GetData(myQuery,'SELECT Account FROM GLM WHERE Account LIKE
''' + medAccount.Account + '%''') then
    begin
      //Αν φέρει μόνο έναν σημαίνει πως δεν υπάρχει άλλος λογαριασμός
που να ξεκινάει με αυτό το string
      //δηλαδή ο λογαριασμός δεν έχει υπολογαριασμούς
      if myQuery.RecordCount = 1 then
      begin
        //Στη περίπτωση αυτή σβήνουμε κανονικά το λοριασμό από τη
βάση
        if ExecQuery(myQuery,'DELETE FROM GLM WHERE Account=''' +
medAccount.Account + '''') then
        begin
            InitiallizeControls ;  //καθαρίζουμε τα controls
            medAccount.SetFocus ;
            medAccount.text := '';
            if GetData(myQuery,'SELECT Account,Title FROM GLM') then
FillLV;  //Ξαναγεμίζουμε το LV
        end
        else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
      end
      else
Application.messageBox(PChar(GetError('DELETE_GLM')),'',MB_ICONERROR)
;
    end
    else
Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);

    if GetData(myQuery,'SELECT Account,Title FROM GLM ORDER BY
Account') then FillLV
    else
Application.messageBox(pchar(GetError('DBError')),'',MB_ICONERROR);
  end;
end;

//Το κουμπί exit
procedure TfrmAccounts.tbtnExitClick(Sender: TObject);
begin
  inherited;
  Close;
end;
```

```
//Το onKeyDown του ListView που χρησιμοποιείται για να σβήνει μια
εγγραφή όταν πατιέται το delete
procedure TfrmAccounts.LVKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  inherited;
  GlobalKeyDown(Sender,key,shift);
  if key=VK_DELETE then
    if tbtnDelete.Enabled = true then
       tbtnDeleteClick(Sender);
end;

//To onEnter του control του λογαριασμού
procedure TfrmAccounts.medAccountEnter(Sender: TObject);
begin
  inherited;
  Enter(Sender);
  if tbtnInsert.Enabled = true then tbtnInsertClick(Sender)
  else if (tbtnAccept.Enabled = true) and (Changes = true) then
tbtnAcceptClick(Sender);

  prev_account := medAccount.Account ;
  Changes := true;
end;

end.
```

4.2 Η διεπαφή των κινήσεων (TfrmAccount_Records)

```
unit AccountRecords;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  TemplateForm, Db, ComCtrls, ExtCtrls, StdCtrls, Mask, Buttons,
  Grids, DBGrids, OleCtrls, DBTables, Spin, LimitSpinEdit,
AccountMaskEdit,FormViews;

type
  TfrmAccountRecords = class(TfrmTemplate)
    dtmGLK_Date: TDateTimePicker;
    medVoucher: TMaskEdit;
    sbtnAction: TSpeedButton;
    sbtnReason: TSpeedButton;
    edPAction: TEdit;
    edPReason: TEdit;
    lbDate: TLabel;
    lbVoucher: TLabel;
    lbAction: TLabel;
    lbReason: TLabel;
    medReason: TMaskEdit;
    medAction: TMaskEdit;
    myPanel: TPanel;
    LV: TListView;
    medAccount: TAccountMaskEdit;
    edAccount_Title: TEdit;
    medActivity: TMaskEdit;
```

```
    edReason: TEdit;
    sedDebit: TLimitSpinEdit;
    sedCredit: TLimitSpinEdit;
    sbtnActivity: TSpeedButton;
    edPActivity: TEdit;
    lbActivity: TLabel;
    lbAccount: TLabel;
    lbDebit: TLabel;
    lbCredit: TLabel;
    lbReason2: TLabel;
    lbAccount_Title: TLabel;
    lbAA: TLabel;
    sbtnAccount: TSpeedButton;
    sedGLK_index: TLimitSpinEdit;
    sumLV: TListView;
    procedure Enter(Sender: TObject);
    procedure GlobalKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure medReasonExit(Sender: TObject);
    procedure edReasonEnter(Sender: TObject);
    procedure myPanelEnter(Sender: TObject);
    procedure LVEnter(Sender: TObject);
    procedure LVSelectItem(Sender: TObject; Item: TListItem;
Selected: Boolean);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure LVKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure LVExit(Sender: TObject);
    procedure sbtnActionClick(Sender: TObject);
    procedure sbtnActivityClick(Sender: TObject);
    procedure sbtnReasonClick(Sender: TObject);
    procedure sbtnAccountClick(Sender: TObject);
    procedure edPActionKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure edPActivityKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
    procedure edAccount_TitleKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
    procedure edPReasonKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure dtmGLK_DateEnter(Sender: TObject);
  private
    { Private declarations }
    GLMCurrency:string;
    function CheckAction():bool;
    function CheckAccount():bool;
    function CheckActivity():bool;
    function CheckReason():bool;
    function InsertRec():bool;

    procedure AddItemToLV();
    procedure LVSumLine();
    procedure ClearPanelControls();
    procedure UpdateLVItem();
    procedure FillForm();

  public
    { Public declarations }
  end;

var
```

```
  frmAccountRecords: TfrmAccountRecords;

implementation

{$R *.DFM}

//Ελέγχει αν ο κωδικός που έχει εισάγει ο χρήστης στο Action πεδίο
είναι υπαρκτός
function TfrmAccountRecords.CheckAction():bool;
begin
  Result :=
CheckIfExists(myQuery,'ACTIONS',medAction.Text,(Application.MainForm.
FindComponent('UserQuery') as
TQuery).fieldByName('LanguageID').AsString);
  if Result then edpAction.Text :=
myQuery.fieldByName('Description').AsString
  else //edpAction.Text := '';
end;

//Ομοίως και για το λογαριασμό απλά εδώ πρέπει να ελέγξει επιπλέον
και αν ο λογαριασμός
//δέχεται εγγραφές
function TfrmAccountRecords.CheckAccount():bool;
begin
  Result := false;
  if GetData(myQuery,'SELECT Account,Title,Currency FROM GLM WHERE
Account=''' + medAccount.Account + ''' AND Accepts_Records=''N'' ')
then
    if myQuery.RecordCount > 0 then
    begin
      Result := true;
      if edAccount_Title.Text = '' then edAccount_Title.Text :=
myQuery.fieldByName('Title').AsString;
      GLMCurrency := myQuery.fieldByName('Currency').AsString;
    end
    else //edAccount_Title.Text := '';
end;

//Ομοίως για το Activity
function TfrmAccountRecords.CheckActivity():bool;
begin
  Result :=
CheckIfExists(myQuery,'ACTIVITIES',medActivity.Text,(Application.Main
Form.FindComponent('UserQuery') as
TQuery).fieldByName('LanguageID').AsString);
  if Result then edPActivity.Text :=
myQuery.fieldByName('Description').AsString
  else //edPActivity.Text := '';
end;

//Ομοίως για το Reason
function TfrmAccountRecords.CheckReason():bool;
begin
  Result :=
CheckIfExists(myQuery,'REASONS',medReason.Text,(Application.MainForm.
FindComponent('UserQuery') as
TQuery).fieldByName('LanguageID').AsString);
  if Result then edPReason.Text :=
myQuery.fieldByName('Description').AsString
  else //edPReason.Text := '';
end;
```

```
//Το onenter event όλων των controls της φόρμας.Εκτελείται όταν το
focus πηγαίνει
//σε οποιοδήποτε control και ελέγχει αν τα προηγούμενά του έχουν
συμπληρωθεί σωστά.
//Αν όχι τον πηγαίνει εκεί που έχει γίνει το λάθος εκτός αν το πεδιό
στο οπίο προσπαθεί
//να μπει είναι το πεδίο περιγραφής του κωδικού στον οποίο υπάρχει το
λάθος
procedure TfrmAccountRecords.Enter(Sender: TObject);
begin
  inherited;

  if (not checkAction) then
  begin
    if (Sender as TWinControl).name <> 'edpAction' then
    begin
      if (Sender as TWinControl).parent = myPanel then
medAction.SetFocus
      else if medAction.TabOrder < (Sender as TWinControl).TabOrder
then medAction.SetFocus;
    end;
    medAction.Text := '';
  end
  else if (not checkAccount) then
  begin
    if (Sender as TWinControl).name <> 'edAccount_Title' then
    begin
      if (Sender as TWinControl).parent = myPanel then
      begin
        if medAccount.TabOrder < (Sender as TWinControl).TabOrder
then medAccount.SetFocus
        else if (medAccount.TabOrder + myPanel.TabOrder) < (Sender as
TWinControl).TabOrder then medAccount.SetFocus;
      end;
    end;
    medAccount.Text := '';
  end
  else if (not checkActivity) then
  begin
    if (Sender as TWinControl).name <> 'edPActivity' then
    begin
      if (Sender as TWinControl).parent = myPanel then
      begin
        if medActivity.TabOrder < (Sender as TWinControl).TabOrder
then medActivity.SetFocus
        else if (medActivity.TabOrder + myPanel.TabOrder) < (Sender
as TWinControl).TabOrder then medActivity.SetFocus;
      end;
    end;
    medActivity.Text := '';
  end
  else
  begin
    if sedDebit.CurrencyValue <> 0 then sedCredit.readonly := true
    else if sedCredit.CurrencyValue <> 0 then sedDebit.readonly :=
true
    else
    begin
      sedDebit.readonly := false;
      sedCredit.readonly := false;
```

```
    end;

    //Επίσης ενημερώνεται κάθε φορά και το LV για τις αλλαγές στην
εγγραφή.Αν η εγγραφή δεν έχει
    //μπει στο LV τότε ελέγχει αν έχει τελειώσει την εισαγωγή και αν
ναι τη προσθέτει
    if LV.FindCaption(0,sedGLK_index.Text,false,true,false) = nil
then
    begin
      if (sedDebit.CurrencyValue <> 0) or (sedCredit.CurrencyValue <>
0) then AddItemToLV;
    end
    else UpdateLVItem ;
  end;
end;

//Το κοινό onKeyDown event
procedure TfrmAccountRecords.GlobalKeyDown(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
begin
  if key = 13 then  SwitchControls(Sender)
  else if key = 27 then
  begin
    if ((Sender as TWinControl).parent = myPanel) and ((Sender as
TWinControl) <> sedGLK_index) then
    begin
      ClearPanelControls;
      sedGLK_index.SetFocus ;
    end
    else if (Sender as TWinControl) <> dtmGLK_Date then
    begin
      dtmGLK_Date.SetFocus ;
    end
    else Close;
  end;
end;

//Η συνάρτηση που εισάγει την εγγραφή στη βάση και ενημερώνει και το
πεδίο του GLM που
//δείχνει πότε έγινε η τελευταία κίνηση
function TfrmAccountRecords.InsertRec():bool;
begin
  Result := true;
  if not ExecQuery(myQuery,'INSERT INTO GLK
(Branch,Voucher,GLK_Action,Account,Account_Title,Reason,Activity,GLK_
index,GLK_Date,Debit,Credit,GLK_User,Currency,Record_Date) VALUES('''
                    +
(Application.MainForm.FindComponent('UserQuery') as
TQuery).fieldByName('BRANCHID').AsString + ''','''
                    + medVoucher.Text + ''','''
                    + medAction.Text + ''','''
                    + medAccount.Account + ''','''
                    + edAccount_Title.Text + ''','''
                    + edReason.Text + ''','''
                    + medActivity.Text + ''','''
                    + sedGLK_index.Text + ''','''
                    + FormatDateTime('yyyy-m-d',dtmGLK_Date.DateTime)
+ ''','
                    + sedDebit.GetStringForSQL  + ','
                    + sedCredit.GetStringForSQL  + ','
```

```
                    +
(Application.MainForm.FindComponent('UserQuery') as
TQuery).fieldByName('USERID').AsString + ','''
                    + GLMCurrency  + ''',GetDate())') then
    begin

Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
      Result := false;
    end
    else
    begin
      if not ExecQuery(myQuery,'UPDATE GLM SET Last_Record_Date=''' +
FormatDateTime('yyyy-m-d hh:mm',dtmGLK_Date.DateTime) +
                        ''' WHERE Account = ''' + medAccount.Account +
''''') then
      begin

Application.messageBox(PChar(GetError('DBError')),'',MB_ICONERROR);
        Result := false;
      end;
    end;
end;

//Το onexit του Reason
procedure TfrmAccountRecords.medReasonExit(Sender: TObject);
begin
  inherited;
  if
CheckIfExists(myQuery,'REASONS',medReason.Text,(Application.MainForm.
FindComponent('UserQuery') as
TQuery).fieldByName('LanguageID').AsString)then

    edPReason.Text := myQuery.fieldByName('Description').AsString
  else edPReason.Text := '';
end;

//Το onenter του Reason.Κάποια πεδία δεν είναι συνδεδεμένα απευθείας
με το κοινό event
//αλλά υλοποιούν δικό του event και καλούν μέσα από αυτό και το κοινό
(προφανώς γιατί
//θα πρέπει να γίνουν στο event αυτό και άλλα πράγματα που δεν
χρειάζονται στο γενικό)
procedure TfrmAccountRecords.edReasonEnter(Sender: TObject);
begin
  inherited;
  Enter(Sender);
  //Ελέγχει αν είναι πρώτη φορά που εισάγεται η εγγραφή στο LV ή αν
υπάρχει ήδη.
  //Αν είναι η πρώτη φορά βάζει στο δεύτερο πεδίο αιτιολογίας ότι
έχει και το πρώτο
  //ενώ αν υπάρχει ήδη η εγγραφή βάζει ότι έχει το LV στο αντίστοιχο
πεδίο
  if (checkAction) and (checkAccount) and (checkActivity) then
  begin
    if LV.FindCaption(0,sedGLK_index.Text,false,true,false) = nil
then edReason.Text := edPReason.Text
    else edReason.Text :=
LV.FindCaption(0,sedGLK_index.Text,false,true,false).SubItems[3];
  end;
end;
```

```
//Το onenter του panel
procedure TfrmAccountRecords.myPanelEnter(Sender: TObject);
begin
  inherited;

  if sedGLK_index.Text = '' then sedGLK_index.Text := '1';
  if LV.Items.Count = 0 then LVSumLine;
end;

//Προσθέτει την εγγραφή στο LV
procedure TfrmAccountRecords.AddItemToLV();
var
  tmpItem : TListItem;
begin
  LV.Items[LV.Items.Count - 1].Delete ;

  tmpItem := LV.Items.Add ;
  tmpItem.Caption := sedGLK_index.Text  ;
  tmpItem.SubItems.Add(medAccount.Account);
  tmpItem.SubItems.Add(edAccount_Title.Text);
  tmpItem.SubItems.Add(medActivity.Text + ' - ' + edPActivity.Text);
  tmpItem.SubItems.Add(edReason.Text);
  tmpItem.SubItems.Add(CurrToStr(sedDebit.CurrencyValue));
  tmpItem.SubItems.Add(CurrToStr(sedCredit.CurrencyValue));

  LVSumLine;
end;

//Ενημερώνει το άθροισμα
procedure TfrmAccountRecords.LVSumLine();
var
  i:integer;
  sumDebit,sumCredit :Currency;
begin

  sumDebit := 0;
  sumCredit := 0;
  if LV.Items.Count > 0 then
    for i:=0 to LV.Items.Count - 1 do
    begin
      sumDebit := sumDebit + StrToCurr(LV.Items[i].SubItems[4]);
      sumCredit := sumCredit + StrToCurr(LV.Items[i].SubItems[5]);
    end;

  sumLV.Columns[2].Caption := CurrToStr(sumDebit);
  sumLV.Columns[3].Caption := CurrToStr(sumCredit);

  LV.Items.Add ;
end;

//Καθαρίζει τα controls του panel
procedure TfrmAccountRecords.ClearPanelControls();
begin
  //Τον αύξοντα αριθμό δεν τον καθαρίζει αλλά τον προχωράει κατά ένα
  if LV.Items.Count > 1 then  sedGLK_index.text :=
IntToStr(StrToInt(LV.Items[LV.Items.Count -2].Caption) + 1)
  else sedGLK_index.text := '1';
  medAccount.text := '';
  edAccount_Title.text := '';
  medActivity.text := '';
  edPActivity.text := '';
```

```
  edReason.text := '';
  sedDebit.readonly := false;
  sedCredit.readonly := false;
  sedDebit.CurrencyValue := 0;
  sedCredit.CurrencyValue := 0;
end;

//Το onenter του LV
procedure TfrmAccountRecords.LVEnter(Sender: TObject);
begin
  inherited;
  Enter(Sender);
  LV.Selected := nil;
  LV.Items[LV.Items.count -1].Focused := true;
  ClearPanelControls;
end;

//To event onSelectItem του LV.Στη περίπτωση αυτή γεμίζει τη φόρμα με
την επιλεγμένη
//εγγραφή
procedure TfrmAccountRecords.LVSelectItem(Sender: TObject; Item:
TListItem;
  Selected: Boolean);
begin
  inherited;
  if Selected then
  begin
    ClearPanelControls;
    if Item <> LV.items[LV.Items.Count - 1] then FillForm;
  end;
end;

//Ενημερώνει το selected item του LV με τις αλλαγές που έχει κάνει ο
χρήστης
procedure TfrmAccountRecords.UpdateLVItem();
var
  tmpItem:TListItem;
begin
  LV.Items[LV.Items.Count -1].Delete ;

  tmpItem := LV.FindCaption(0,sedGLK_index.Text,false,true,false) ;
  tmpItem.Caption := sedGLK_index.Text  ;
  tmpItem.SubItems[0] := medAccount.Account;
  tmpItem.SubItems[1] := edAccount_Title.Text;
  tmpItem.SubItems[2] := medActivity.Text + ' - ' + edPActivity.Text;
  tmpItem.SubItems[3]:= edReason.Text;
  tmpItem.SubItems[4]:= CurrToStr(sedDebit.CurrencyValue);
  tmpItem.SubItems[5] := CurrToStr(sedCredit.CurrencyValue);

  LVSumLine;
end;

//Γεμίζει τη φόρμα με τα στοιχεία του LV
procedure TfrmAccountRecords.FillForm();
var
  tmpActivityStr :string;
begin
  sedGLK_index.Text := LV.Selected.Caption ;
  medAccount.Text := LV.Selected.SubItems[0];
  edAccount_Title.Text := LV.Selected.SubItems[1];
```

```
  tmpActivityStr := LV.Selected.SubItems[2];
  delete(tmpActivityStr,4,Length(tmpActivityStr)-1);
  medActivity.Text := tmpActivityStr;

  edReason.Text := LV.Selected.SubItems[3];
  sedDebit.Text := LV.Selected.SubItems[4];
  sedCredit.Text := LV.Selected.SubItems[5];

  CheckAccount;
  CheckActivity;
end;

//Το event αυτό εκτελείται όταν προσπαθεί ο χρήστης να κλείσει τη
φόρμα.Εμείς ελέγχουμε
//αν τα ποσά πίστωσης και χρέωσης συμφωνούν και αν ναι τότε μόνο του
επιτρέπουμε να φύγει από
//τη φόρμα (μέσω του CanClose).Επίσης ζητάμε επιβεβαίωση για την
αποθήκευση των αλλαγών
procedure TfrmAccountRecords.FormCloseQuery(Sender: TObject; var
CanClose: Boolean);
var i:integer;
begin
  inherited;
  //Ελέγχει τα ποσά
  if LV.Items.Count > 1 then
    if (StrToCurr(sumLV.columns[2].Caption) <>
StrToCurr(sumLV.columns[3].Caption)) then
    begin

Application.messageBox(PChar(GetError('Amounts_Disagree')),'',MB_ICON
ERROR);
      CanClose := false;

    end
    else
    begin
      //Ρωτάει για την αποθήκευση
      if LV.Items.Count > 1 then
      begin
        if
Application.MessageBox(PChar(GetError('Save')),'',MB_YESNO+MB_ICONWAR
NING)=IDYES then
          for i:=0 to LV.Items.Count - 2 do
          begin
            LV.Selected := LV.Items[i];
            InsertRec;
          end;
      end;
    end;
end;

//Το onKeyDown του LV.Ουσιαστικά σβήνει την επιλεγμένη εγγραφή από το
LV όταν πατηθεί
//το delete
procedure TfrmAccountRecords.LVKeyDown(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
var
  SelIdx : integer;
begin
  inherited;
```

```
  GlobalKeyDown(Sender,Key,Shift);
  if Key = VK_DELETE then
    if (LV.Selected <> nil) then
      if LV.Selected <> LV.Items[LV.Items.Count - 1] then
      begin
        SelIdx := LV.Selected.Index;
        LV.Selected.Delete ;
        if LV.Items.Count > 1 then
        begin
          if SelIdx > 0 then
          begin
            LV.Selected := LV.Items[SelIdx - 1];
            LV.Items[SelIdx - 1].Focused := true;
          end
          else
          begin
            LV.Selected := LV.Items[0];
            LV.Items[0].Focused := true;
          end;
        end
        else ClearPanelControls ;
      end;
end;

//Το onexit του LV.Στέλνει το focus στον αύξοντα αριθμό
procedure TfrmAccountRecords.LVExit(Sender: TObject);
begin
  inherited;
  sedGLK_index.SetFocus;
end;



//Το onClick του κουμπιού για την εμφάνιση της βοήθειας στο Action
procedure TfrmAccountRecords.sbtnActionClick(Sender: TObject);
var
  Criteria:string;
begin
  inherited;

  //Αν ο χρήστης έχει γράψει κάτι στο πεδίο περιγραφής τότε το
περνάει σαν κριτήριο
  criteria := '';
  if not CheckAction then criteria := edpAction.Text;

  //και καλεί τη φόρμα των προβολών
  frmViews := TfrmViews.Create(Self);
  medAction.Text :=
frmViews.ShowViewForFields('frmAccountRecords','medAction',Criteria);

  if CheckAction then medVoucher.SetFocus ;
end;

//Ομοίως και για το Activity
procedure TfrmAccountRecords.sbtnActivityClick(Sender: TObject);
var
  Criteria:string;
begin
  inherited;

  criteria := '';
```

```
  if not CheckActivity then criteria := edPActivity.Text;

  frmViews := TfrmViews.Create(Self);
  medActivity.Text :=
frmViews.ShowViewForFields('frmAccountRecords','medActivity',Criteria
);

  if CheckActivity then edReason.SetFocus ;
end;

//Ομοίως και για το Reason
procedure TfrmAccountRecords.sbtnReasonClick(Sender: TObject);
var
  Criteria:string;
begin
  inherited;

  criteria := '';
  if not CheckReason then criteria := edPReason.Text;

  frmViews := TfrmViews.Create(Self);
  medReason.Text :=
frmViews.ShowViewForFields('frmAccountRecords','medReason',Criteria);

  CheckReason ;
  sedGLK_index.SetFocus ;

end;

//Ομοίως και για το Account
procedure TfrmAccountRecords.sbtnAccountClick(Sender: TObject);
var
  Criteria:string;
begin
  inherited;

  criteria := '';
  if not CheckAccount then criteria := edAccount_Title.Text;

  frmViews := TfrmViews.Create(Self);
  medAccount.Text :=
frmViews.ShowViewForFields('frmAccountRecords','medAccount',Criteria)
;

  edAccount_Title.Text := '';
  CheckAccount ;
  edAccount_Title.SetFocus ;
end;

//Το onKeyDownEvent του πεδίου της περιγραφής του Action.Αν πατηθεί
το enter
//και δεν είναι σωστά συμπληρωμένο το πεδίο του κωδικού τότε καλέι το
onClick
//του κουμπιού δηλαδή ουσιαστικά καλεί τη φόρμα των προβολών
procedure TfrmAccountRecords.edPActionKeyDown(Sender: TObject; var
Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if (Key = 13) and (not CheckAction) then sbtnActionClick(Sender)
  else GlobalKeyDown(Sender,Key,Shift);
```

```
end;

//Ομοίως και για το Activity
procedure TfrmAccountRecords.edPActivityKeyDown(Sender: TObject; var
Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if (Key = 13) and (not CheckActivity) then
sbtnActivityClick(Sender)
  else GlobalKeyDown(Sender,Key,Shift);
end;

//Ομοίως και για το Account
procedure TfrmAccountRecords.edAccount_TitleKeyDown(Sender: TObject;
var Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if (Key = 13) and (not CheckAccount) then sbtnAccountClick(Sender)
  else GlobalKeyDown(Sender,Key,Shift);
end;

//Ομοίως και για το Reason
procedure TfrmAccountRecords.edPReasonKeyDown(Sender: TObject; var
Key: Word;
  Shift: TShiftState);
begin
  inherited;
  if (Key = 13) and (not CheckReason) then sbtnReasonClick(Sender)
  else GlobalKeyDown(Sender,Key,Shift);
end;

procedure TfrmAccountRecords.dtmGLK_DateEnter(Sender: TObject);
begin
  inherited;
  dtmGLK_Date.Date := Date;
  Enter(Sender);
end;

end.
```

4.3 Η διεπαφή προβολής του λογιστικού σχεδίου (TfrmLP)

```
unit LPForm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, TemplateForm, StdCtrls, Buttons, ExtCtrls, Spin,
LimitSpinEdit,
  Mask, AccountMaskEdit, DB, DBTables;

type
  TfrmLP = class(TfrmTemplate)
    lbFromAccount: TLabel;
    lbToAccount: TLabel;
```

```
    lbLevel: TLabel;
    lbLowerLevel: TLabel;
    medApoLog: TAccountMaskEdit;
    medEwsLog: TAccountMaskEdit;
    sedBathmou: TLimitSpinEdit;
    pnlMikrBathmou: TPanel;
    rbNaiMB: TRadioButton;
    rbOxiMB: TRadioButton;
    btnShow: TBitBtn;
    procedure btnShowClick(Sender: TObject);
    procedure GlobalKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
  private
    { Private declarations }
    Criteria:string;
  public
    { Public declarations }
  end;

var
  frmLP: TfrmLP;

implementation

uses formViews;

{$R *.dfm}

//Το keydown event όπου συνδέονται όλα τα keydown events της φόρμας
procedure TfrmLP.GlobalKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  //Αν πατηθεί το enter πήγαινε στο απόμενο control
  if key = 13 then SwitchControls(Sender) ;
  if key = 27 then
  begin
    if medApoLog.Focused then Close
    else
    begin
      medApoLog.SetFocus ;
      medApoLog.text := '';
      medEwsLog.text := '';
      sedBathmou.Value := 0;
      rbNaiMB.Checked := true;
    end;
  end;
end;

//Το onclick event της εμφάνισης
procedure TfrmLP.btnShowClick(Sender: TObject);
var flag : boolean;
    BathmosLength:string;
begin
  inherited;
  flag := false;
  Criteria := '';

  //Αν το πεδίο λογαριασμού από τον οποίο θέλουμε εκτύπωση είναι
συμπληρωμένο τότε
  //θέτουμε σαν κριτήριο οτι ο λογαριασμός πρέπει να είναι
μεγαλύτερος από αυτόν που
```

```
  //έχει συμπληρωθεί
  if medApoLog.Account <> '' then
  begin
    Criteria := Criteria + 'Account >= ''' + medApoLog.Account + '''
';
    flag := true;
  end;

  //Το αντίστοιχο με πριν αλλά για το άλλο πεδίο λογαριασμού
  if medEwsLog.Account <> '' then
  begin
    //Το flag χρειάζεται για να ξέρουμε αν θα βάλουμε 'and' ή όχι
    if flag then Criteria := Criteria + 'AND ';
    Criteria := Criteria + 'Account <= ''' + medEwsLog.Account + '''
';
    flag := true;
  end;

  //Εδώ φτιάχνουμε το κριτήριο για το βαθμό του λογαριασμού
(ουσιαστικά το μήκος του)
  if sedBathmou.Value > 0 then
  begin
    case sedBathmou.Value of
      1: BathmosLength := '3';
      2: BathmosLength := '6';
      3: BathmosLength := '9';
      4: BathmosLength := '12';
      5: BathmosLength := '17';
    end;

    if flag then Criteria := Criteria + 'AND ';
    if rbNaiMB.Checked then Criteria := Criteria + 'LEN(Account) <=
''' + BathmosLength + ''' '
    else Criteria := Criteria + 'LEN(Account) = ''' + BathmosLength +
''' ';
  end;

  //Τέλος καλούμε τη φόρμα των προβολών δίνοντας σαν παράμετρο και τα
επιπλέον κριτήρια
  //που έχουμε φτιάξει
  frmViews := TfrmViews.Create(Self);
  frmViews.ShowView('frmLP',Criteria);
end;

end.
```

4.4 η φόρμα διαχείρισης των σημειώσεων (TfrmNotes)

```
unit NotesForm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, TemplateForm, StdCtrls, Buttons, DB, DBTables;

type
```

```
  TfrmNotes = class(TfrmTemplate)
    memNotes1: TMemo;
    lbLevel1: TLabel;
    memNotes2: TMemo;
    lbLevel2: TLabel;
    memNotes3: TMemo;
    lbLevel3: TLabel;
    memNotes4: TMemo;
    lbLevel4: TLabel;
    memNotes5: TMemo;
    lbLevel5: TLabel;
    memNotes6: TMemo;
    lbLevel6: TLabel;
    memNotes7: TMemo;
    lbLevel7: TLabel;
    btnOk: TBitBtn;
    btnCancel: TBitBtn;
    procedure btnOkClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure GlobalKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure GlobalEnter(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    Notes: array[1..7] of string;
    function ShowAllNotes(inNotes:array of string):integer;
  end;

var
  frmNotes: TfrmNotes;

implementation

{$R *.dfm}

procedure TfrmNotes.btnOkClick(Sender: TObject);
begin
  inherited;
  Notes[1] := memNotes1.Text ;
  Notes[2] := memNotes2.Text ;
  Notes[3] := memNotes3.Text ;
  Notes[4] := memNotes4.Text ;
  Notes[5] := memNotes5.Text ;
  Notes[6] := memNotes6.Text ;
  Notes[7] := memNotes7.Text ;
end;

procedure TfrmNotes.FormShow(Sender: TObject);
var i,level:integer;
begin
  inherited;
  level := (Application.MainForm.FindComponent('UserQuery') as
TQuery).fieldByName('AUTHORITYLEVEL').AsInteger;
  for i:=1 to 7 do
    if level>=i then
    begin
      (FindComponent('memNotes'+inttostr(i)) as TMemo).Enabled :=
true;
```

```
      (FindComponent('memNotes'+inttostr(i)) as TMemo).Text :=
Notes[i];
    end
    else (FindComponent('memNotes'+inttostr(i)) as TMemo).Enabled :=
false;
end;

function TfrmNotes.ShowAllNotes(inNotes:array of string):integer;
var i:integer;
begin
  for i:=1 to 7 do Notes[i] := inNotes[i-1];
  result := ShowModal ;
end;

procedure TfrmNotes.GlobalKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if key = 13 then SwitchControls(Sender) ;  //Αν πατήθηκε το enter
πήγαινε στο επόμενο control
  if key = 27 then  //Αν πατήθηκε το backspace τότε
  begin
    if (Sender as TWinControl).name = 'memNotes1' then Close   //Αν
είναι στο πρώτο πεδίο κλείνει η φόρμα
    else
//αλλιώς καθαρίζει τη φόρμα
    begin
      memNotes1.Text := Notes[1];
      memNotes2.Text := Notes[2];
      memNotes3.Text := Notes[3];
      memNotes4.Text := Notes[4];
      memNotes5.Text := Notes[5];
      memNotes6.Text := Notes[6];
      memNotes7.Text := Notes[7];

      memNotes1.SetFocus ;
    end;
  end;
end;

procedure TfrmNotes.GlobalEnter(Sender: TObject);
begin
  if Sender is TMemo then
  begin
    (Sender as TMemo).SelStart:=length((Sender as TMemo).Text);
    (Sender as TMemo).SelLength := 0;
  end;

  memNotes1.Text := trim(memNotes1.Text);
  memNotes2.Text := trim(memNotes2.Text);
  memNotes3.Text := trim(memNotes3.Text);
  memNotes4.Text := trim(memNotes4.Text);
  memNotes5.Text := trim(memNotes5.Text);
  memNotes6.Text := trim(memNotes6.Text);
  memNotes7.Text := trim(memNotes7.Text);
end;

end.
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Software Engineering (6th Edition), Ian Sommerville, Addison-Wesley Pub Co (2000)

[2] Software Requirements Second Edition, Karl E. Wiegers, Microsoft Press (2003)

[3] Software Reuse: A Standards-Based Guide, Carma McClure, Wiley-IEEE Press (2001)

[4] The Object-Oriented Thought Process, Matt Weisfeld-Bill McCarty, Sams (2000)

[5] Data Modeling Essentials 2nd Edition: A Comprehensive Guide to Data Analysis, Design, and Innovation, Graeme C. Simsion-Graham C. Witt, The Coriolis Group (2000)

[6] Microsoft SQL Server 2000 Unleashed (2nd Edition), Ray Rankins-Paul Jensen-Paul Bertucci, Sams (2002)

[7] Mastering Delphi 6, Marco Cantu, Sybex (2001)

[8] Delphi 6 Developer Guide, Xavier Pacheco-Steve Teixeira-David Intersimone, Sams (2001)

[9] Delphi Component Design, Danny Thorpe, Addison-Wesley Pub Co (1997)

ΗΜΕΡΟΛΟΓΙΟ ΣΥΝΑΝΤΗΣΕΩΝ

Ημερομηνία	Συμμετέχοντες	Περιεχόμενο συνάντησης
4/10/2002	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Πρώτη γνωριμία και καθορισμός του θέματος με ακρίβεια (απόφαση για την ανάπτυξη του υποσυστήματος του λογιστηρίου)
15/10/2002	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Συζήτηση για τη πλατφόρμα ανάπτυξης και εισαγωγή στις έννοιες του λογιστηρίου
24/10/2002	κ. Δημήτριος Κονετάς	Καθορισμός της βιβλιογραφίας σε γενικές γραμμές καθώς και κατευθυντήριες γραμμές για το μοντέλο ανάπτυξης που θα ακολουθηθεί
6/11/2002	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Πρώτο μέρος της απόκτησης των απαιτήσεων μέσω της μελέτης του υπάρχοντος συστήματος και συζήτηση με τον κ. Ναυρόζογλου
25/11/2002	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Το δεύτερο μέρος της συζήτησης για τις λειτουργικές απαιτήσεις καθώς και συζήτηση για τα ιδιαίτερες απαιτήσεις του συγκεκριμένου συστήματος
4/12/2002	κ. Δημήτριος Κονετάς	Ομαδοποίηση απαιτήσεων και αρχιτεκτονική σχεδίαση του συστήματος
21/12/2002	κ. Δημήτριος Κονετάς	Συζήτηση επί των λεπτομερειών της σχεδίασης και καθορισμός των επιλογών σχετικά με την υλοποίηση
16/2/2003	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Επίδειξη του μέχρι στιγμής προϊόντος της υλοποίησης. Παρατηρήσεις για το σύστημα και προτάσεις επέκτασης της λειτουργηκότητας.
12/3/2003	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Επίδειξη πρώτης έκδοσης με πλήρη λειτουργικότητα. Νέες προτάσεις αλλαγών και διορθώσεις.
29/3/2003	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Μία ακόμα βελτιωμένη έκδοση που ορίζεται ως beta. Οι παρατηρήσεις περιορίζονται μόνο σε θέματα γραφικών διεπαφών.
20/4/2003	κ. Αβραάμ Ναυρόζογλου κ. Δημήτριος Κονετάς	Μετά από συνεχή ηλεκτρονική επικοινωνία για διορθώσεις της beta έκδοσης γίνεται η τελική επικύρωση και αποδοχή του συστήματος
4/5/2003	κ. Δημήτριος Κονετάς	Καθορισμός της δομής του γραπτού μέρους και επιλογή αποσπασμάτων από τη βιβλιογραφία
22/5/2003	κ. Δημήτριος Κονετάς	Παράδοση του κειμένου και προτάσεις για διόρθωση και αναθεώρηση του γραπτού
30/5/2003	κ. Δημήτριος Κονετάς	Τελική επικύρωση της εργασίας και αποτίμηση της όλης διαδικασίας