Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Τραπεζικό σύστημα Διαχείρισης Πελατειακών Σχέσεων myBank**
Title	**Banking Customer Relationship Management system myBank**
Ονοματεπώνυμο Φοιτητή	**Στέφανος Καρατζογιάννης**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΣΠ/10042**
Επιβλέπων	**Μαρία Βίρβου, Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2013**

Τριμελής Εξεταστική Επιτροπή

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

Περιεχόμενα

Ευρετήριο Εικόνων - Πινάκων

Πρόλογος

Στην κοινωνία που ζούμε, όπου η αγορά έχει γίνει πάρα πολύ ανταγωνιστική, οι διοικούντες των επιχειρήσεων θα πρέπει να είναι πάρα πολύ προσεκτικοί στον τρόπο που διαχειρίζονται την επιχείρησή τους, στις πολιτικές που ακολουθούν και στις αποφάσεις που παίρνουν.

Ειδικότερα, στον τραπεζικό κλάδο, στις μέρες μας, τα πράγματα είναι ακόμη πιο δύσκολα. Οι τράπεζες βρίσκουν πολλές δυσκολίες στην προσπάθεια να ανταπεξέλθουν στην αγορά και ο ανταγωνισμός μεταξύ τους είναι πολύ μεγάλος.

Το διοικητικό προσωπικό μιας τράπεζας, θα πρέπει να σχεδιάσει με πολλή προσοχή το πώς θα ενεργήσει σε σχέση με τον οργανισμό του. Πέρα όμως από τις διοικητικές ικανότητες και την εμπειρία που θα πρέπει να έχουν οι διοικούντες μιας τράπεζας, για ανταπεξέλθουν στις ανάγκες της αγοράς, θα πρέπει να έχουν ένα πληροφοριακό σύστημα που να τους επιτρέπει να γνωρίζουν την κατάσταση του οργανισμού τους, να ξέρουν τους πελάτες τους και τις ανάγκες τους και να είναι λειτουργικό ως προς τις υπηρεσίες που προσφέρει ο οργανισμός.

Ένα τέτοιο πληροφοριακό σύστημα, αρχικά θα πρέπει να περιλαμβάνει όλα τα βασικά κομμάτια ενός κλασσικού τραπεζικού συστήματος, όπως τα παρακάτω:

- Σύστημα Διαχείρισης Πελατείας
- Σύστημα Λογαριασμών (Καταθέσεις)
- Σύστημα Δανείων

Πέρα όμως από αυτά τα βασικά στοιχεία χρειάζεται, όπως όλοι οι σύγχρονοι οργανισμοί, ένα σύστημα το οποίο να επιτρέπει στην τράπεζα να γνωρίζει τους πελάτες της και τις ανάγκες τους τις οποίες μπορεί να ικανοποιήσει, να σχεδιάζει στρατηγικές και γενικότερα να δημιουργεί μια καλή σχέση του οργανισμού με τον πελάτη.

Αυτό λοιπόν που χρειάζεται μια σύγχρονη τράπεζα, είναι ένα τραπεζικό σύστημα που μέσα να συμπεριλαμβάνει και ένα σύστημα «Διαχείρισης Πελατειακών Σχέσεων» (CRM). Ένα τραπεζικό σύστημα μπορούμε να καταλάβουμε τι εξυπηρετεί, όμως τι είναι ένα σύστημα διαχείρισης πελατειακών σχέσεων (πλέον CRM);

Παρακάτω, θα προσπαθήσουμε να εξηγήσουμε σε λίγες γραμμές ένα σύστημα CRM και να καταλάβουμε πόσο μεγάλο όφελος έχει ένας οργανισμός που χρησιμοποιεί ένα τέτοιο σύστημα.

Ένα επιτυχές σύστημα CRM, αναπτύσσεται με γνώμονα την βελτίωση της σχέσης της επιχείρησης με τον πελάτη. Ένα σύστημα CRM βοηθάει έναν οργανισμό να αποκτήσει περισσότερους πελάτες και κατ' επέκταση αυξάνει τα κέρδη.

Στις μέρες μας, όλες οι αναπτυσσόμενες επιχειρήσεις διαχειρίζονται τη σχέση με τους πελάτες τους και όλες τις πληροφορίες που αφορούν αυτή τη σχέση, με διάφορους τρόπους. Κάποιοι κρατάνε κάρτες πελατών, κάποιοι άλλοι κρατούν σημειώσεις στις φορητές τους συσκευές χωρίς κανείς άλλος να μπορεί να δει αυτές τις πληροφορίες. Άλλοι χρησιμοποιούν αρχεία Excel ή έγγραφα στο διαδίκτυο (Google Drive, Microsoft SkyDrive κ.α.). Αυτά μπορεί να βοηθήσουν για κάποιο καιρό για μικρές επιχειρήσεις οι οποίες δεν έχουν τα φόντα να αναπτυχθούν. Αν θέλουν λοιπόν να αναπτυχθούν, μάλλον θα πρέπει να σκεφτούν να χρησιμοποιήσουν ένα σύστημα CRM, το οποίο θα τους βοηθήσει να συλλέξουν χρήσιμες πληροφορίες σε ένα κεντρικό σύστημα, οι οποίες θα είναι διαθέσιμες σε όλους τους χρήστες του συστήματος.

Ένα σύστημα CRM είναι ένα πελατοκεντικό σύστημα το οποίο, αν ένας οργανισμός θέλει να το χρησιμοποιήσει, θα πρέπει να αποκτήσει και την αντίστοιχη φιλοσοφία. Θα πρέπει αυτή η φιλοσοφία να υιοθετηθεί από όλα τα μέλη της επιχείρησης, από τον διευθύνοντα σύμβουλο μέχρι τον τελευταίο υπάλληλο. Για να είναι επιτυχές, ένα σύστημα CRM σε μια επιχείρηση, θα πρέπει να έχουν οριστεί κατάλληλες διαδικασίες ώστε να προάγουν αυτή τη φιλοσοφία, αλλά επίσης να επιλεγεί και η σωστή τεχνολογία η οποία θα είναι σε θέση να εξυπηρετήσει αυτές τις διαδικασίες και να μπορεί εμφανίσει όλες αυτές τις πληροφορίες στους

τελικούς χρήστες, με τον καλύτερο δυνατό τρόπο, ώστε οι τελικοί χρήστες να μην διστάζουν να χρησιμοποιήσουν αυτό το σύστημα CRM.

Η διαχείριση πελατειακών σχέσεων (CRM) είναι μια στρατηγική που χρησιμοποιεί μια επιχείρηση για να μάθει περισσότερα για τις ανάγκες και τις συμπεριφορές των πελατών της, ώστε να μπορέσει να αναπτύξει καλές σχέσεις μαζί τους. Τέλος, να αναφέρουμε πως όταν σκεφτόμαστε τη διαχείριση πελατειακών σχέσεων (CRM) ως κάτι τεχνικό, αυτό είναι λάθος. Η διαχείριση πελατειακών σχέσεων είναι μια διαδικασία, η οποία θα βοηθήσει την επιχείρηση να συλλέξει διάφορα κομμάτια πληροφοριών για τους πελάτες της, τις πωλήσεις της, την αποδοτικότητα του marketing, τις τάσεις της αγοράς, την αποτελεσματικότητα. Όλα αυτά λοιπόν, θα οδηγήσουν τους διοικούντες της επιχείρησης στο να έχουν μια συνολική και ενημερωμένη εικόνα για τον οργανισμό τους και θα τους οδηγήσει σε σωστότερες αποφάσεις.

Abstract

In our society, where the market has become very competitive, the business executives of an organization should be very careful concerning the way they manage their businesses, the policies they abide by and the decisions they take.

Especially, in the banking industry, in our days, things are even more difficult. Banks have many difficulties trying to cope with the market needs and the competition among them is great.

The executives of a bank should be very careful when designing their acts upon their organization. Apart from the administrative skills and the experience that the executives of a bank should have, in order to cope with market needs, they should have an information system that allows them to be informed about their organization's status, know their customers and their needs, but also a system that is functional, concerning the services that they provide.

Such an information system, at first should include all the basic components of a classic banking system, like the following:

- Customer Management System
- Account System (Deposits)
- Loans System

Beyond these basic components, what's needed, as in every modern organization, is a system allows the bank to know their customers and their needs which the organization can satisfy, to be able to design strategies and in general, establish a good relationship between the customer and the organization.

Thus, what a modern bank needs is a banking system that also contains a CRM System (Customer Relationship Management). We can understand what a banking system serves, but what is a CRM system?

In the following few lines, we are going to try to explain what a CRM system is and realize the great benefits that an organization should have, when using such a system.

A successful CRM system is built around the people relationships and aims on improving these relationships. A CRM system helps an organization get more customers, which of course means more profits.

Today, all growing businesses manage customer connections and all information around them, in a variety of ways. Others use note cards, others keep notes on their mobile devices without being able to share them with anybody else. Others use Excel files or online documents (Google Drive, Microsoft SkyDrive etc). These things can help a small business for a certain amount of time, but they don't have any perspective of growing. What they actually need, if they want to have fast growth, is to use a CRM system that is going to help them collect useful information on a central system, which will be available throughout all the system users.

A CRM system is a customer-centric system. If an organization wants to use a CRM system, they should evolve such a philosophy. This philosophy should be adopted by every member of the organization, from the CEO to the last end user. In order to be successful, a CRM system in a business, proper processes should be established, that enhance this philosophy, but also the right technology should be selected, that can support these processes and provide all this data to the end users, in the best way, so that the end users won't hesitate to extensively use this CRM system.

Customer Relationship Management is a strategy used to learn more about customer's needs and behaviors in order to develop stronger business relationships with them. Concluding, we should say that when we think of CRM as something technological, we are wrong. Customer Relationship Management is a process, which will help a business bring together pieces of information about customers, sales, marketing effectiveness, market trends and responsiveness. All these things, will help the business executives to have an overall and updated view of their organization and lead them to better decision making.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω το Ίδρυμα Κρατικών Υποτροφιών (ΙΚΥ) που επιλέγοντάς με ως υπότροφο στα προγράμματα υποτροφιών για μεταπτυχιακές σπουδές στην Ελλάδα, μου έδωσε την δυνατότητα να παρακολουθήσω αυτό το μεταπτυχιακό πρόγραμμα. Η οικονομική βοήθεια του ΙΚΥ σε αυτή μου την προσπάθεια ήταν πολύ σημαντική γιατί διαφορετικά ίσως να μην είχα τη δυνατότητα να παρακολουθήσω μεταπτυχιακές σπουδές.

Στη συνέχεια θα ήθελα να ευχαριστήσω τους συναδέλφους στο εργασιακό μου περιβάλλον που με στήριξαν στην προσπάθειά μου και τους προϊσταμένους μου που με διευκόλυναν, καθώς συμμετείχα στο μεταπτυχιακό παράλληλα με τη δουλειά μου. Θα ήθελα να ευχαριστήσω ιδιαιτέρως τον προϊστάμενό μου Μαρίνο Γεώργιο, που όχι μόνο δημιούργησε ευνοϊκές συνθήκες για να ανταπεξέλθω στις υποχρεώσεις μου στο μεταπτυχιακό, αλλά με έχει βοηθήσει τόσα χρόνια να αποκτήσω πολλές γνώσεις στον χώρο της πληροφορικής, τις οποίες χρησιμοποίησα και για την εκπόνηση αυτής της εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου που και αυτοί με στήριξαν ψυχολογικά και οικονομικά στο να παρακολουθήσω το μεταπτυχιακό πρόγραμμα, ενώ παράλληλα εργαζόμουν.

Τελευταίο αλλά εξίσου σημαντικό, θα ήθελα να ευχαριστήσω το Πανεπιστήμιο Πειραιά που με επέλεξε και μου έδωσε την ευκαιρία να συμμετάσχω στο μεταπτυχιακό πρόγραμμα και να αποκτήσω πολλές γνώσεις που θα με βοηθήσουν στη συνέχεια. Ειδικά θα ήθελα να ευχαριστήσω από καρδιάς, τους καθηγητές με τους οποίους έκανα αυτή την εργασία, κα Μαρία Βίρβου κο Ευθύμιο Αλέπη. Η συνεργασία μας ήταν άψογη, υπήρχε άμεση επικοινωνία και καθοδήγηση και η εκπόνηση της εργασίας αυτής ήταν μια ευχάριστη εμπειρία για εμένα.

1. Εισαγωγή

Σε αυτή την εργασία θα γίνει η υλοποίηση ενός πληροφοριακού συστήματος που θα έχει τα χαρακτηριστικά που αναφέραμε προηγουμένως, δηλαδή ένα πληροφοριακό σύστημα που θα έχει όλα τα βασικά στοιχεία ενός τραπεζικού συστήματος αλλά επίσης και πολλά στοιχεία ενός συστήματος CRM (Διαχείρισης Πελατειακών Σχέσεων).

Αυτό το σύστημα προορίζεται για τραπεζικούς οργανισμούς και προσφέρει πολλές δυνατότητες, τόσο στο τραπεζικό κομμάτι όσο και στο κομμάτι του CRM (Operational CRM). Το σύστημα αποτελείται από τα εξής βασικά κομμάτια:

1.1 Πελάτες

Είναι το υποσύστημα που διαχειρίζεται τα βασικά στοιχεία των πελατών της τράπεζας. Οι χρήστες μπορούν να αναζητήσουν, να επεξεργαστούν και να εισάγουν νέους πελάτες στο σύστημα.

1.2 Λογαριασμοί

Είναι το υποσύστημα των λογαριασμών που εξυπηρετεί ένα βασικό σύστημα καταθέσεων. Οι χρήστες μπορούν να αναζητήσουν, να διαχειριστούν τους λογαριασμούς των πελατών, καθώς και να ανοίξουν νέο λογαριασμό (Τρεχούμενο, Όψεως ή Ταμιευτηρίου). Επίσης, μπορεί να γίνει ανάληψη και κατάθεση ποσού.

1.3 Αιτήσεις Δανείων

Είναι το σύστημα που τηρείται η διαδικασία με την οποία γίνεται η έγκριση των δανείων που χορηγεί η τράπεζα στους πελάτες της. Όταν ένας πελάτης θέλει να πάρει ένα δάνειο, τότε ο χρήστης του καταστήματος θα πρέπει να εισάγει μια νέα αίτηση δανείου στο σύστημα, στην οποία συμπληρώσει κάποια βασικά στοιχεία του πελάτη, όπως είναι το ετήσιο εισόδημά του, το προϊόν που επιθυμεί (καταναλωτικό, στεγαστικό δάνειο κτλ) και το αιτούμενο ποσό δανείου. Όταν συμπληρωθούν όλα τα απαραίτητα στοιχεία της αίτησης, τότε ο υπάλληλος του καταστήματος θα πρέπει να συλλέξει όλα τα απαραίτητα δικαιολογητικά που είναι απαραίτητα, ανάλογα με το προϊόν και τέλος να στείλει την αίτηση προς έγκριση.

Αφού η αίτηση είναι σε κατάσταση προς έγκριση, τότε ο εγκριτής δανείου θα πρέπει να μπει στο σύστημα, να δει ποιες αιτήσεις υπάρχουν σε αυτή την κατάσταση. Τότε θα πρέπει να εξετάσει τα στοιχεία της αίτησης, δηλαδή να κάνει τα παρακάτω:

- Να ελέγξει αν ο πελάτης έχει ήδη κάποιο άλλο δανειακό προϊόν στο σύστημα.
- Να ελέγξει αν ο πελάτης έχει κάποιο άλλο δανειακό προϊόν εκτός της τράπεζας. Αυτό γίνεται μέσα από το εθνικό σύστημα του Τειρεσία, στο οποίο έχουν πρόσβαση όλες οι τράπεζες.
- Να εξετάσει το αιτούμενο ποσό δανείου σε σχέση το σύνολο των πιστώσεων που ήδη έχει ο πελάτης, σε συνδυασμό με το ετήσιο του εισόδημα. Ουσιαστικά να εξετάσει την πιστοληπτική ικανότητα του πελάτη.

Αφού λοιπόν γίνουν όλα αυτά, τότε ο εγκριτής θα πρέπει να κρίνει αν η τράπεζα θα πρέπει να δώσει στον πελάτη το δάνειο, σύμφωνα με την πολιτική του οργανισμού σχετικά με τις δανειοδοτήσεις. Είναι σημαντικό για την τράπεζα να δίνει δάνεια μόνο στους πελάτες που μπορούν πραγματικά να αποπληρώσουν, διότι διαφορετικά δεν θα μπορέσει να πάρει πίσω τα κεφάλαια που έδωσε και τους τόκους, με αποτέλεσμα να έχει ζημιά.

Αν εγκριθεί ένα δάνειο, τότε ο υπάλληλος του καταστήματος θα πρέπει να πάει στην αίτηση, η οποία θα είναι πλέον σε κατάσταση «Ολοκληρωμένη» και να κάνει άνοιγμα δανείου, αφότου φυσικά ο πελάτης υπογράψει την σύμβαση του δανείου.

1.4 Δάνεια

Αφότου ανοιχτεί ένα νέο δάνειο, τότε ο υπάλληλος του καταστήματος θα πρέπει να πάει να το συνδέσει με κάποιο λογαριασμό καταθέσεων. Σε αυτό τον λογαριασμό θα εκταμιευτεί το ποσό του δανείου κι από τότε θα ξεκινήσει ο υπολογισμός των δόσεων. Ο πελάτης θα πρέπει κάθε μήνα να κάνει εξόφληση δόσης, μέχρι να γίνει ολική εξόφληση του δανείου.

1.5 Ημερολόγιο

Οι χρήστες του συστήματος μπορούν να προγραμματίσουν ραντεβού και ενέργειες σε σχέση με πελάτες (ή και όχι) και να μπορούν να τις βλέπουν στο προσωπικό τους ημερολόγιο. Έτσι ένας υπάλληλος μπορεί να προγραμματίσει συνάντηση με κάποιον πελάτη στο κατάστημα κάποια συγκεκριμένη στιγμή, ή τηλεφωνική επικοινωνία με τον πελάτη και να καταγράψει στο σύστημα πληροφορίες σχετικά με αυτή την επικοινωνία με τον πελάτη. Αυτό είναι ένα από τα κομμάτια CRM του συστήματος και είναι πάρα πολύ σημαντικό διότι κάθε χρήστης του συστήματος θα μπορεί να βλέπει όλο το ιστορικό της επικοινωνίας του πελάτη με την τράπεζα και θα έχει πιο ολοκληρωμένη εικόνα γι' αυτόν.

1.6 Ειδοποιήσεις - Καμπάνιες

Σε αυτό το κομμάτι, ο επιχειρηματικός χώρος της τράπεζας, μπορεί να σχεδιάσει και να υλοποιήσει στρατηγικές μέσω καμπανιών. Αυτές οι καμπάνιες μπορεί να είναι προωθητικές, δηλαδή να αποσκοπούν στην πώληση προϊόντων προς τους πελάτες είτε και όχι.

Το προσωπικό του επιχειρηματικού χώρου, μπορεί μέσα από το σύστημα να δημιουργήσει μια νέα καμπάνια και αφού ορίσει όλες τις παραμέτρους του να δημιουργήσει ειδοποιήσεις προς τους υπαλλήλους των καταστημάτων ώστε να έρθουν σε επικοινωνία με τον πελάτη ή να προβεί στις απαραίτητες ενέργειες για την καμπάνια.

Οι υπάλληλοι με τη σειρά τους θα, πρέπει να πάνε στις ειδοποιήσεις, που θα αφορούν πελάτες που ανήκουν στο κατάστημά τους και να κάνουν τις κατάλληλες ενέργειες για την κάθε ειδοποίηση. Σύμφωνα με την πρόοδο που έχουν κάνει, μπορούν να αλλάζουν την κατάσταση της ειδοποίησης, από εκκρεμή σε «σε εξέλιξη», «ολοκλήρωση» ή ότι άλλο έχει οριστεί στις παραμέτρους της εκάστοτε καμπάνιας. Επίσης, οι χρήστες μπορούν να επισυνάψουν κάποιο σχόλιο ή έγγραφο πάνω στην ειδοποίηση ή να προγραμματίσουν κάποια ενέργεια που αφορά την ειδοποίηση. Αυτές είναι πολύ χρήσιμες πληροφορίες που συλλέγονται από το σύστημα και βοηθάνε, όχι μόνο τους χρήστες αλλά και τον επιχειρηματικό χώρο, ο οποίος έτσι μπορεί να πληροφορηθεί για την πρόοδο και την αποτελεσματικότητα των καμπανιών που έχει σχεδιάσει. Αυτό το κομμάτι του συστήματός μας είναι πάρα πολύ σημαντικό για ένα σύστημα CRM.

1.7 Επισυναπτόμενα - Σχόλια

Οι χρήστες έχουν τη δυνατότητα να επισυνάψουν σχόλια και έγγραφα που μπορεί να αφορούν κάποιον πελάτη, ή κάποια ειδοποίηση του πελάτη. Φυσικά όλες αυτές οι πληροφορίες είναι διαθέσιμες κεντρικά και όλοι οι χρήστες μπορούν να τις δουν στην οθόνη του πελάτη. Κι αυτό είναι πολύ χρήσιμο ώστε οι χρήστες που έχουν επαφή με τον πελάτη, να έχουν πιο ολοκληρωμένη εικόνα γι' αυτόν.

1.8 Πληροφόρηση Διοίκησης

Η πληροφόρηση διοίκησης γνωστή και ως Bisiness Intelligence (BI), είναι μια σειρά από αναφορές που απευθύνονται στο επιχειρηματικό προσωπικό της τράπεζας από τις οποίες μπορούν να πληροφορηθούν για την κατάσταση του συστήματος. Για παράδειγμα υπάρχουν αναφορές για το συνολικά κεφάλαια της τράπεζας, τα συνολικά ποσά δανείων, εισροές και εκροές κεφαλαίων και ενημέρωση ειδοποιήσεων καμπανιών. Όλα αυτά δίνουν τη δυνατότητα να σε αυτούς τους χρήστες να έχουν πλήρη εικόνα για την τράπεζα, για τα διαθέσιμα κεφάλαια, την πρόοδο των δανείων και την πρόοδο των καμπανιών που έχουν σχεδιάσει.

1.9 Διαχείριση Χρηστών

Στο σύστημα υπάρχουν διάφοροι ρόλοι, σύμφωνα με τις αρμοδιότητες του κάθε χρήστη. Ο κάθε χρήστης έχει διαφορετικές δυνατότητες πάνω στην εφαρμογή. Οι ρόλοι του συστήματος είναι οι παρακάτω:

- Teller Καταστήματος
- Διευθυντής Καταστήματος
- Εγκριτής Δανείων
- Διαχειριστής Χρηστών
- Προσωπικό Επιχειρηματικού Χώρου

Από αυτούς τους ρόλους, ο Διευθυντής καταστήματος και ο διαχειριστής χρηστών μπορούν να δημιουργήσουν χρήστες και να αλλάξουν τις προσβάσεις τους σύμφωνα με τις ανάγκες που υπάρχουν. Η διαφορά τους είναι ότι ο διευθυντής καταστήματος μπορεί να διαχειριστεί μόνο χρήστες που ανήκουν στο κατάστημά του.

Τέλος να αναφέρουμε, πως όλη η πληροφόρηση σχετικά με κάποιον πελάτη που υπάρχει στο σύστημα, είναι ορατή στην οθόνη πελάτη. Λογαριασμοί, δάνεια, ειδοποιήσεις, σχόλια, ενέργειες και ραντεβού. Έτσι, όταν οι χρήστες του συστήματος κάνουν κάποια εργασία πάνω στον πελάτη ή βρίσκονται σε επαφή με τον πελάτη μαζί του, είναι πλήρως ενημερωμένοι για την σχέση του με την τράπεζα, πράγμα που είναι πολύ σημαντικό και αποτελεί στόχο ενός συστήματος CRM.

Ένα απλό παράδειγμα το οποίο δείχνει πόσο σημαντική είναι αυτή η πληροφόρηση είναι η παρακάτω. Η τράπεζα θέλει να προωθήσει πιστωτικές κάρτες προς πελάτες που έχουν κεφάλαιο μεγαλύτερο των 5.000€ αλλά δεν έχει το κατάλληλο σύστημα για να το υποστηρίξει μηχανογραφικά. Ο πελάτης την πρώτη εβδομάδα πηγαίνει στο κατάστημα και εξυπηρετείται για κάποια δουλειά από τον υπάλληλο ο οποίος αφού βλέπει ότι οι καταθέσεις του πελάτη ξεπερνάνε τα 5.000€ του προτείνει αγορά πιστωτικής κάρτας. Ο πελάτης δέχεται και κάνει την αίτηση για την κάρτα.

Την επόμενη βδομάδα ο πελάτης πηγαίνει πάλι στο κατάστημα για κάποια άλλη δουλειά, αλλά αυτή την φορά εξυπηρετείται από άλλον υπάλληλο. Αυτός ο υπάλληλος με την σειρά του ξαναπροτείνει στον πελάτη αγορά πιστωτικής κάρτας, ενώ αυτός έχει ήδη δεχτεί, αφού ο υπάλληλος δεν μπορεί να γνωρίζει το ιστορικό της σχέσης της τράπεζας με τον πελάτη. Αυτό λοιπόν δημιουργεί αρνητικά συναισθήματα στον πελάτη και θα μπορούσε να μην συμβεί αν η τράπεζα είχε ένα στοιχειώδες CRM σύστημα. Στην περίπτωσή μας θα μπορούσε να είναι μια καμπάνια προώθησης πιστωτικών καρτών, στην οποία ο πρώτος υπάλληλος θα είχε σημειώσει ότι ο πελάτης δέχτηκε και έκανε αίτηση. Το ίδιο παράδειγμα θα μπορούσε να υπάρχει και σε ένα τηλεφωνικό κέντρο μιας εταιρίας όπου κάνει τηλεφωνήματα για προώθηση προϊόντων.

2. Ανασκόπηση Πεδίου

Σε αυτή την ενότητα εξετάσουμε διάφορες εφαρμογές που χρησιμοποιούνται στον κόσμο, οι οποίες εξυπηρετούν παρόμοιες ανάγκες σε τραπεζικές επιχειρήσεις. Οι περισσότερες από αυτές τις εφαρμογές έχουν αναπτυχθεί από πολύ μεγάλες εταιρίες πληροφορικής και εξυπηρετούν συστήματα CRM και τραπεζικά (Banking) συστήματα.

Αυτές λοιπόν οι εφαρμογές, που θα δούμε χωρίζονται σε δύο κατηγορίες: CRM και Banking και είναι οι παρακάτω:

- SugarCRM
- Microsoft Dynamics CRM
- SAP CRM
- T24 Temenos

2.1 Εφαρμογές CRM

2.1.1 SugarCRM

Το SugarCRM είναι η μεγαλύτερη εφαρμογή CRM ανοιχτού κώδικα (open source). Έχει πάνω από 7.000 πελάτες και περισσότερους από μισό εκατομμύριο χρήστες που βασίζονται σε αυτό για το marketing, την αύξηση των πωλήσεων, την διατήρηση και προσέλκυση πελατών και δημιουργία προσαρμοσμένων επιχειρησιακών εφαρμογών.

Ένα από τα μοναδικά πράγματα του SugarCRM στον κόσμο των συστημάτων CRM, είναι ότι είναι εξ' ολοκλήρου ανοιχτού κώδικα, που σημαίνει ότι ο πηγαίος κώδικας της εφαρμογής είναι διαθέσιμος σε οποιονδήποτε χρήστη, προγραμματιστή ή πελάτη του προϊόντος. Αυτό βοηθάει τους προγραμματιστές να προσαρμόσουν και να προγραμματίσουν πάνω του με μεγάλη ευκολία. Οι συνεργάτες του SugarCRM μπορούν να προσαρμόσουν εφαρμογές που θα είναι έτοιμες να χρησιμοποιηθούν από πολλές διαφορετικές επιχειρήσεις, σε διαφορετικές αγορές.

Μερικά από τα βασικά σημεία το SugarCRM είναι τα παρακάτω:

- Πωλήσεις, κρατάει πλήρη ιστορικότητα της δραστηριότητας του πελάτη, έτσι βοηθάει την επιχείρηση να κινεί τις πωλήσεις σε συγκεκριμένες κατευθύνσεις, χωρίς εκπλήξεις, έχοντας καλύτερη εικόνα των αναγκών των πελατών.
- Μάρκετινγκ, κρατώντας πληροφόρηση της αλληλεπίδρασης του πελάτη από όλα τα κανάλια της επιχείρησης, οδηγεί σε σχεδιασμό έγκαιρων και σωστών καμπανιών.
- Υποστήριξη, έχοντας πλήρη εικόνα του πελάτη στο σύστημα, οι χρήστες είναι ικανοποιημένοι και έχουν θέληση να χρησιμοποιήσουν τις δυνατότητές του.
- Ενσωμάτωση με εφαρμογές κοινωνικής δικτύωσης (Social Media)
- Αναφορές, με τις οποίες οι χρήστες μπορούν να παρακολουθήσουν και να μετρήσουν τα σημαντικά «νούμερα» που αφορούν την επιχείρηση.
- Πλατφόρμα, η οποία βοηθά την κάθε επιχείρηση να παραμετροποιήσει και να επεκτείνει την εφαρμογή σύμφωνα με τις διαδικασίες της.
- Διαχείριση Έργω, η οποία βοηθάει στην παρακολούθηση των έργων που τρέχουν μέσα στην επιχείρηση.

2.1.2 Microsoft Dynamics CRM

Το Microsoft Dynamics CRM είναι μια εφαρμογή για επιχειρήσεις που τους επιτρέπει να ανιχνεύουν, να διαχειρίζονται και να ενημερώνονται για δεδομένα που αφορούν την αλληλεπίδραση της επιχείρησης με τους πελάτες της. Το Microsoft Dynamics CRM προσφέρει πολλά προϊόντα για να βοηθήσει τις επιχειρήσεις να απλοποιήσουν διάφορες διαδικασίες, όπως οικονομικές αναλύσεις, σχέσεις με πελάτες, διαχείριση προσωπικού, παραγωγής και πολλά άλλα.

Το Microsoft Dynamics CRM περιέχει τα παρακάτω τρία βασικά κομμάτια:

- Πωλήσεις
- Μάρκετινγκ
- Υπηρεσίες

Για καθένα από αυτά τα κομμάτια, το Microsoft Dynamics CRM επιτρέπει στις επιχειρήσεις να ανιχνεύσουν διάφορους τύπους πληροφοριών σχετικά με τους πελάτες, όπως βλέπουμε παρακάτω.

- Πωλήσεις
    - Λογαριασμοί
    - Επαφές
    - Ευκαιρίες – Δυνατότητες
    - Ανταγωνιστές
    - Προϊόντα
    - Λογική πωλήσεων
    - Παραγγελίες
    - Αποδείξεις
    - Στόχοι
    - Μέτρηση στόχων
    - Συλλογή ερωτημάτων
- Μάρκετινγκ
    - Λογαριασμοί
    - Επαφές
    - Λίστες μάρκετινγκ
    - Καμπάνιες
    - Προϊόντα
- Υπηρεσίες
    - Λογαριασμοί
    - Επαφές
    - Ημερολόγιο υπηρεσιών
    - Υποθέσεις
    - Βάση γνώσης
    - Συμβόλαια
    - Προϊόντα
    - Υπηρεσίες
    - Στόχοι
    - Μέτρηση στόχων
    - Συλλογή ερωτημάτων

Μπορεί κάποια επιχείρηση να θέλει να κρατήσει πληροφορίες πελατών μόνο για μερικά κομμάτια και μερικά κομμάτια να μην έχουν εφαρμογή στην επιχείρηση. Πολλές επιχειρήσεις επεκτείνουν την εφαρμογή για συλλογή άλλων τύπων πληροφοριών. Η ευελιξία του Microsoft Dynamics CRM επιτρέπει στις επιχειρήσεις να αποτυπώσουν οποιουδήποτε τύπου πληροφορία αφορά τους πελάτες. Επιπλέον, οι επιχειρήσεις μπορούν να συλλέξουν δεδομένα που αφορούν τους προμηθευτές, τους συνεργάτες και άλλους σχετικούς φορείς.

Το Microsoft Dynamics CRM είναι μια web-based εφαρμογή που έχει αναπτυχθεί στην πλατφόρμα τεχνολογίας Microsoft .NET Framework. Οι χρήστες μπορούν να έχουν πρόσβαση στην εφαρμογή μέσω από κινητές συσκευές όπως κινητά τηλέφωνα, πέραν από τους ηλεκτρονικούς υπολογιστές.

2.1.3 SAP CRM

Το SAP CRM είναι ένα εργαλείο που όχι μόνο βοηθάει τις επιχειρήσεις να ανταποκριθούν σε βραχυπρόθεσμες ανάγκες, όπως η μείωση του κόστους και η καλύτερη λήψη αποφάσεων, αλλά είναι ένα εργαλείο που ενεργοποιεί δυνατότητες για όλο το μήκος των οργανισμών που βοηθούν τις επιχειρήσεις να είναι ανταγωνιστικές στην αγορά.

Τα βασικά κομμάτια του SAP CRM είναι τα παρακάτω:

- Μάρκετινγκ
- Πωλήσεις
- Εξυπηρέτηση πελατών

Το κομμάτι του μάρκετινγκ του SAP CRM δίνει τη δυνατότητα στις επιχειρήσεις να αποκτούν τη συνολική εικόνα που απαιτείται για τη σωστή λήψη αποφάσεων, την εστίαση στους πελάτες που μπορεί να ενισχύσει την πιστότητα, και τέλος την καλύτερη διαχείριση των πόρων, ώστε να γίνονται περισσότερες ενέργειες με μικρότερο κόστος. Συγκεκριμένα, οι βασικοί άξονες του είναι οι εξής:

- Brand management
- Campaign Management
- Τμηματοποίηση αγοράς
- Real-time offer management
- Διαχείριση πιστότητας πελατών
- E-marketing

Το κομμάτι των πωλήσεων μπορεί να μετατρέψει το δυναμικό των πωλήσεων σε ομάδα αξιόπιστων συμβούλων, ενθαρρύνοντας ταυτόχρονα τη συνεργασία μεταξύ πωλήσεων, marketing και εξυπηρέτησης πελατών στοχεύοντας στην καλύτερη εξυπηρέτηση. Τα συστατικά στοιχεία του SAP CRM που αφορούν τις πωλήσεις είναι τα εξής:

- Πωλήσεις
- e-Commerce
- Interaction Center
- Κανάλια και Συνεργάτες

Τέλος, το κομμάτι της εξυπηρέτησης πελατών βοηθάει τις επιχειρήσεις να μειώσουν τα κόστη της εξυπηρέτησης και ταυτόχρονα να αυξήσουν την ικανοποίηση των πελατών χρησιμοποιώντας το σύστημα.

2.2 Banking Εφαρμογές

2.2.1 T24 Temenos

Το T24 Temenos είναι ένα ολοκληρωμένο τραπεζικό σύστημα για front και back office λειτουργίες τραπεζών, αλλά επίσης και CRM λειτουργίες. Το T24 υποστηρίζει τραπεζικές λειτουργίες όπως οι παρακάτω:

- Retail Banking
- Corporate Banking
- Wholesale Banking
- Universal Banking
- Private Banking

Με ένα σύστημα που λειτουργεί 24 ώρες την ημέρα, το T24 συνδυάζει πλήρη επιχειρηματική λειτουργικότητα με μια προηγμένη, ασφαλή και επεκτάσιμη αρχιτεκτονική που προστατεύει τις τραπεζικές λειτουργίες ενός οργανισμού, ώστε να ανταπεξέλθει στις προκλήσεις της αγοράς και των τραπεζικών τεχνολογιών του σήμερα και του μέλλοντος.

Το T24 είναι μέσα στα πιο επιτυχημένα συστήματα core banking στον κόσμο, τα τελευταία χρόνια, και έχει αναπτυχθεί με αρχιτεκτονική SOA (Service Oriented Architecture) που είναι χωρισμένη σε κομμάτια, ώστε να μπορούν οι τράπεζες αναπτύσσουν και να ενσωματώνουν τη λειτουργικότητα που χρειάζονται με μεγάλη ευκολία.

Μπορεί πολύ εύκολα να ενσωματωθεί με ήδη υπάρχοντα τραπεζικά συστήματα, δίνοντας τη δυνατότητα ενσωμάτωσης και ελέγχου διαφόρων καναλιών τραπεζικών υπηρεσιών για τους πελάτες αποτελεσματικά, με ασφάλεια και χαμηλό κόστος. Το T24 μπορεί επίσης να υλοποιηθεί ως ένα ολοκληρωμένο τραπεζικό μοντέλο με ενσωματωμένες βέλτιστες πρακτικές για τραπεζικές λειτουργίες, αλλά επίσης και ως ένα εύκολα παραμετροποιήσιμο τραπεζικό μοντέλο που μπορεί να προσαρμοστεί για συγκεκριμένες ανάγκες.

3. Παρουσίαση και χρήση εφαρμογής

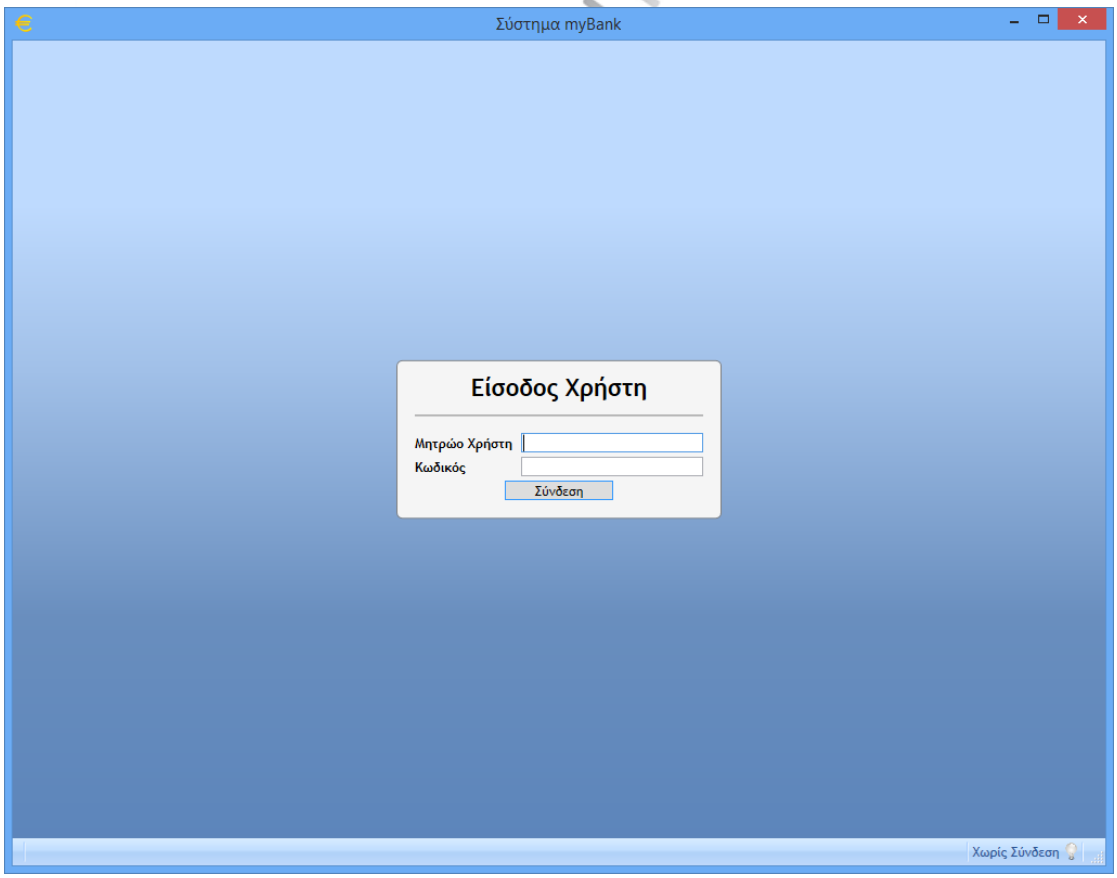
3.1 Αρχική Οθόνη

Όταν ανοίγει η εφαρμογή εμφανίζεται η πρώτη οθόνη στην οποία ο χρήστης πρέπει να κάνει είσοδο στο σύστημα. Ο χρήστης καλείται να βάλει το μητρώο του (κωδικός χρήστη) και τον κωδικό του. Βάζοντας το μητρώο του και τον κωδικό του, το σύστημα αναγνωρίζει τον χρήστη και πλέον έχει διαθέσιμες όλες τις πληροφορίες που τον αφορούν. Αυτά τα στοιχεία είναι το ονοματεπώνυμό του, το κατάστημα στο οποίο ανήκει καθώς και τις προσβάσεις που έχει ο χρήστης στην εφαρμογή. Σε περίπτωση που το μητρώο χρήστη και ο κωδικός δεν συμφωνούν με τα στοιχεία του συστήματος, τότε βγαίνει μήνυμα που λέει στον χρήστη ότι τα στοιχεία που έδωσε είναι λανθασμένα και δεν μπορεί να προχωρήσει.

Όταν ο χρήστης συνδεθεί επιτυχώς στο σύστημα τότε έχει διαθέσιμες όλες τις λειτουργίες για τις οποίες έχει πρόσβαση. Όλες οι πιθανές προσβάσεις που μπορεί να έχει ένας χρήστης είναι οι παρακάτω:

- Teller καταστήματος
- Διευθυντής καταστήματος
- Εγκριτής Δανείων
- Διαχειριστής Χρηστών
- Προσωπικό Επιχειρηματικού Χώρου

Η αρχική οθόνη της εφαρμογής φαίνεται στην παρακάτω εικόνα.



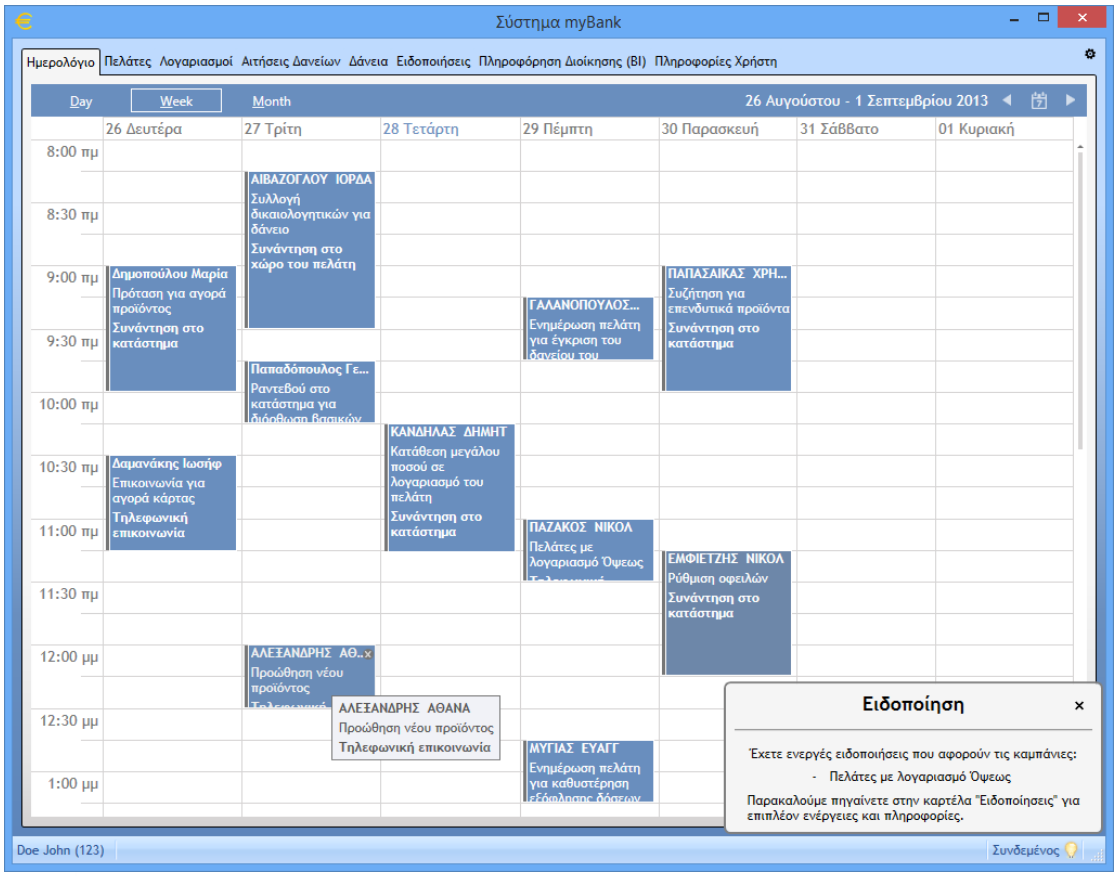
Εικόνα 1: Αρχική Οθόνη

3.2 Ημερολόγιο

Μετά την επιτυχή σύνδεση του χρήστη, η πρώτη οθόνη που εμφανίζεται είναι το Ημερολόγιο. Σε αυτή την οθόνη φαίνεται το ημερολόγιο του χρήστη στο οποίο εμφανίζονται όλα τα ραντεβού – ενέργειες που έχει προγραμματίσει ο χρήστης. Αυτές οι ενέργειες μπορούν να αφορούν κάποια ενέργεια σχετικά με έναν πελάτη και είναι οι παρακάτω:

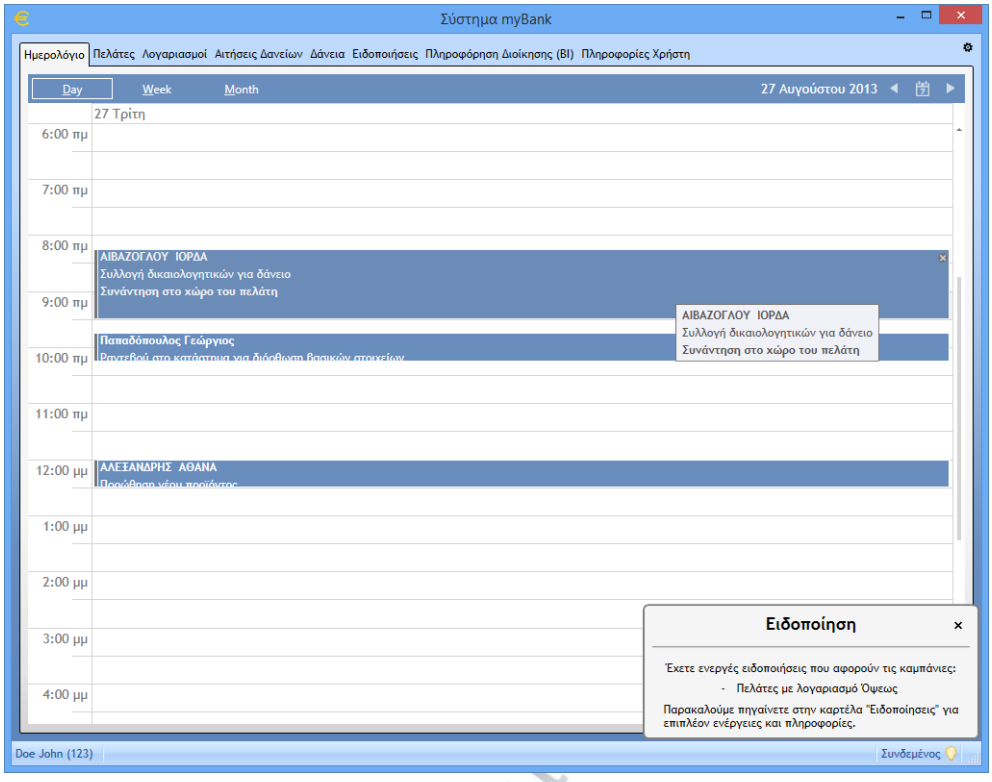
- Ενημέρωση με email
- Συνάντηση στο κατάστημα
- Τηλεφωνική επικοινωνία
- Αποστολή επιστολής
- Συνάντηση στο χώρο του πελάτη

Το ημερολόγιο του χρήστη αρχικά εμφανίζεται σε μορφή εβδομάδας, δηλαδή φαίνονται στο ημερολόγιο οι μέρες και ώρες της τρέχουσας εβδομάδας. Παρακάτω φαίνεται η αρχική οθόνη του ημερολογίου.

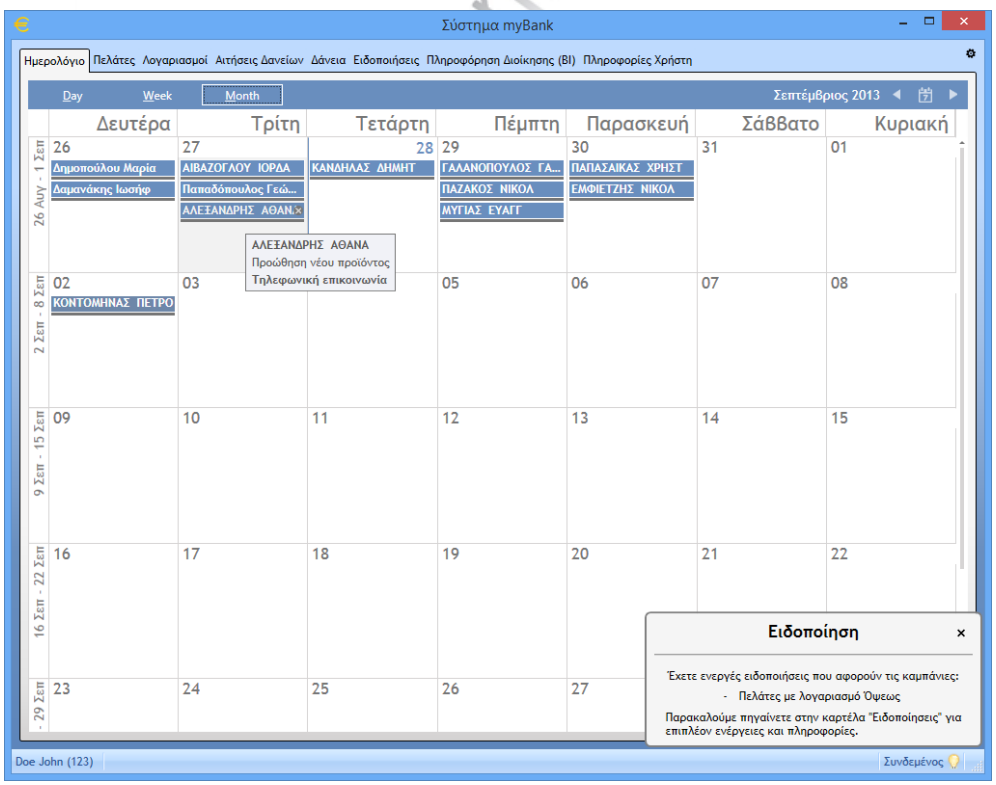


Εικόνα 2: Ημερολόγιο (Εβδομάδα)

Ο χρήστης μπορεί να επιλέξει να δει το ημερολόγιο για συγκεκριμένη μέρα ή για όλο τον μήνα, με τα κουμπιά που βρίσκονται πάνω αριστερά. Έχοντας το δείκτη του ποντικιού πάνω σε κάποιο ραντεβού – ενέργεια, ο χρήστης μπορεί να δει τα συνοπτικά στοιχεία αυτού του ραντεβού – ενέργειας. Παρακάτω φαίνεται το ημερολόγιο σε μορφή ημέρας και μήνα.



Εικόνα 3: Ημερολόγιο (Ημέρα)



Εικόνα 4: Ημερολόγιο (Μήνας)

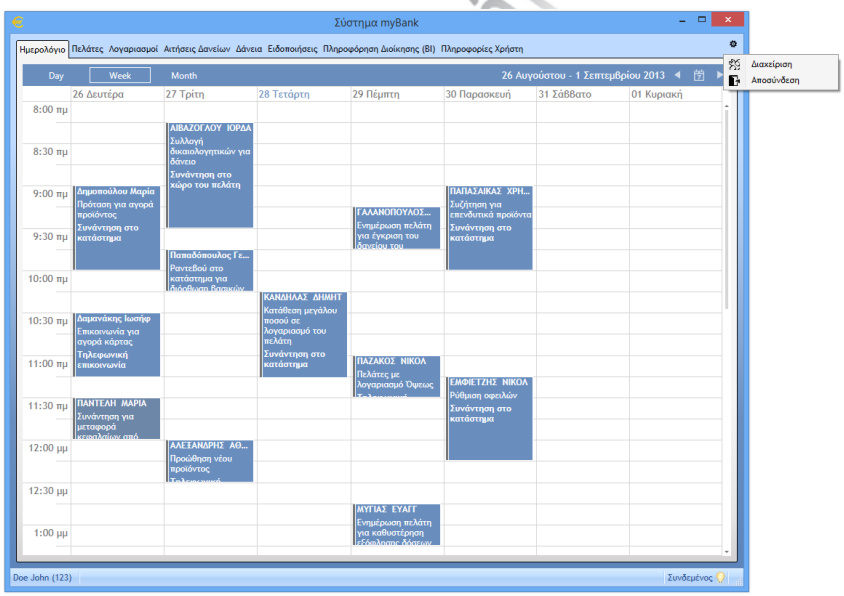
Επίσης, όπως φαίνεται στις παραπάνω οθόνες, υπάρχει μια «Ειδοποίηση» για τον χρήστη που θα πρέπει να δώσει προσοχή. Αυτή είναι μια ειδοποίηση που εμφανίζεται σε μορφή popup κατά την είσοδο στο σύστημα και πληροφορεί τον χρήστη για ενέργειες που πρέπει να κάνει. Συνήθως αυτό αφορά ειδοποιήσεις που έχει ο χρήστης για καμπάνιες που αφορούν πελάτες του καταστήματος. Ο χρήστης μπορεί να δει και να διαχειριστεί αυτές τις ειδοποιήσεις στην αντίστοιχη καρτέλα, την οποία θα δούμε παρακάτω.

Σε αυτή την αρχική οθόνη, επίσης υπάρχουν διάφορες καρτέλες (tabs) στις οποίες μπορεί να περιηγηθεί ο χρήστης για να κάνει διάφορες ενέργειες. Αυτές οι καρτέλες είναι:

- Ημερολόγιο (αρχική καρτέλα κατά την είσοδο)
- Πελάτες (Αναζήτηση – Διαχείριση πελατών)
- Λογαριασμοί (Αναζήτηση – Διαχείριση λογαριασμών)
- Αιτήσεις δανείων (Αναζήτηση – Διαχείριση αίτησης)
- Δάνεια (Αναζήτηση – Διαχείριση δανείου)
- Ειδοποιήσεις
- Πληροφόρηση Διοίκησης BI (Business Intelligence)
- Πληροφορίες Χρήστη

Ανάλογα με τις προσβάσεις που έχει ο χρήστης, έχει διαθέσιμες τις καρτέλες στις οποίες έχει πρόσβαση. Πχ. Ο Teller καταστήματος δεν έχει πρόσβαση στην καρτέλα «Πληροφόρηση Διοίκησης (BI)» γιατί αυτές αφορούν αναφορές για το σύνολο του συστήματος και αφορούν μόνο το Προσωπικό Επιχειρηματικού Χώρου (Γενικοί διευθυντές, σύμβουλοι κτλ).

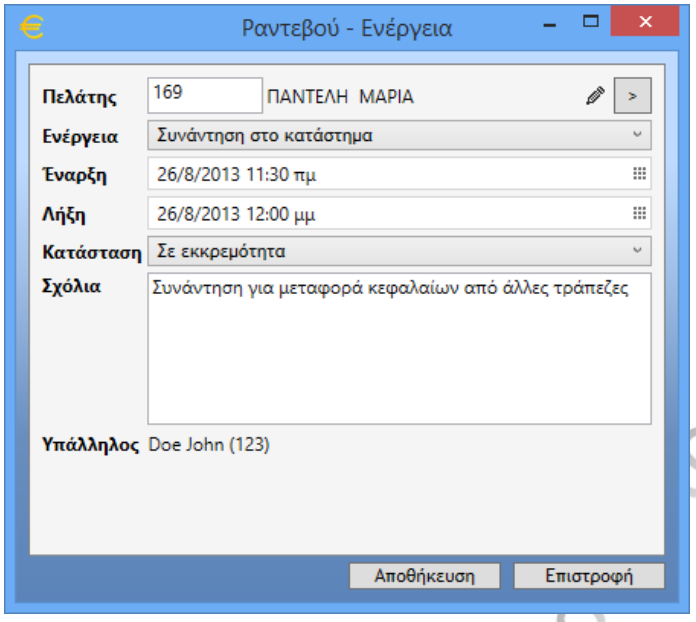
Τέλος, υπάρχει ένα μενού στο πάνω δεξιά μέρος της αρχικής οθόνης, στο οποίο ο χρήστης μπορεί να μπει στις οθόνες διαχείρισης είτε να αποσυνδεθεί. Αν ο χρήστης αποσυνδεθεί τότε η εφαρμογή πηγαίνει στην αρχική οθόνη που ζητά το μητρώο και τον κωδικό χρήστη. Αν ο χρήστης επιλέξει διαχείριση τότε η εφαρμογή πηγαίνει στην οθόνη διαχείρισης στην οποία έχει πρόσβαση μόνο ο διευθυντής καταστήματος, ο διαχειριστής χρηστών και το προσωπικό επιχειρηματικού χώρου. Τις οθόνες διαχείρισης θα τις δούμε παρακάτω. Στην παρακάτω οθόνη φαίνεται το μενού στο πάνω δεξιά μέρος της οθόνης.



Εικόνα 5: Μενού Διαχείρισης - Αποσύνδεσης

Επιστρέφοντας στην καρτέλα «Ημερολόγιο», ο χρήστης μπορεί να διαχειριστεί τα υπάρχοντα ραντεβού είτε να φτιάξει νέο, κάνοντας διπλό κλικ με το ποντίκι πάνω σε υπάρχον ραντεβού, ή πάνω σε κάποιο timeslot του ημερολόγιο για δημιουργία νέου. Τότε εμφανίζεται η οθόνη διαχείρισης ραντεβού – ενέργειας που φαίνεται στην παρακάτω οθόνη. Σε αυτή την οθόνη ο χρήστης μπορεί να αναζητήσει και να επιλέξει τον πελάτη που αφορά το ραντεβού, να

ορίσει τον τύπο ενέργειας (τους τύπους που αναφέρθηκαν παραπάνω), θα ορίσει πότε ξεκινά και πότε τελειώνει, να θέσει την κατάσταση της ενέργειας (Σε εκκρεμότητα, Αναβλήθηκε, Ακυρώθηκε, Ολοκληρώθηκε) και τέλος να βάλει κάποιο κείμενο περιγραφής. Πάνω σε αυτή την οθόνη, αν ο χρήστης έχει επιλέξει ήδη τον πελάτη, μπορεί να πατήσει το κουμπί με το μολύβι στο οποίο μπορεί να δει την οθόνη πελάτη, την οποία θα δούμε παρακάτω. Αφού ο χρήστης τελειώσει την επεξεργασία νέου ή υπάρχοντος ραντεβού – ενέργειας, μπορεί είτε να ακυρώσει τις αλλαγές που έκανε, με το κουμπί επιστροφή, είτε να αποθηκεύσει τις αλλαγές του.

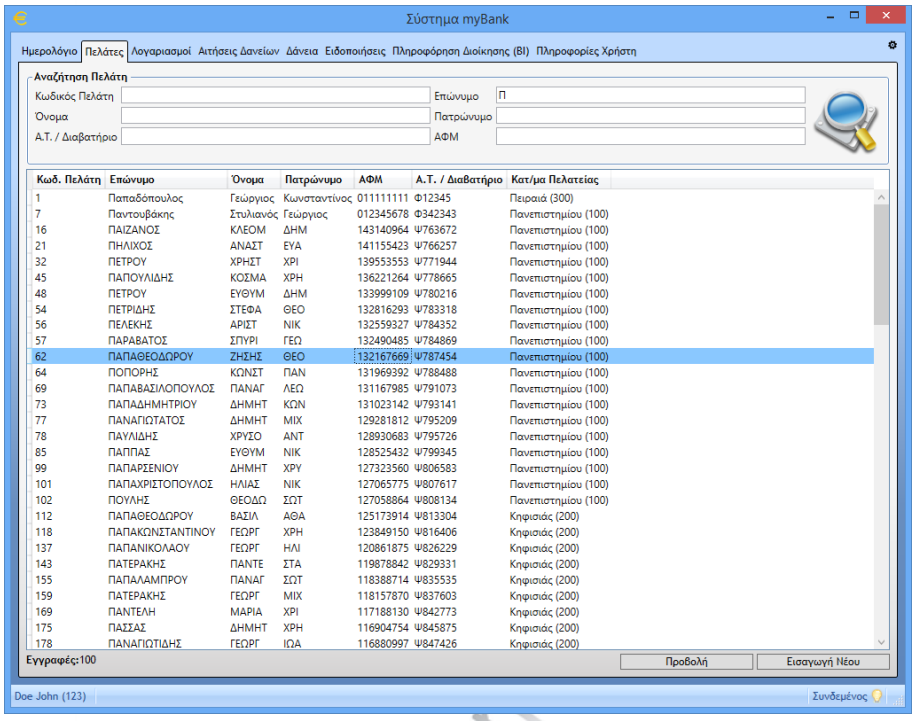


Εικόνα 6: Επεξεργασία Ραντεβού - Ενέργειας

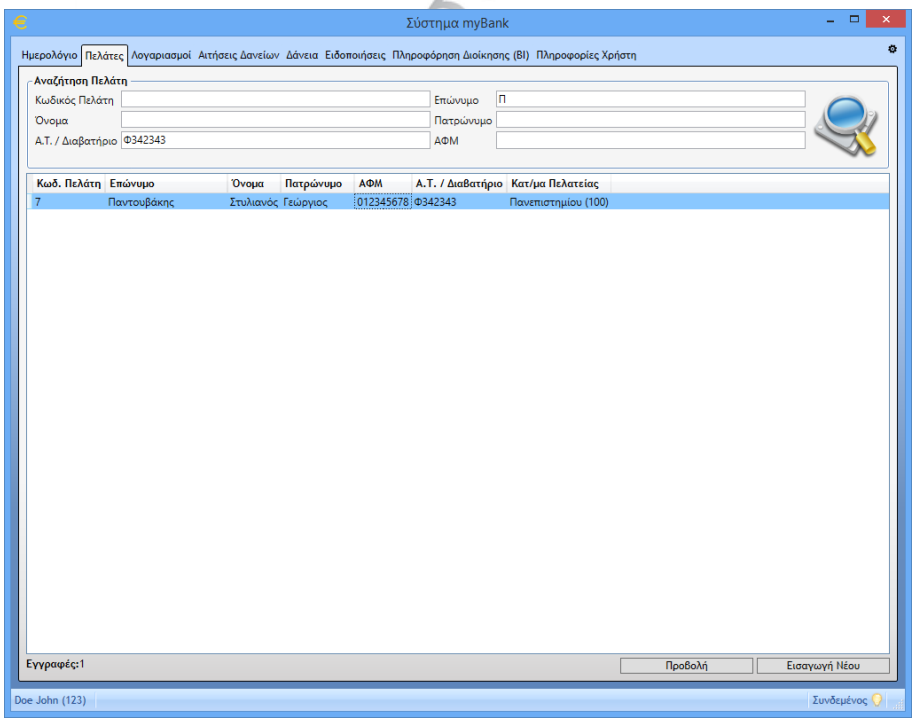
3.3 Πελάτες

3.3.1 Αναζήτηση

Η δεύτερη καρτέλα είναι η καρτέλα «Πελάτες». Είναι διαθέσιμη στους χρήστες που έχουν πρόσβαση Teller καταστήματος ή Διευθυντής καταστήματος και μπορεί να γίνει αναζήτηση πελάτη σύμφωνα με τα κριτήρια που φαίνονται στις παρακάτω οθόνες.

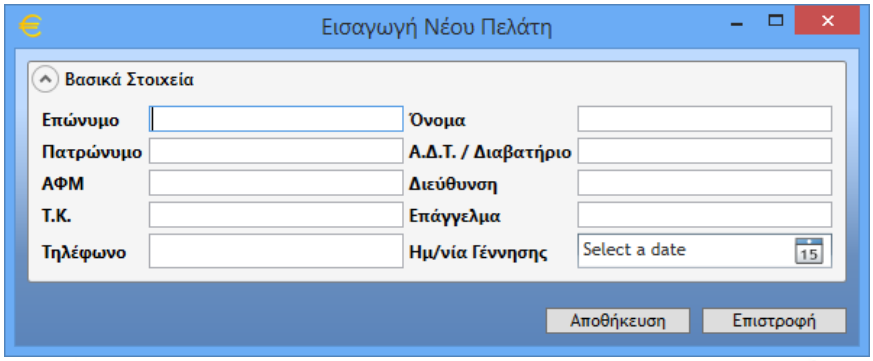


Εικόνα 7: Αναζήτηση Πελάτη με κριτήρια (1)



Εικόνα 8: Αναζήτηση Πελάτη με κριτήρια (2)

Στα αποτελέσματα αναζήτησης, ο χρήστης μπορεί να δει και να επεξεργαστεί τα στοιχεία του πελάτη είτε πατώντας το κουμπί «Προβολή», είτε πατώντας διπλό κλικ. Με το κουμπί «Εισαγωγή Νέου», μπορεί να εισάγει νέο πελάτη στο σύστημα. Παρακάτω φαίνεται η οθόνη εισαγωγής νέου πελάτη.



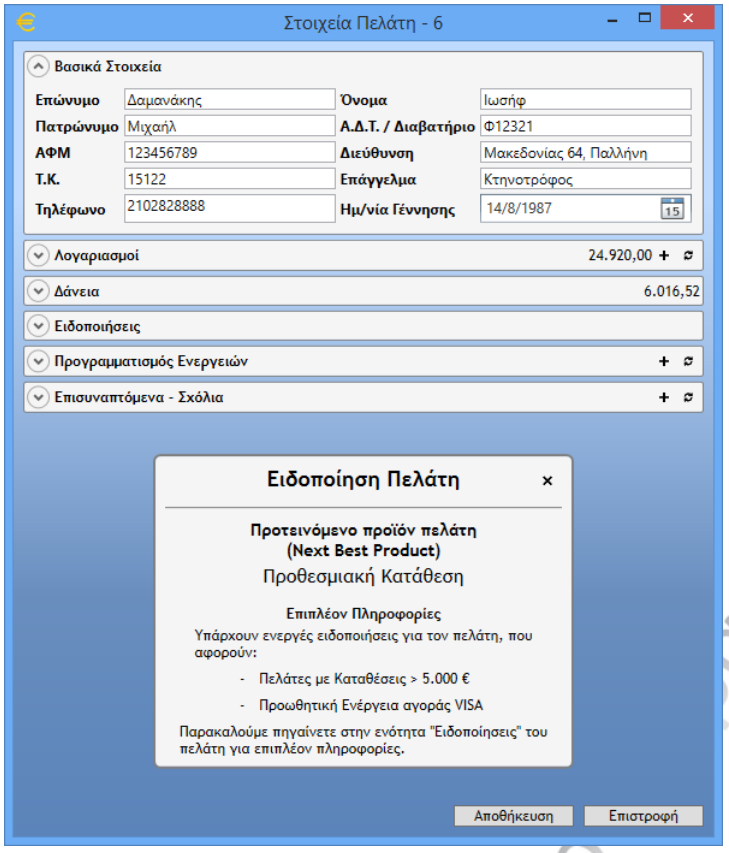
Εικόνα 9: Εισαγωγή Νέου Πελάτη

3.3.2 Οθόνη Πελάτη

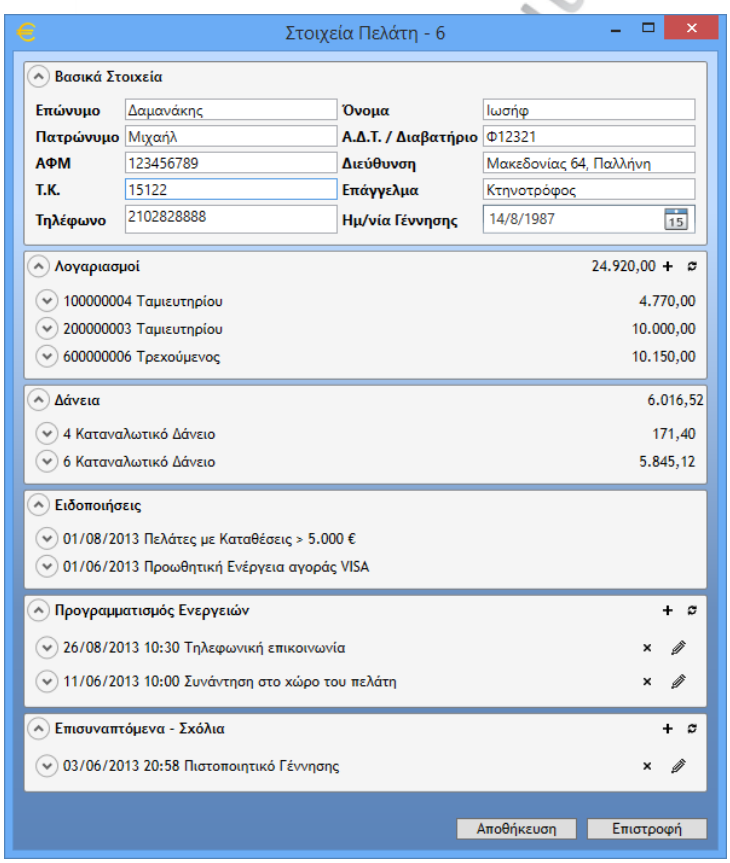
Στην οθόνη πελάτη, ο χρήστης μπορεί να δει τη συνολική εικόνα του πελάτη με όλα τα στοιχεία που υπάρχουν γι' αυτόν στο σύστημα. Αυτά είναι:

- Βασικά Στοιχεία, που αφορούν ονόματα, στοιχεία ταυτοποίησης, διεύθυνση, τηλέφωνο, επάγγελμα και ημερομηνία γέννησης.
- Λογαριασμοί πελάτη, που είναι όλοι οι λογαριασμοί καταθέσεων που έχει ο πελάτης, Τρεχούμενοι, Όψεως και ταμιευτηρίου.
- Δάνεια, που είναι όλα τα ανοιχτά δάνεια που έχει ο πελάτης στο σύστημα.
- Ειδοποιήσεις, που αφορούν ειδοποιήσεις για τον πελάτη πάνω σε καμπάνιες που έχουν σχεδιαστεί από τον επιχειρηματικό χώρο του οργανισμού.
- Προγραμματισμός Ενεργειών, που είναι όλο το ιστορικό των ραντεβού – ενεργειών που έχουν προγραμματιστεί για τον συγκεκριμένο πελάτη στο σύστημα.
- Επισυναπτόμενα – Σχόλια, που είναι αρχεία οποιαδήποτε μορφής που αφορούν τον πελάτη καθώς και κάποιο κείμενο – σχόλιο που έχει γράψει κάποιος χρήστης για τον πελάτη.

Έτσι, όταν ένας χρήστης ανοίγει την οθόνη του πελάτη, έχει την πλήρη εικόνα του, βλέποντας όλα τα προϊόντα που κατέχει ο πελάτης καθώς και το ιστορικό επικοινωνίας και ενεργειών του πελάτη. Αυτό είναι πολύ καλό για τον οργανισμό (τράπεζα) γιατί όχι μόνο καλύπτει τις ανάγκες ενός βασικού τραπεζικού συστήματος αλλά έχει και τα απαραίτητα βασικά στοιχεία ενός συστήματος CRM (Customer Relationship Management System). Με αυτό τον τρόπο, όταν ο χρήστης βρίσκεται σε επικοινωνία με τον πελάτη, μπορεί να γνωρίζει συνολικά τη σχέση του πελάτη με τον οργανισμό, πράγμα που είναι πλέον απαραίτητο για ένα σύγχρονο τραπεζικό σύστημα. Έχοντας λοιπόν, όλη αυτή την πληροφόρηση, ο πελάτης αποκτά την αίσθηση ότι ο οργανισμός, τον γνωρίζει και έχει προσωπική σχέση μαζί του. Να σημειωθεί επίσης, ότι σε οποιοδήποτε σημείο του συστήματος γίνεται αναφορά σε κάποιον πελάτη, ο χρήστης έχει τη δυνατότητα να δει κατευθείαν την οθόνη πελάτη, χωρίς να χρειαστεί να πλοηγηθεί στην οθόνη «Πελάτες». Παρακάτω φαίνεται η βασική οθόνη πελάτη.



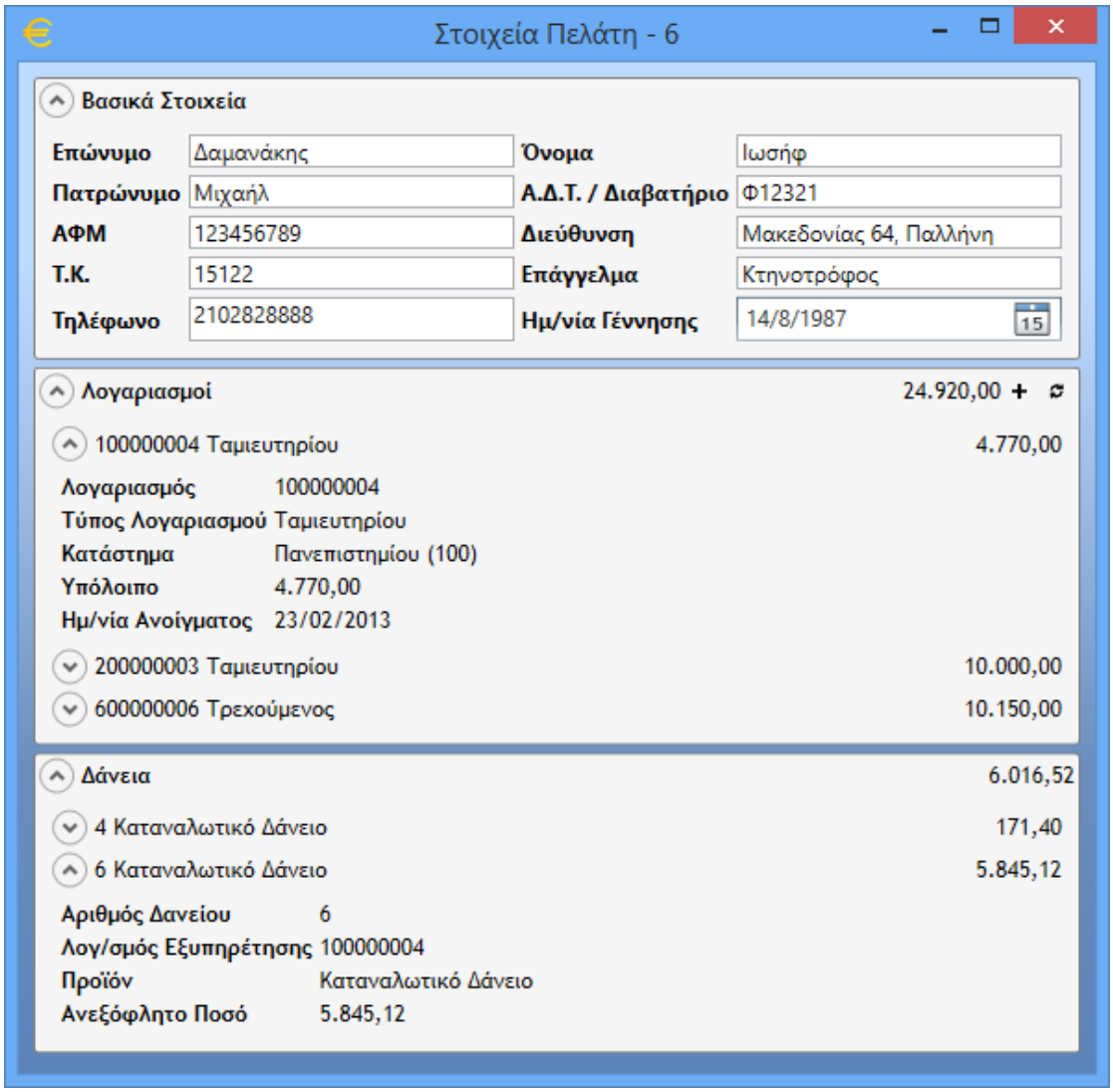
Εικόνα 10: Οθόνη Πελάτη (1)



Εικόνα 11: Οθόνη Πελάτη (2)

Στην οθόνη αυτή, ο χρήστης μπορεί πατώντας πάνω στους λογαριασμούς του πελάτη να δει όλα τα στοιχεία τους, όπως τον τύπο, το κατάστημα το υπόλοιπο και την ημερομηνία ανοίγματος. Επίσης μπορεί να ανοίξει νέο λογαριασμό για τον πελάτη.

Όπως και στους λογαριασμούς έτσι και στα δάνεια μπορεί να δει τις λεπτομέρειες τους, αλλά δεν μπορεί να ανοίξει νέο δάνειο, γιατί απαιτείται αίτηση για δάνειο και μια διαδικασία έγκρισης, την οποία θα δούμε στη συνέχεια. Παρακάτω φαίνονται οι λεπτομέρειες λογαριασμών και δανείων του πελάτη.

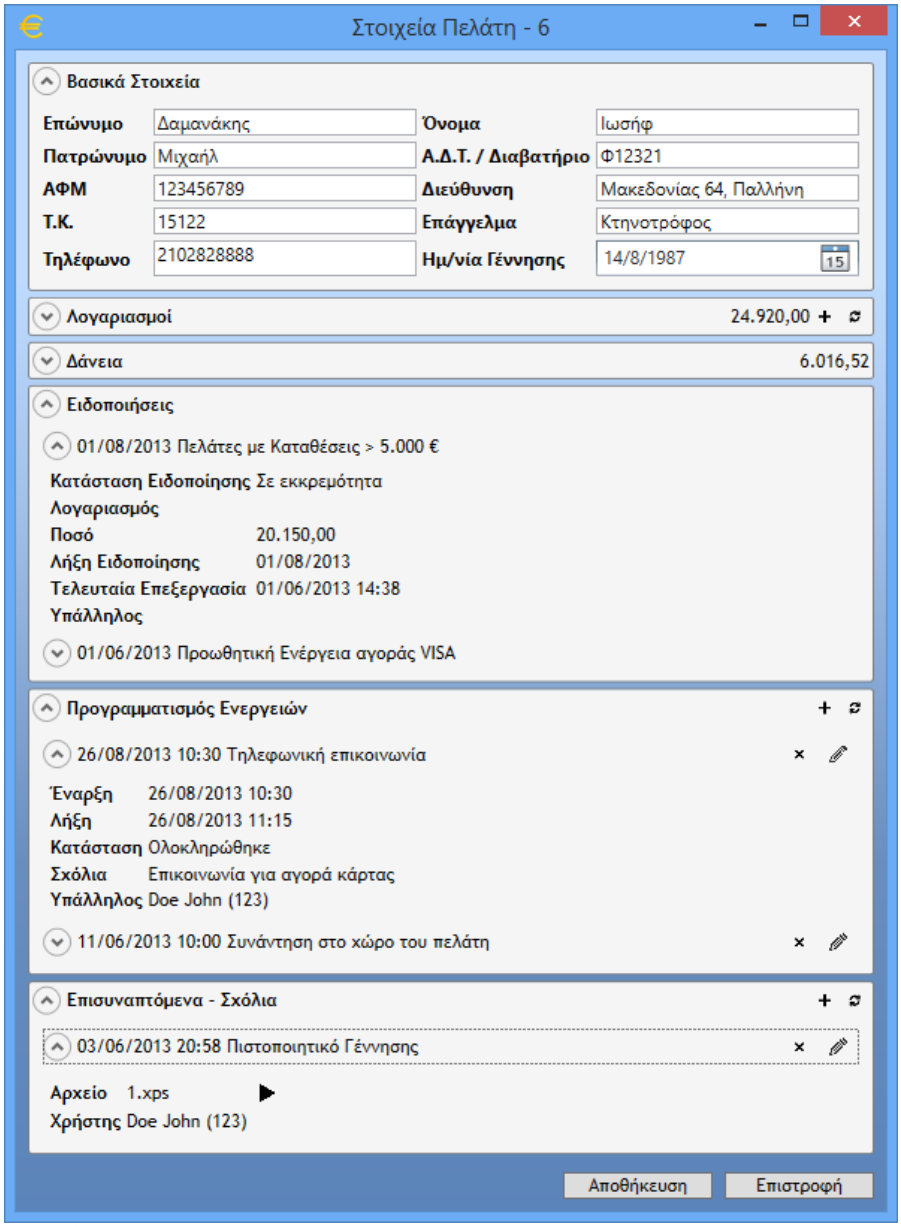


Εικόνα 12: Οθόνη Πελάτη (3)

Το ίδιο πράγμα ισχύει και για τις ειδοποιήσεις του πελάτη. Εδώ εμφανίζονται όλες οι ειδοποιήσεις του πελάτη, ακόμη κι αυτές που δεν είναι ενεργές. Η διαχείριση των ενεργών ειδοποιήσεων μπορεί να γίνει στην καρτέλα «Ειδοποιήσεις» που θα δούμε στη συνέχεια.

Στην παρακάτω οθόνη φαίνεται πως μπορούμε να δούμε όλα τα ραντεβού – ενέργειες του πελάτη. Υπάρχει δυνατότητα να γίνει είτε επεξεργασία, είτε εισαγωγή νέου. Όλα αυτά ενημερώνονται και στην καρτέλα «Ημερολόγιο» του χρήστη.

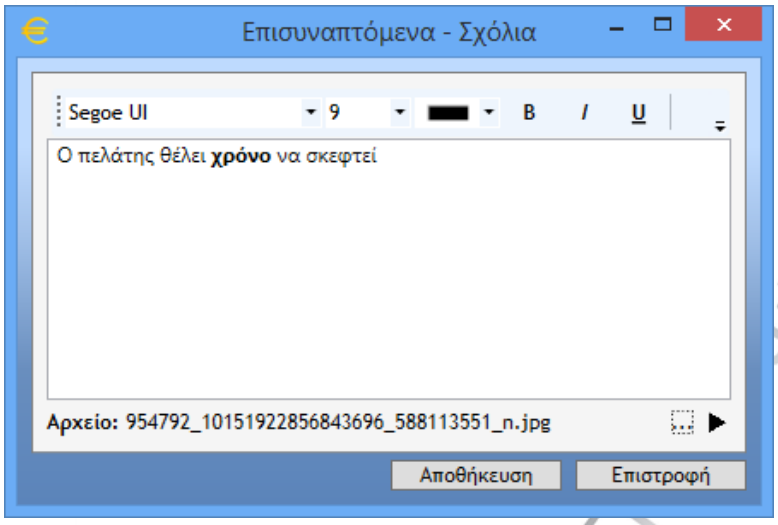
Στην ενότητα «Επισυναπτόμενα - Σχόλια», ο χρήστης μπορεί να δει όλα τα έγγραφα και σχόλια που έχουν εισαχθεί από οποιονδήποτε χρήστη που αφορούν τον πελάτη. Επίσης μπορεί να βάλει νέο επισυναπτόμενο – σχόλιο, είτε να επεξεργαστεί τα επισυναπτόμενα τα οποία έχει δημιουργήσει ο ίδιος.



Εικόνα 13: Οθόνη Πελάτη (4)

3.3.3 Επισυναπτόμενα - Σχόλια

Παρακάτω, φαίνεται η οθόνη εισαγωγής νέου επισυναπτόμενου – σχολίου, ή επεξεργασίας σχολίου. Ο χρήστης μπορεί να γράψει κάποιο κείμενο με μορφοποίηση όπως θα το έκανε και στο word (μπορεί επίσης να κάνει αντιγραφή – επικόλληση από word και να διατηρηθεί η μορφοποίησης). Τέλος, μπορεί να επιλέξει να εισάγει κάποιο έγγραφο οποιασδήποτε μορφής με το κουμπί με τις τρεις τελίτσες (ανοίγει οθόνη επιλογής αρχείου), είτε να ανοίξει ένα υπάρχον έγγραφο που υπάρχει πατώντας το κουμπί με το δεξί βελάκι, κάτω δεξιά.



Εικόνα 14: Επισυναπτόμενα – Σχόλια Επεξεργασία

3.3.4 Μηνύματα Πελάτη

Το τελευταίο πράγμα στην οθόνη του πελάτη είναι μια ειδοποίηση που αφορά τον πελάτη. Σε αυτή την ειδοποίηση, το σύστημα εμφανίζει στον υπάλληλο ένα προτεινόμενο προϊόν για τον πελάτη. Ο χρήστης, αν έχει επικοινωνία με τον πελάτη, θα πρέπει να προτείνει στον πελάτη να αγοράσει αυτό το προϊόν. Αυτό είναι γνωστό και ως «Next Best Product» πράγμα που το έχουν πολλά συστήματα. Αυτό το προϊόν βγαίνει από κάποιους κανόνες που έχουν ορισθεί από το σύστημα, σύμφωνα με την πολιτική του οργανισμού και κατ' επέκταση την τάση της αγοράς. Αυτό φυσικά παίρνει ως παραμέτρους τα ήδη υπάρχοντα προϊόντα που έχει ο πελάτης. Συγκεκριμένα, το «Next Best Product» βγαίνει σύμφωνα με τον παρακάτω κανόνα:

- Αν ο πελάτης έχει συνολικό υπόλοιπο καταθέσεων μεγαλύτερο ή ίσο των 20.000 € τότε προτείνεται Προθεσμιακή Κατάθεση.
- Αν δεν ισχύουν τα παραπάνω και ο πελάτης δεν έχει κάποιο δάνειο, ούτε κάποιο λογαριασμό καταθέσεων, τότε προτείνεται «Λογαριασμός Ταμιευτηρίου».
- Αν δεν ισχύουν τα παραπάνω και ο πελάτης έχει κάποιο δανειακό προϊόν τότε σύμφωνα με αυτό το προϊόν του προτείνεται νέο προϊόν σύμφωνα με το παρακάτω πινακάκι:

Δανειακό Προϊόν	Προτεινόμενο Προϊόν
Καταναλωτικό Δάνειο	Πιστωτική κάρτα
Στεγαστικό Δάνειο	Ασφάλεια Σπιτιού
Δάνειο Για Μικρές Επιχειρήσεις	Εταιρική Πιστωτική Κάρτα
Δάνειο Μεγάλων Επιχειρήσεων	Ασφάλεια Εγκαταστάσεων

Εικόνα 15: Πίνακας αντιστοίχισης Δανειακών Προϊόντων με Προτεινόμενο Προϊόν

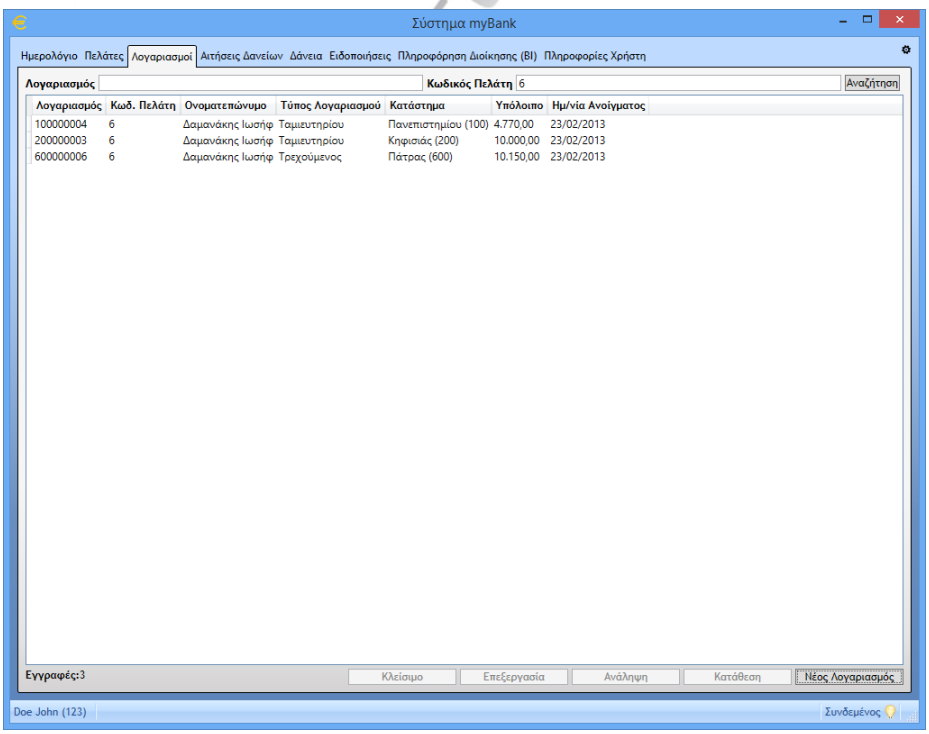
- Τέλος, αν δεν ισχύει τίποτα από τα παραπάνω, τότε προτείνεται καταναλωτικό δάνειο.

Πέρα από το «Next Best Product», αν ο πελάτης έχει ενεργές ειδοποιήσεις σε κάποια καμπάνια, τότε εμφανίζονται οι καμπάνιες στις οποίες συμμετέχει και το σύστημα προτρέπει τον χρήστη να πάει στην καρτέλα «Ειδοποιήσεις» για λεπτομέρειες. Παρακάτω φαίνεται η ειδοποίηση που εμφανίζεται, όπως είναι στην οθόνη πελάτη.

Εικόνα 16: Μηνύματα Πελάτη

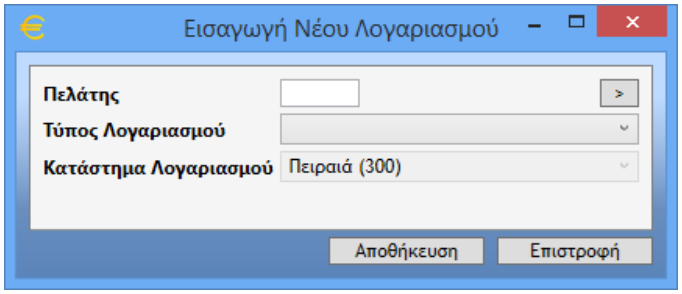
3.4 Λογαριασμοί

Η τρίτη καρτέλα είναι η καρτέλα «Λογαριασμοί». Είναι διαθέσιμη στους χρήστες που έχουν πρόσβαση Teller καταστήματος ή Διευθυντής καταστήματος και μπορεί να γίνει αναζήτηση πελάτη σύμφωνα με τον αριθμό λογαριασμού ή τον κωδικό πελάτη όπως φαίνεται στην παρακάτω οθόνη. Ο χρήστης μπορεί να κάνει τις ίδιες λειτουργίες όπως και στην οθόνη πελάτη (άνοιγμα νέου, προβολή στοιχείων λογαριασμών). Επίσης μπορεί να κάνει και κλείσιμο λογαριασμού (απαιτεί τακτοποίηση του υπολοίπου, δηλαδή αν είναι αρνητικό να εξοφληθεί και αν είναι θετικό να γίνει ανάληψη όλου του υπολοίπου).

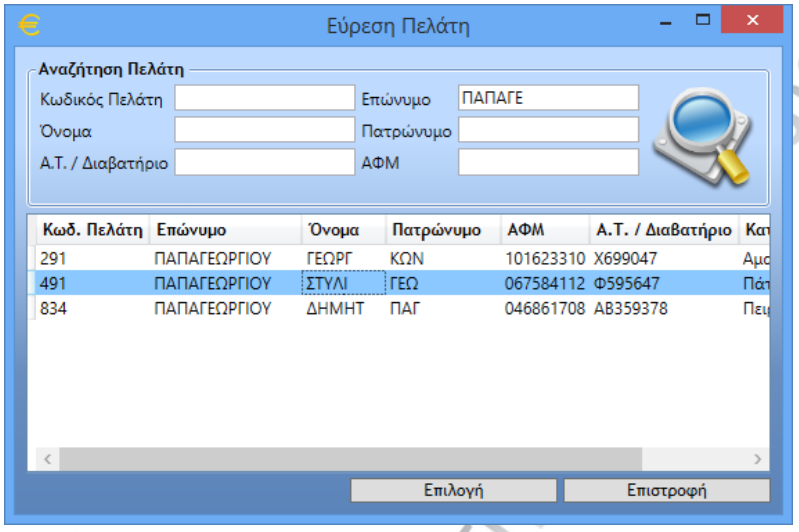


Εικόνα 17: Αναζήτηση- Διαχείριση Λογαριασμών

Η οθόνη εισαγωγής λογαριασμού φαίνεται παρακάτω, όπως και η οθόνη εύρεσης και επιλογής πελάτη. Όπως βλέπουμε, το πεδίο «Κατάστημα Λογαριασμού» είναι κλειδωμένο γιατί το άνοιγμα λογαριασμού μπορεί να γίνει μόνο στο κατάστημα στο οποίο ανήκει ο χρήστης.

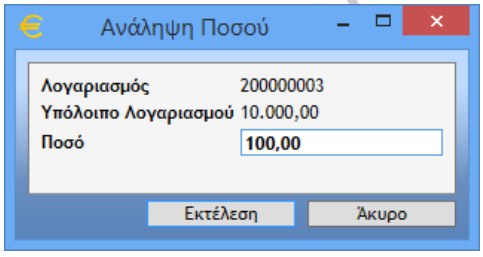


Εικόνα 18: Εισαγωγή Νέου Λογαριασμού

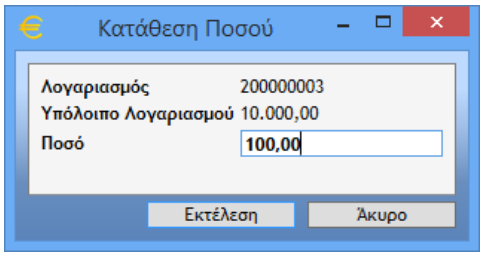


Εικόνα 19: Επιλογή Πελάτη

Παρακάτω φαίνονται οι οθόνες ανάληψης και κατάθεσης. Για να γίνει ανάληψη ποσού μεγαλύτερου του υπολοίπου θα πρέπει ο τύπος λογαριασμού να έχει ποσό υπερανάληψης, διαφορετικά εμφανίζεται σχετικό μήνυμα (μόνο οι τρεχούμενοι λογαριασμοί έχουν ποσό υπερανάληψης).



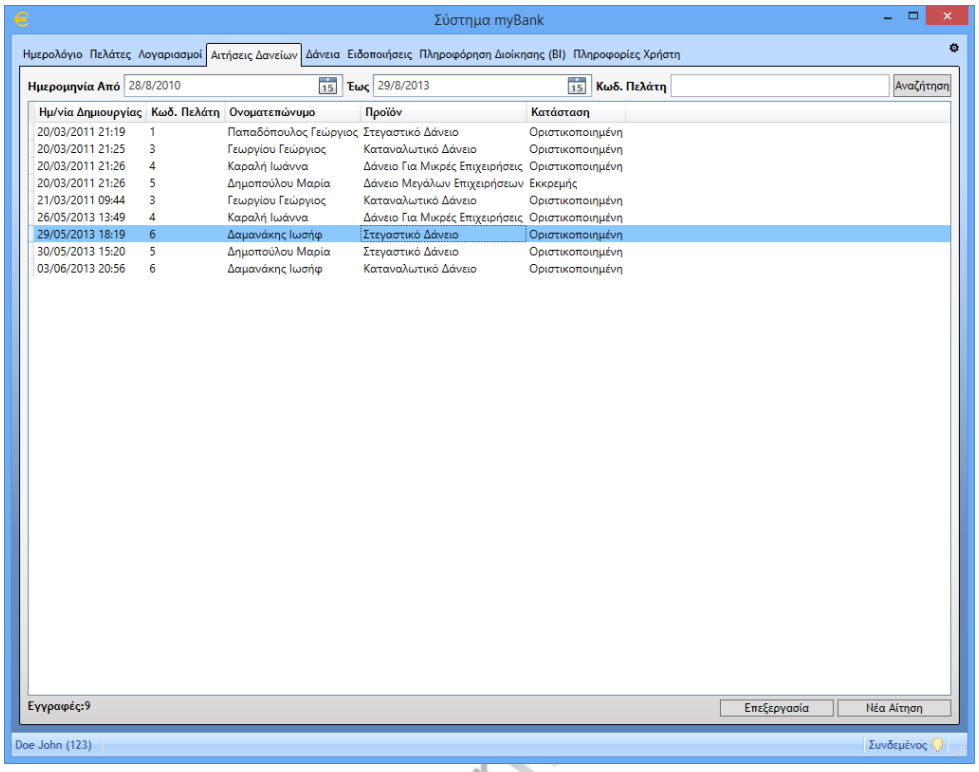
Εικόνα 20: Ανάληψη Ποσού από λογαριασμό



Εικόνα 21: Κατάθεση Ποσού σε λογαριασμό

3.5 Αιτήσεις Δανείων

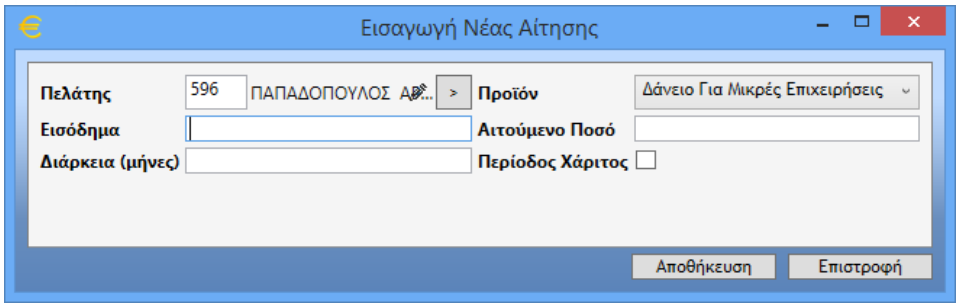
Η τέταρτη καρτέλα είναι η καρτέλα «Αιτήσεις Δανείων». Είναι διαθέσιμη στους χρήστες που έχουν πρόσβαση Teller καταστήματος ή Εγκριτής Δανείων και μπορεί να γίνει αναζήτηση με την ημερομηνία δημιουργίας του αιτήματος ή και τον κωδικό πελάτη, όπως φαίνεται στην παρακάτω οθόνη.



Εικόνα 22: Αναζήτηση Αιτήσεων Δανείων

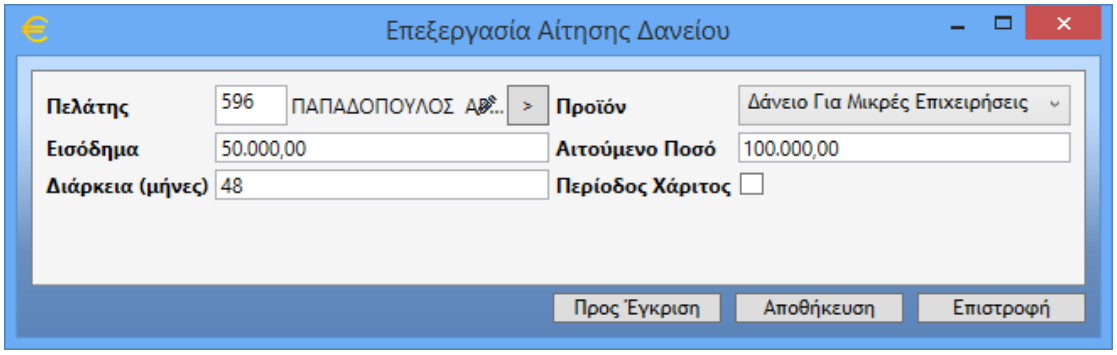
Οι αιτήσεις έχουν διάφορες καταστάσεις (status) και ανάλογα με την πρόσβαση του χρήστη μπορεί να κάνει και τις ανάλογες ενέργειες. Ο Teller καταστήματος μπορεί να εισάγει μια νέα αίτηση η οποία παίρνει την αρχική κατάσταση «Εκκρεμής». Στη συνέχεια πρέπει να συμπληρώσει διάφορα πεδία που είναι απαραίτητα για να κριθεί η πιστοληπτική ικανότητα του πελάτη, και ο εγκριτής δανείων στη συνέχεια να εγκρίνει. Αυτά τα πεδία φαίνονται στην παρακάτω οθόνη και είναι:

- Πελάτης
- Προϊόν που θέλει να αγοράσει
- Ετήσιο εισόδημα πελάτη
- Αιτούμενο ποσό δανείου
- Διάρκεια σε μήνες
- Ένδειξη περιόδου χάριτος



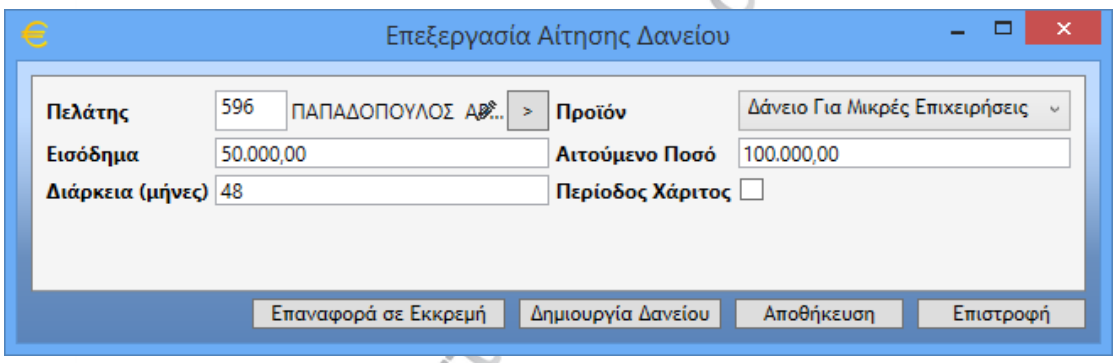
Εικόνα 23: Εισαγωγή Νέας Αίτησης Δανείου

Αφού ο Teller καταστήματος εισάγει την αίτηση ως εκκρεμή στο σύστημα και σιγουρευτεί ότι όλα τα στοιχεία είναι σωστά, τότε μπορεί να στείλει την αίτηση προς έγκριση, όπως φαίνεται στην παρακάτω οθόνη.



Εικόνα 24: Επεξεργασία Αίτησης Δανείου – Αποστολή Προς Έγκριση

Ο Εγκριτής δανείων με τη σειρά του, μπορεί να δει τα στοιχεία των αιτήσεων προς έγκριση και να αποφασίσει αν θα την εγκρίνει η όχι. Αν ο εγκριτής κάνει την έγκριση (το αίτημα είναι σε κατάσταση «Εγκεκριμένο»), τότε θα πρέπει να γίνει δημιουργία δανείου με το αντίστοιχο κουμπί που φαίνεται στην παρακάτω οθόνη.

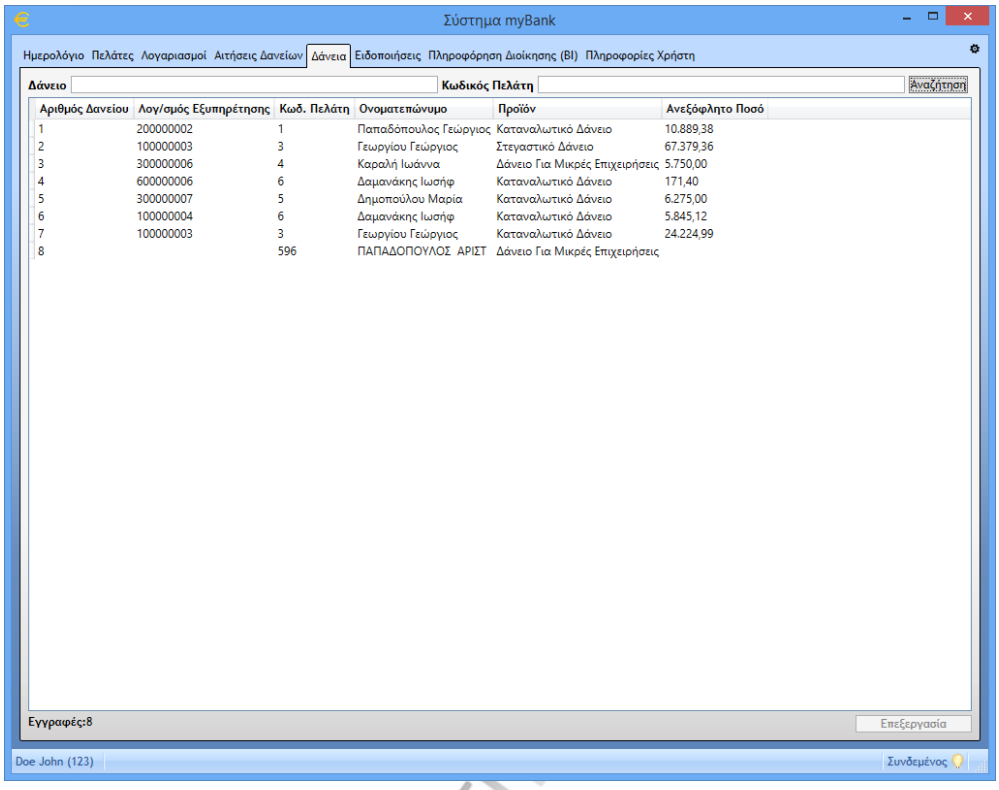


Εικόνα 25: Επεξεργασία Αίτησης Δανείου – Έγκριση – Δημιουργία Δανείου

Εδώ τελειώνει ο κύκλος της αίτησης δανείου και ήδη έχει δημιουργηθεί ένα μη εκταμιευμένο δάνειο για τον πελάτη με τα στοιχεία που είχαν εισαχθεί στην αίτηση.

3.6 Δάνεια

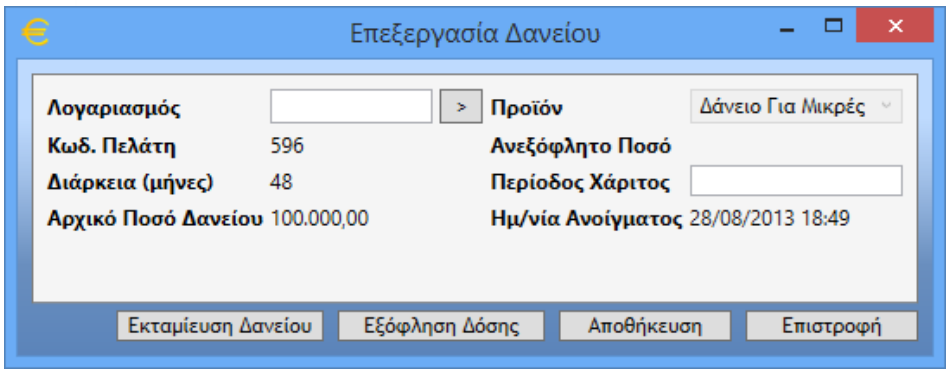
Η πέμπτη καρτέλα λοιπόν, είναι η καρτέλα «Δάνεια». Αυτή η καρτέλα είναι διαθέσιμη μόνο στους χρήστες που έχουν πρόσβαση Teller καταστήματος ή Διευθυντής καταστήματος. Εδώ μπορεί να γίνει αναζήτηση δανείου σύμφωνα με τον αριθμό δανείου ή τον κωδικό πελάτη. Αυτή η οθόνη φαίνεται παρακάτω.



Εικόνα 26: Αναζήτηση Δανείων

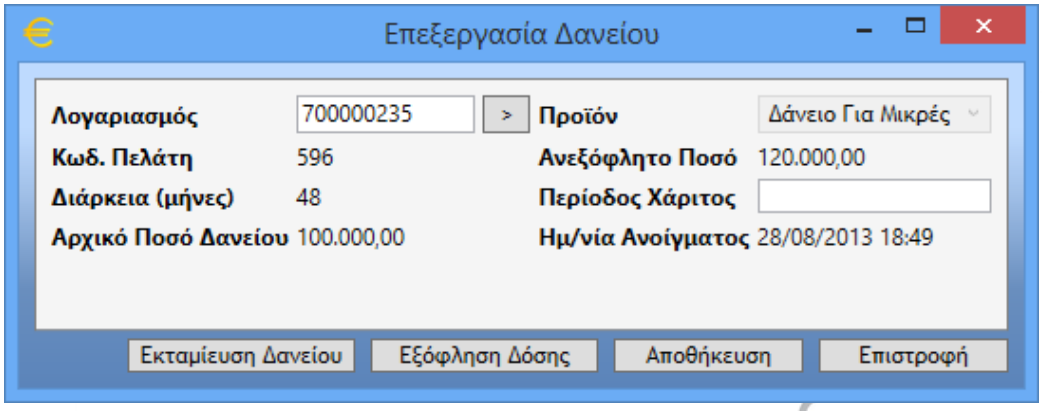
Ο χρήστης μπορεί λοιπόν να αναζητήσει δάνεια και να κάνει επεξεργασία τους. Όταν ένα δάνειο έχει μόλις δημιουργηθεί με την διαδικασία των αιτήσεων, τότε ο χρήστης θα πρέπει να πάει στο δάνειο και να το συνδέσει με κάποιον λογαριασμό (λογαριασμός εξυπηρέτησης), να εισάγει την πιθανή διάρκεια περιόδου χάριτος (σε μήνες) και τέλος να κάνει εκταμίευση του δανείου.

Τη στιγμή που ο χρήστης κάνει την εκταμίευση του δανείου, τότε το αιτούμενο ποσό δανείου που είχε εγκριθεί για τον πελάτη, εκταμιεύεται στον λογαριασμό εξυπηρέτησης που έχει οριστεί, οπότε το ποσό είναι πλέον διαθέσιμο στον πελάτη προς κατανάλωση. Εκείνη τη στιγμή λοιπόν γίνεται και ο υπολογισμός του δοσολογίου του δανείου και ξεκινούν να μετράνε οι τόκοι, σύμφωνα με το επιτόκιο του προϊόντος και το ποσό δανείου.

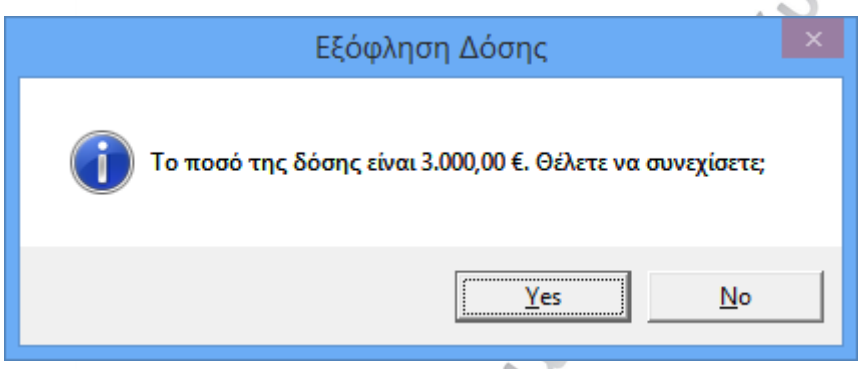


Εικόνα 27: Επεξεργασία Δανείου (1) – Σύνδεση με λογαριασμό

Αφότου έχει γίνει λοιπόν η εκταμίευση, ο χρήστης μπορεί να κάνει εξόφληση δόσης όπως φαίνεται στις παρακάτω οθόνες. Ο χρήστης θα πρέπει να εισπράξει το ποσό που αναλογεί στη δόση από τον πελάτη, το οποίο εμφανίζεται από το σύστημα.



Εικόνα 28: Επεξεργασία Δανείου (2) – Εκταμίευση, Εξόφληση Δόσης



Εικόνα 29: Εξόφληση Δόσης Δανείου – Επιβεβαιωτικό Μήνυμα

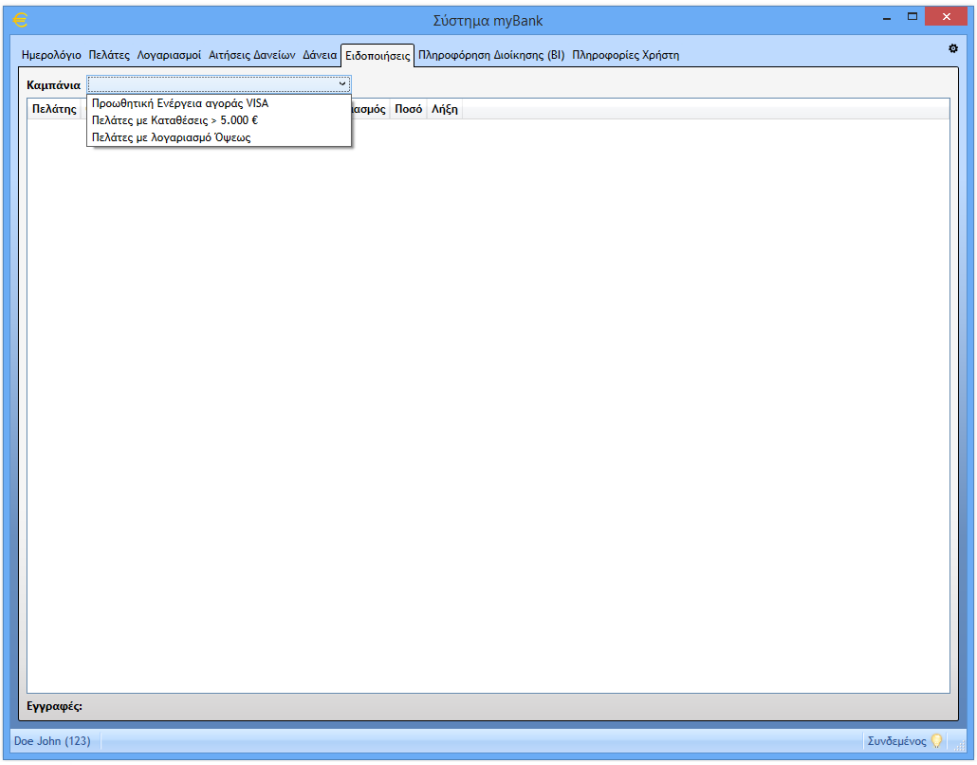
3.7 Ειδοποιήσεις

Η έκτη καρτέλα, είναι η καρτέλα «Ειδοποιήσεις». Η καρτέλα ειδοποιήσεις είναι διαθέσιμη στους χρήστες με πρόσβαση Teller καταστήματος ή Διευθυντής καταστήματος (χρήστες καταστήματος, αφού αυτοί διαχειρίζονται την επαφή του οργανισμού με τον πελάτη).

Σε αυτή την καρτέλα, ο χρήστης μπορεί να δει όλες τις ενεργές ειδοποιήσεις που υπάρχουν για πελάτες του καταστήματος. Να σημειώσουμε ότι ο κάθε πελάτης έχει ένα κατάστημα πελατείας, που ουσιαστικά είναι το κατάστημα στο οποίοι δημιουργήθηκε ο πελάτης και θεωρούμε πως είναι το κατάστημα που ο πελάτης επιθυμεί να εξυπηρετείται (για διάφορους λόγους, είτε είναι κοντά στην οικία του, είτε στην εργασία του κτλ). Αυτό υπάρχει ώστε να μπορούν να μοιράζεται στα καταστήματα η διαχείριση της σχέσης του οργανισμού με τον πελάτη.

Αρχικά λοιπόν, ο χρήστης επιλέγει μία από τις διαθέσιμες ενεργές καμπάνιες και του εμφανίζονται όλες οι ειδοποιήσεις που αφορούν αυτή την καμπάνια για πελάτες του καταστήματος του χρήστη. Αυτές οι καμπάνιες έχουν σχεδιαστεί από τον επιχειρηματικό χώρο και αφορούν διάφορες ενέργειες που έχει αποφασίσει η διοίκηση να κάνει σχετικά με τους πελάτες.

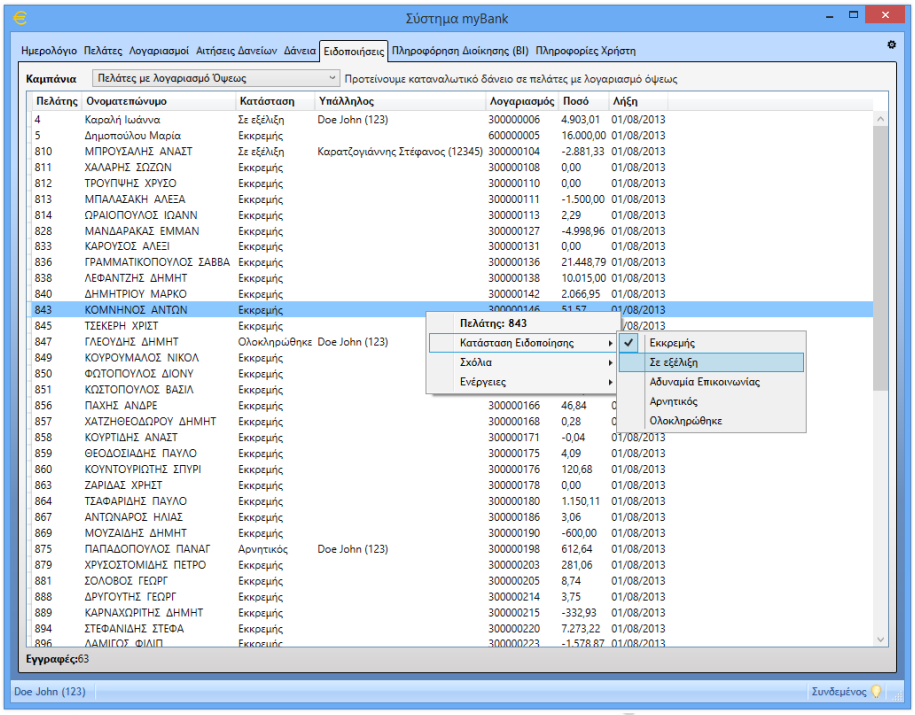
Όταν επιλέγεται μία καμπάνια, εμφανίζεται ένα μήνυμα – οδηγία προς τον χρήστη, που έχει μια σύντομη περιγραφή της καμπάνιας. Αυτή η περιγραφεί είτε έχει μια σύντομη περιγραφή της καμπάνιας, σε τι αποσκοπεί και τι πρέπει να κάνουν οι υπάλληλοι καταστημάτων για τις ειδοποιήσεις. Συνήθως αυτές οι ειδοποιήσεις (και κατ' επέκταση καμπάνιες) αφορούν προώθηση προϊόντων, στοχευμένα σε συγκεκριμένες ομάδες πελατών, με διάφορα κριτήρια που θα δούμε στη συνέχεια. Επίσης θα μπορούσε μια καμπάνια να αφορά μια εκκρεμότητα που έχουν κάποιοι πελάτες απέναντι στο οργανισμό (πχ. Ρύθμιση ληξιπρόθεσμων οφειλών, απαραίτητα δικαιολογητικά). Παρακάτω φαίνεται η αρχική οθόνη ειδοποιήσεων.



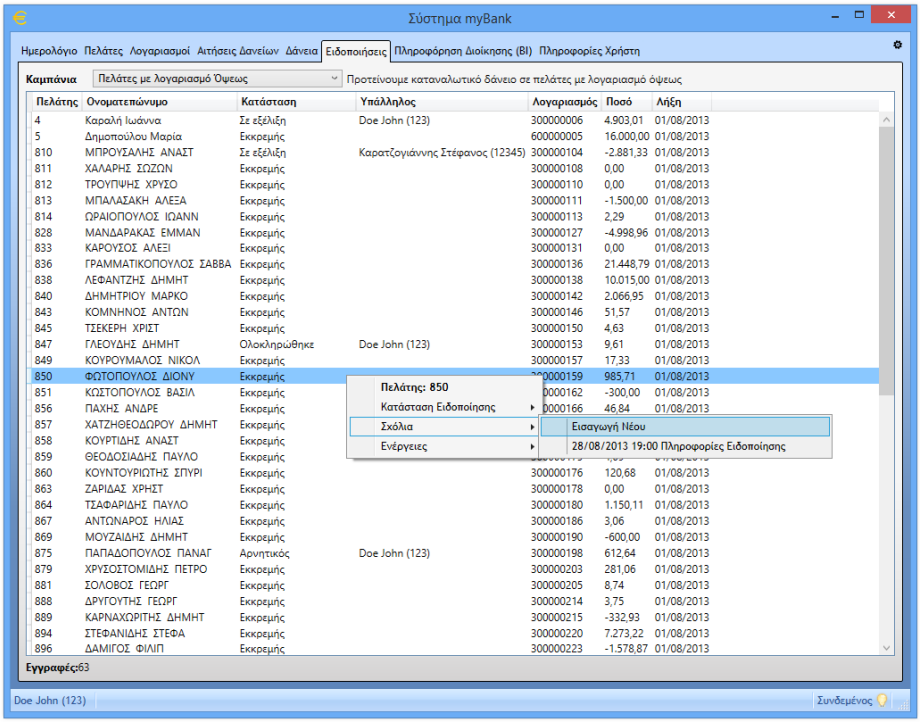
Εικόνα 30: Ειδοποιήσεις (1)

Παρακάτω φαίνεται η λίστα με τις ενεργές ειδοποίησης της επιλεγμένης καμπάνιας, για το κατάστημα του χρήστη. Με δεξί κλικ μπορεί να γίνει επεξεργασία της ειδοποίησης. Ο χρήστης μπορεί από εκεί να πάει κατευθείαν στην οθόνη πελάτη, να αλλάξει την κατάσταση της ειδοποίησης σύμφωνα με την πρόοδο επικοινωνίας που είχε με τον πελάτη, να επισυνάψει

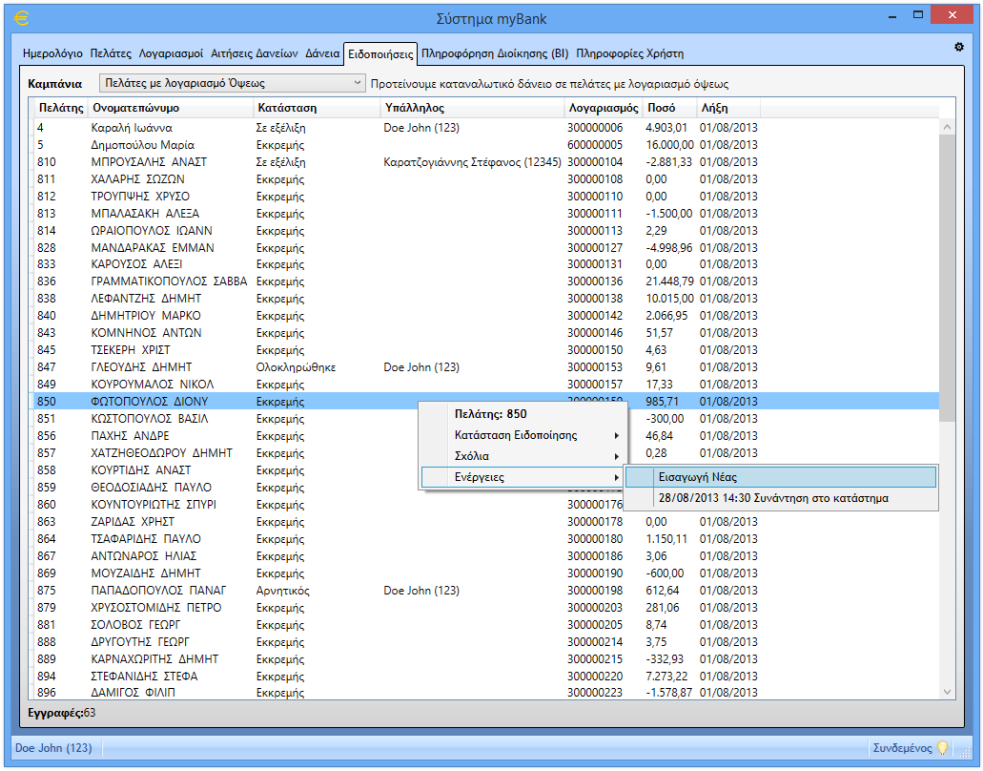
κάποιο σχόλιο στην ειδοποίηση που αφορά την πρόοδό της και να προγραμματίσει κάποια ενέργεια σχετικά με αυτή την ειδοποίηση για τον πελάτη, η οποία θα φανεί και στο ημερολόγιο. Στις παρακάτω οθόνες, φαίνονται αυτές οι βασικές λειτουργίες επεξεργασίας μιας ειδοποίησης.



Εικόνα 31: Ειδοποιήσεις (2) – Αλλαγή Κατάστασης Ειδοποίησης



Εικόνα 32: Ειδοποιήσεις (3) – Εισαγωγή, Επεξεργασία Σχολίου



Εικόνα 33: Ειδοποιήσεις (4) – Εισαγωγή, Επεξεργασία Ενέργειας

3.8 Πληροφόρηση Διοίκησης (BI)

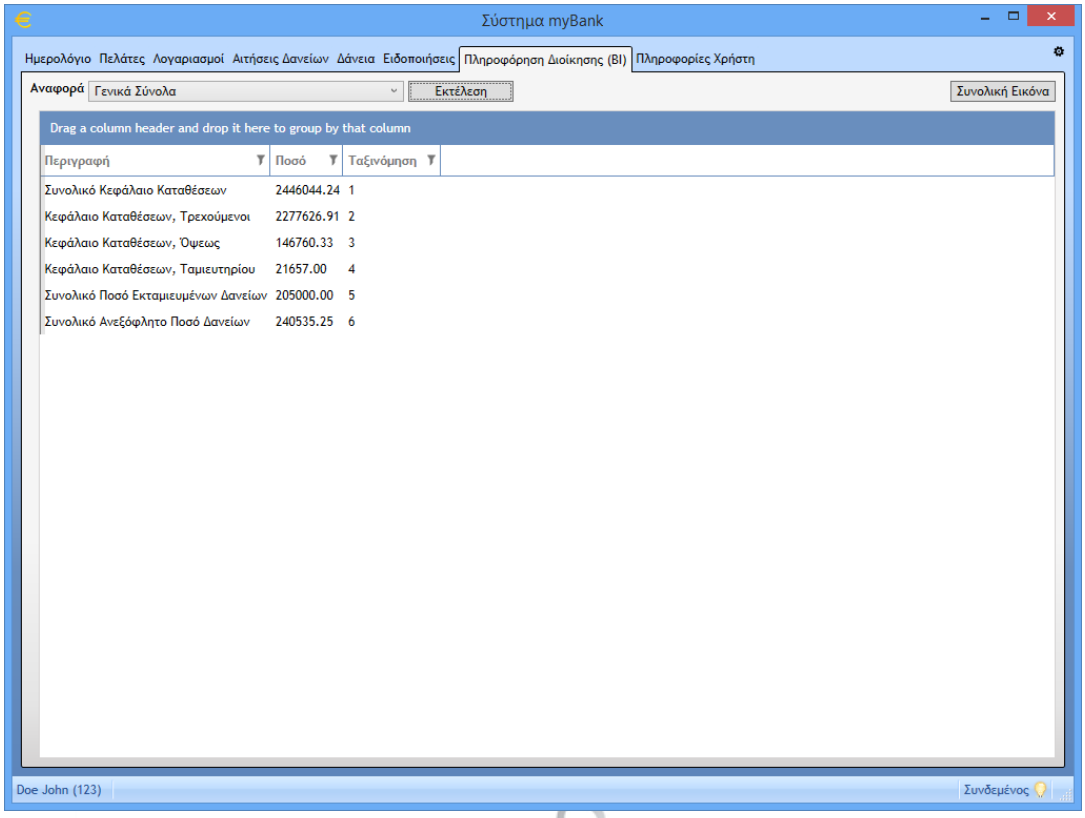
Η έβδομη καρτέλα, είναι η καρτέλα «Πληροφόρηση Διοίκησης (BI)». Αυτή ο οθόνη είναι διαθέσιμοι μόνο στους χρήστες με πρόσβαση «Προσωπικό Επιχειρηματικού Χώρου». Είναι αναφορές που αφορούν τη συνολική εικόνα – κατάσταση του οργανισμού.

Αρχικά ο χρήστης πρέπει να επιλέξει μία από τις διαθέσιμες αναφορές που υπάρχουν. Αυτές είναι:

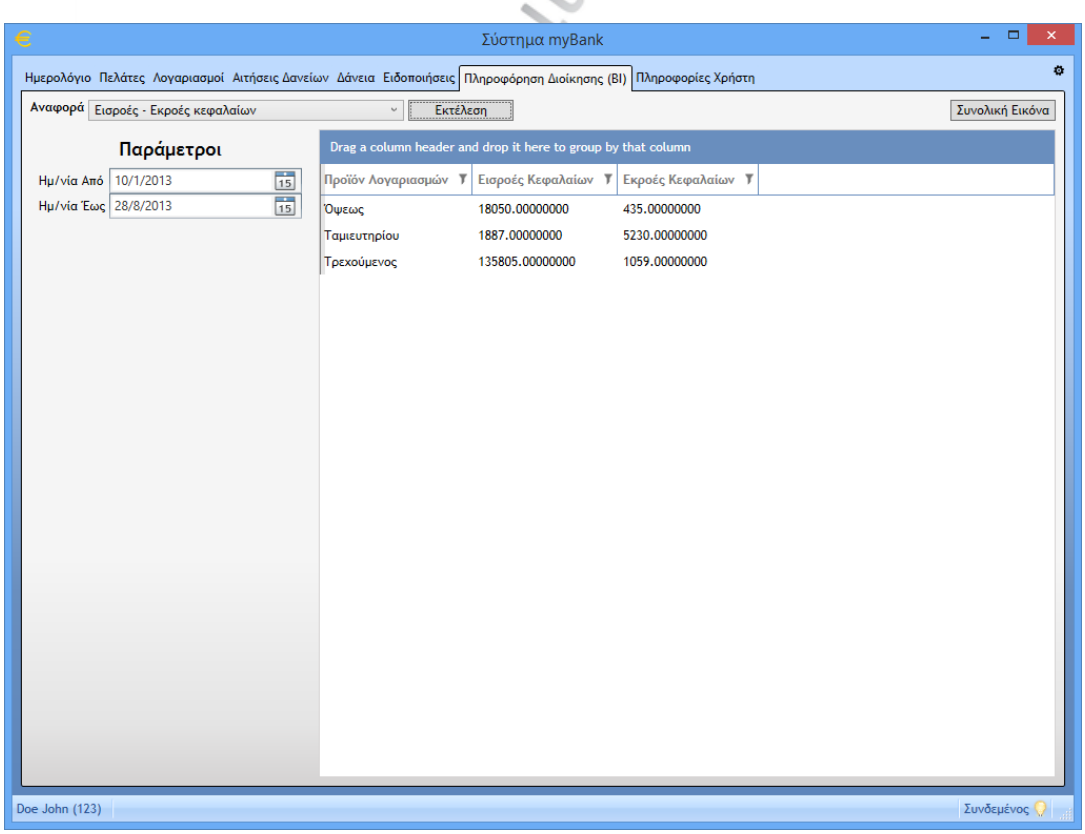
- Γενικά σύνολα, που αφορά τα συνολικά ποσά καταθετικών και δανειακών προϊόντων του οργανισμού.
- Καταθέσεις ανά Κατάστημα, που αφορά την κατανομή των διαθέσιμων κεφαλαίων σε καταθέσεις που έχει ο οργανισμός, σε καταστήματα.
- Δάνεια ανά Κατάστημα, που αφορά την κατανομή δανείων σε καταστήματα
- Εισροές – Εκροές κεφαλαίων (για συγκεκριμένο διάστημα).
- Ενημέρωση Ειδοποιήσεων, που αφορά την κατάσταση ειδοποιήσεων για συγκεκριμένη καμπάνια, ώστε η διοίκηση να μπορεί να γνωρίζει ποια είναι η πρόοδος των καμπανιών που έχει σχεδιάσει.

Αφού ο χρήστης επιλέξει κάποια αναφορά, πατάει το κουμπί εκτέλεση για να του εμφανιστούν τα αποτελέσματα. Αν η αναφορά έχει παραμέτρους, ο χρήστης καλείται να συμπληρώσει αυτές τις παραμέτρους.

Παρακάτω φαίνονται δύο οθόνες από την καρτέλα «Πληροφόρηση Διοίκησης (BI)».



Εικόνα 34: Πληροφόρηση Διοίκησης – Αναφορές (1)

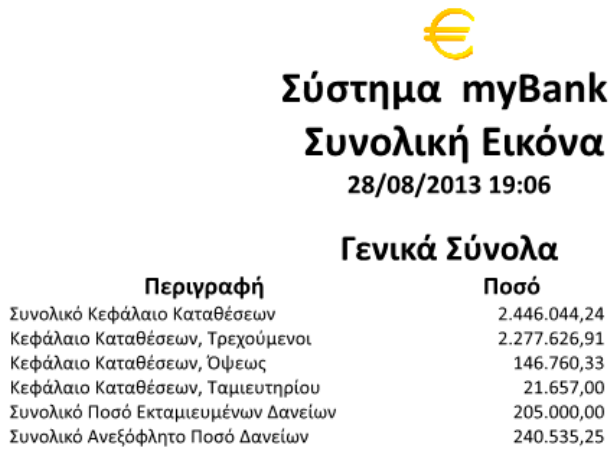


Εικόνα 35: Πληροφόρηση Διοίκησης – Αναφορές (2)

Τέλος, σε αυτή την καρτέλα, ο χρήστης μπορεί να πατήσει το κουμπί συνολική εικόνα και γίνει παραγωγή εκτυπώσιμης αναφοράς για τη συνολική εικόνα του συστήματος. Αυτή η εκτυπώσιμη αναφορά, περιέχει:

- Διαθέσιμα κεφάλαια
- Δάνεια
- Κατανομή κεφαλαίων ανά κατάστημα
- Εισροές εκροές

Η πρώτη σελίδα αυτής της αναφοράς φαίνεται παρακάτω.



Σύστημα myBank

Συνολική Εικόνα

28/08/2013 19:06

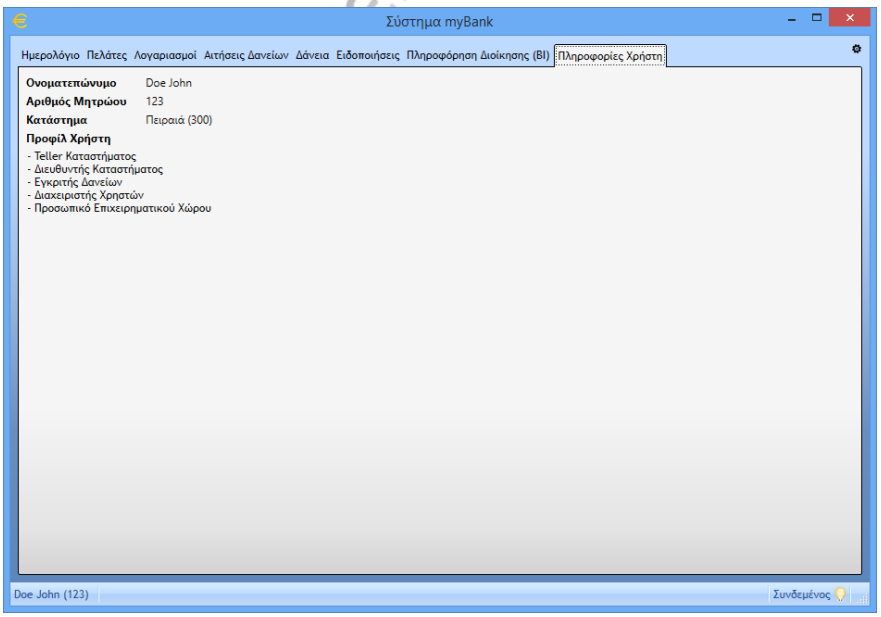
Γενικά Σύνολα

Περιγραφή	Ποσό
Συνολικό Κεφάλαιο Καταθέσεων	2.446.044,24
Κεφάλαιο Καταθέσεων, Τρεχούμενοι	2.277.626,91
Κεφάλαιο Καταθέσεων, Όψεως	146.760,33
Κεφάλαιο Καταθέσεων, Ταμιευτηρίου	21.657,00
Συνολικό Ποσό Εκταμιευμένων Δανείων	205.000,00
Συνολικό Ανεξόφλητο Ποσό Δανείων	240.535,25

Εικόνα 36: Πληροφόρηση Διοίκησης – Συγκεντρωτική Αναφορά

3.9 Πληροφορίες Χρήστη

Η τελευταία καρτέλα είναι η «Πληροφορίες Χρήστη». Αυτή η καρτέλα είναι διαθέσιμη σε όλους τους χρήστες και εμφανίζει τα βασικά στοιχεία του τρέχον χρήστη, όπως το ονοματεπώνυμο του, τον αριθμό μητρώου του, το κατάστημα που ανήκει και τις προσβάσεις που έχει. Παρακάτω φαίνεται η οθόνη πληροφοριών χρήστη.



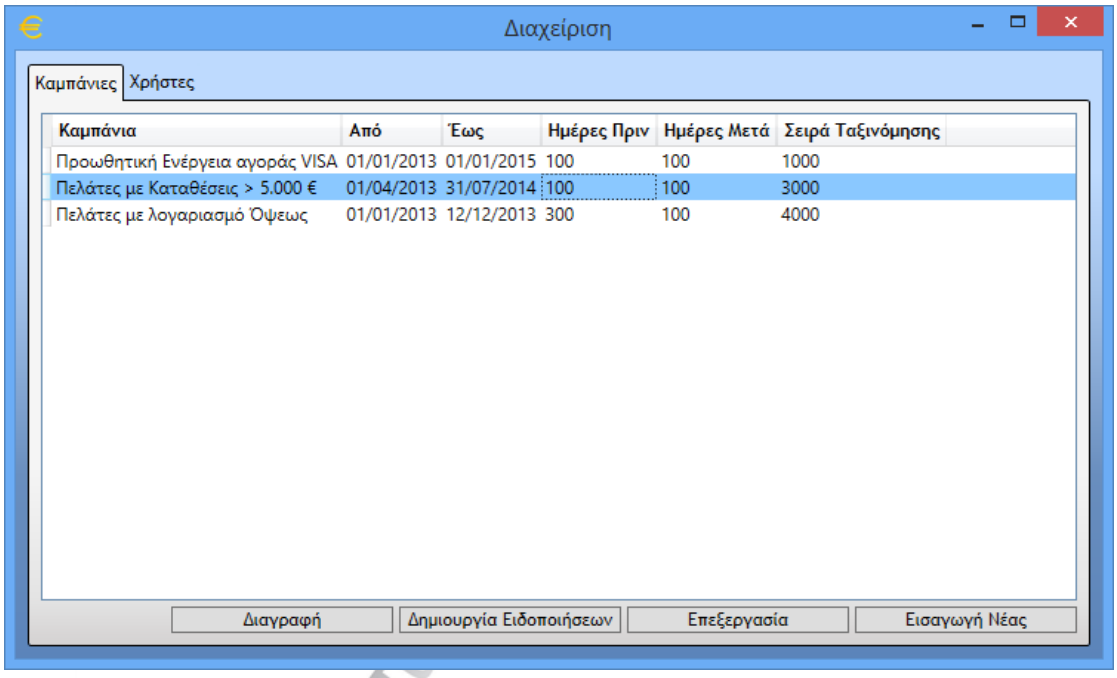
Εικόνα 37: Πληροφορίες Χρήστη

3.10 Διαχείριση

Τέλος, θα αναφερθούμε στο τελευταίο κομμάτι του συστήματος, που είναι η διαχείριση. Η διαχείριση έχει δύο κομμάτια.

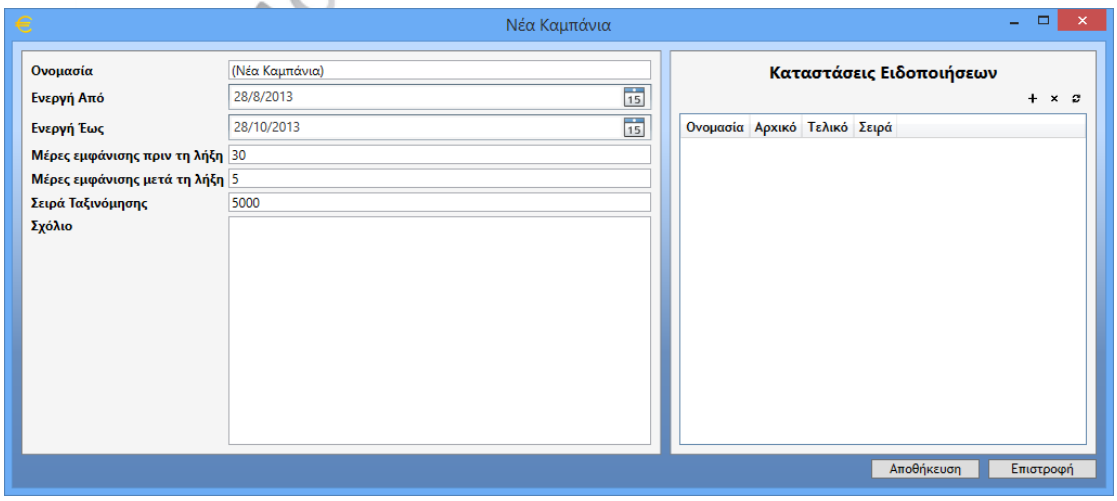
3.10.1 Καμπάνιες

Το πρώτο κομμάτι είναι η καρτέλα «Καμπάνιες», όπου γίνεται η διαχείριση των καμπανιών του συστήματος και αυτό είναι διαθέσιμο μόνο στους χρήστες με πρόσβαση «Προσωπικό Επιχειρηματικού Χώρου». Ο χρήστης μπορεί να εισάγει νέα καμπάνια, να επεξεργαστεί υπάρχουσα και να παράγει ειδοποιήσεις για συγκεκριμένη καμπάνια, που θα αφορούν συγκεκριμένες ομάδες πελατών, με διάφορα κριτήρια που θα δούμε στη συνέχεια. Παρακάτω φαίνεται η καρτέλα «Καμπάνιες».



Εικόνα 38: Διαχείριση Καμπανιών - Αναζήτηση

Παρακάτω φαίνεται η οθόνη εισαγωγή νέας καμπάνιας, η οποία ζητά όλα τα απαραίτητα πεδία που χρειάζεται να οριστούν σε μια προωθητική καμπάνια.

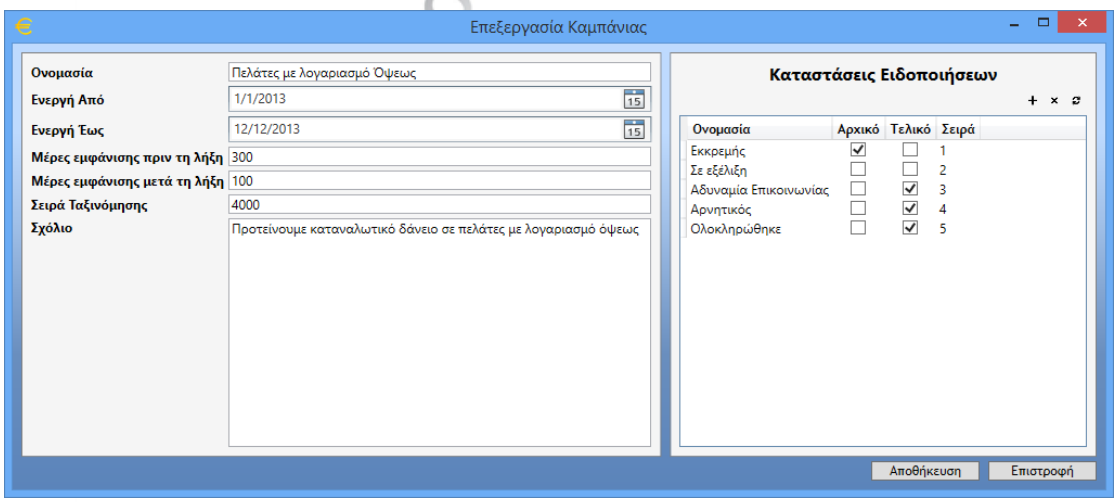


Εικόνα 39: Εισαγωγή Νέας Καμπάνιας

Η καμπάνια χρειάζεται διάφορα πεδία ώστε να οριστεί, τα οποία επεξηγούνται παρακάτω:

- Ονομασία
- Ενεργή Από, που είναι η ημερομηνία από την οποία θα είναι ενεργή η καμπάνια.
- Ενεργή Έως, που είναι η ημερομηνία μέχρι την οποία θα είναι ενεργή η καμπάνια. Αν υπάρχουν ειδοποιήσεις μιας μη ενεργής καμπάνιας στο σύστημα, τότε αυτές δεν θα εμφανίζονται στους χρήστες.
- Μέρες εμφάνισης πριν τη λήξη, που είναι ο αριθμός των ημερών πριν τη λήξη μιας ειδοποίησης, που η ειδοποίηση θα είναι ορατή στους χρήστες. Όταν δημιουργούνται ειδοποιήσεις για μία καμπάνια, τότε αυτές έχουν μια ημερομηνία λήξης (πέρα από τις ημερομηνίες από και έως της καμπάνιας).
- Μέρες εμφάνισης μετά τη λήξη, που είναι ο αριθμός των ημερών μετά τη λήξη μιας ειδοποίησης, που η ειδοποίηση θα είναι ορατή στους χρήστες.
- Σειρά Ταξινόμησης, που είναι ένας αριθμός που καθορίζει την σειρά που θα εμφανίζεται στους χρήστες η καμπάνια, σε σχέση με τις υπόλοιπες.
- Σχόλιο, που είναι μια σύντομη περιγραφή της καμπάνιας – οδηγία, ώστε οι χρήστες να καταλαβαίνουν τη σκοπιμότητα και τι θα πρέπει να κάνουν.
- Καταστάσεις Ειδοποιήσεων. Εδώ ορίζουμε τις πιθανές καταστάσεις που μπορούν να έχουν οι ειδοποιήσεις, οι οποίες θα είναι διαθέσιμες στο μενού καταστάσεις του χρήστη κατά τη διαχείριση μιας ειδοποίησης. Οι ειδοποιήσεις έχουν:
    - Ονομασία
    - Ένδειξη Αρχικού, αυτή την ένδειξη πρέπει να την πάρει μία και μόνο κατάσταση, ώστε όταν δημιουργούνται νέες ειδοποιήσεις, να παίρνουν αυτή την κατάσταση
    - Ένδειξη Τελικού, αυτή την ένδειξη μπορούν να την έχουν παραπάνω από μία καταστάσεις και υποδηλώνει ότι μια ειδοποίηση έχει φτάσει σε κατάσταση που έχει ολοκληρωθεί η διαχείρισή της
    - Σειρά, που είναι αριθμός που ταξινομεί τις καταστάσεις σχετικά με τη σειρά που θα εμφανίζονται στο μενού του χρήστη που τις διαχειρίζεται.

Παρακάτω φαίνεται συμπληρωμένη η οθόνη επεξεργασίας μιας καμπάνιας.



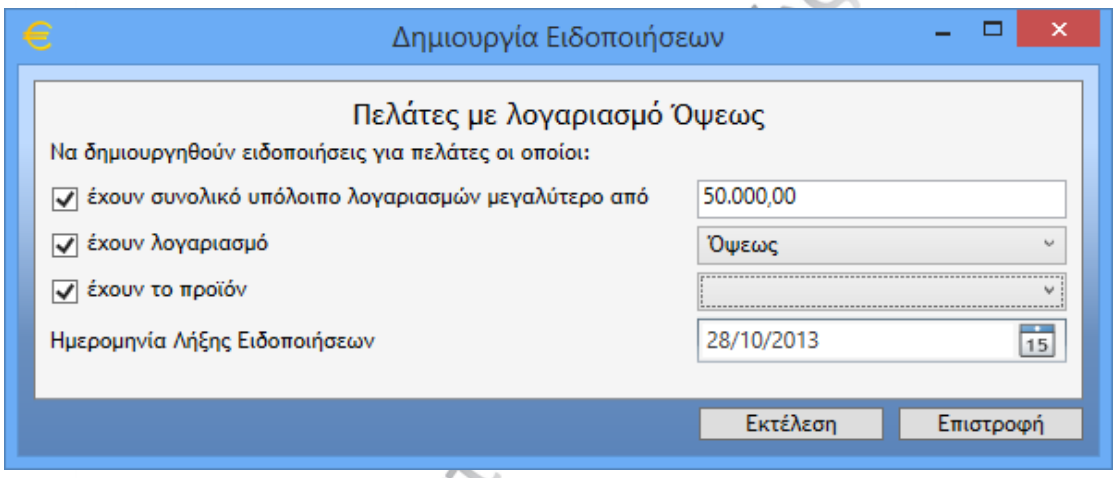
Εικόνα 40: Επεξεργασία Καμπάνιας

Ο χρήστης λοιπόν, έχει τη δυνατότητα να επιλέξει μια καμπάνια και να δημιουργήσει ειδοποιήσεις για ομάδες πελατών που πληρούν κάποια κριτήρια. Τα διαθέσιμα κριτήρια είναι:

- Αν ο πελάτης έχει συνολικό υπόλοιπο λογαριασμών μεγαλύτερο από ποσό που συμπληρώνει ο χρήστης
- Αν οι πελάτες έχουν συγκεκριμένο τύπο λογαριασμού που επιλέγει ο χρήστης
- Αν οι πελάτες έχουν συγκεκριμένο δανειακό προϊόν που επιλέγει ο χρήστης

Τέλος, ο χρήστης πρέπει να επιλέξει μια ημερομηνία λήξης για τις ειδοποιήσεις που θα δημιουργηθούν. Αφού λοιπόν πατήσει το κουμπί «Εκτέλεση», εκείνη τη στιγμή δημιουργούνται στο σύστημα ειδοποιήσεις σύμφωνα με τα κριτήρια που επέλεξε, για την συγκεκριμένη καμπάνια.

Οι ειδοποιήσεις που δημιουργούνται παίρνουν ημερομηνία λήξης, την ημερομηνία που συμπληρώθηκε, ως κατάστημα παίρνουν το κατάστημα πελατείας του κάθε πελάτη και αν αφορούν λογαριασμό, τότε συμπληρώνεται και ο λογαριασμός του πελάτη. Όλα αυτά τα πεδία είναι διαθέσιμα στην καρτέλα «Ειδοποιήσεις» των χρηστών καταστήματος. Παρακάτω φαίνεται η οθόνη δημιουργίας ειδοποιήσεων.



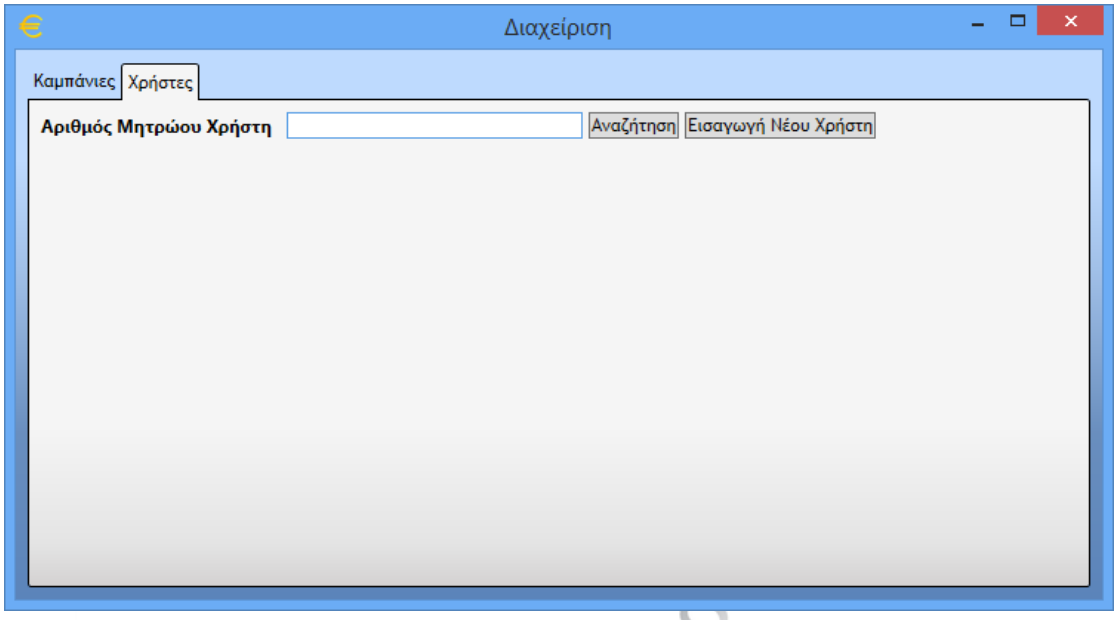
Εικόνα 41: Δημιουργία Ειδοποιήσεων Καμπάνιας με κριτήρια

Αν η διοίκηση του οργανισμού επιθυμεί να δημιουργήσει ειδοποιήσεις για μία καμπάνια που δεν μπορούν να εξυπηρετηθούν από αυτή την αυτοματοποιημένη οθόνη (τα συγκεκριμένα κριτήρια), τότε θα πρέπει να γίνει μηχανογραφική παρέμβαση για το φόρτωμα των νέων ειδοποιήσεων στο σύστημα, σύμφωνα με τις ανάγκες του οργανισμού.

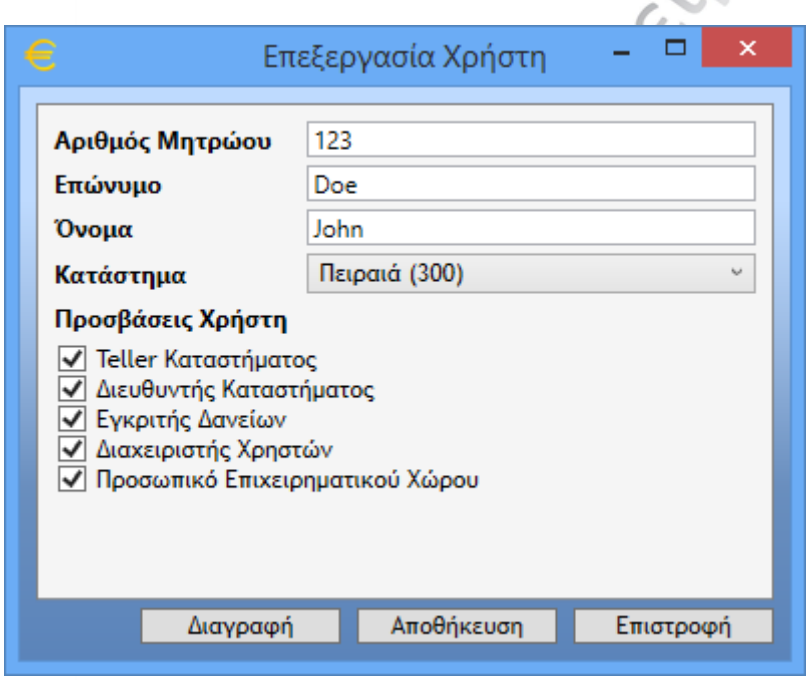
3.10.2 Χρήστες

Το δεύτερο κομμάτι της διαχείρισης είναι η καρτέλα «Χρήστες». Αυτή η καρτέλα είναι διαθέσιμη μόνο στους χρήστες που έχουν πρόσβαση «Διευθυντής Καταστήματος» και «Διαχειριστής Χρηστών». Ο διευθυντής καταστήματος μπορεί να δημιουργεί και να επεξεργάζεται τα στοιχεία και τις προσβάσεις μόνο των χρηστών του καταστήματός του, ενώ ο διαχειριστής χρηστών μπορεί να διαχειρίζεται οποιονδήποτε χρήστη. Αυτός ο ρόλος υπάρχει ώστε να υπάρχει κάποιος κεντρικός χρήστης που θα δημιουργεί χρήστες σε καταστήματα που δεν έχουν κανέναν χρήστη, ή να μεταφέρει χρήστες σε άλλα καταστήματα.

Παρακάτω μπορούμε να δούμε την οθόνη αναζήτησης χρήστη σύμφωνα με τον αριθμό μητρώου και την οθόνη επεξεργασίας του χρήστη.



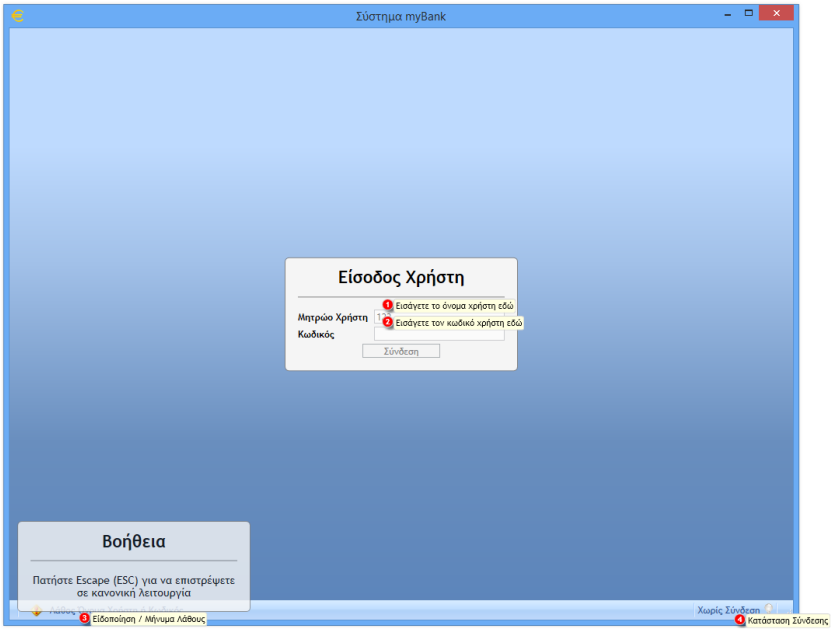
Εικόνα 42: Αναζήτηση Χρήστη – Εισαγωγή Νέου



Εικόνα 43: Επεξεργασία Χρήστη και προσβάσεων

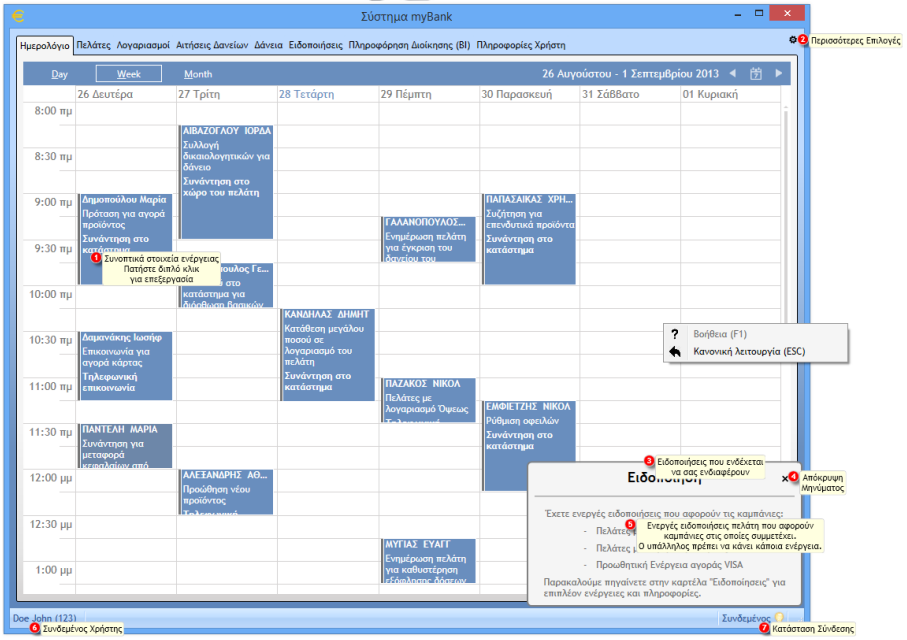
3.11 Λειτουργία Βοήθειας Χρήστη

Αυτές είναι όλες οι λειτουργίες του συστήματος. Τέλος, θα πρέπει να σημειώσουμε πως οποιοσδήποτε χρήστης χρειαστεί βοήθεια σχετικά με το πώς λειτουργεί κάποια οθόνη του συστήματος, μπορεί να πατήσει το κουμπί F1 (Βοήθεια), είτε να κάνει δεξί κλικ σε κάποιο σημείο της οθόνης και να επιλέξει F1 (Βοήθεια). Η εφαρμογή μπαίνει σε λειτουργία βοήθειας και εμφανίζονται κάποια βοηθητικά μηνύματα, που του εξηγούν τι πρέπει να κάνει στην συγκεκριμένη οθόνη που βρίσκεται. Για να επιστρέψει στην κανονική λειτουργία της εφαρμογής, θα πρέπει να πατήσει το κουμπί ESC (Escape), είτε με δεξί κλικ να επιλέξει «Κανονική Λειτουργία». Παρακάτω φαίνεται η αρχική οθόνη της εφαρμογής σε λειτουργία βοήθειας.



Εικόνα 44: Λειτουργία Βοήθειας Χρήστη (1)

Παρακάτω φαίνεται το μενού που εμφανίζεται, όταν ο χρήστης πατάει δεξί κλικ σε κάποιο σημείο της οθόνης.



Εικόνα 45: Λειτουργία Βοήθειας Χρήστη (2)

Στα παραρτήματα μπορείτε να βρείτε όλες τις υπόλοιπες οθόνες της εφαρμογής, που υποστηρίζουν λειτουργία βοήθειας.

4. Αρχιτεκτονική Συστήματος

Σε αυτή την ενότητα θα γίνει η ανάλυση της αρχιτεκτονικής του συστήματος. Θα παρουσιαστεί ο τρόπος με τον οποίο υλοποιήθηκε η εφαρμογή, ποια εργαλεία χρησιμοποιήθηκαν, ποια γλώσσα προγραμματισμού, ποια βάση δεδομένων καθώς και μεθοδολογίες ανάπτυξης κώδικα.

Η εφαρμογή μας είναι μια client εφαρμογή (fat client) που επικοινωνεί με έναν κεντρικό server, ο οποίος έχει πάνω του μια βάση δεδομένων Sql. Φτιάχτηκε με το Microsoft Visual Studio 2012 στη γλώσσα προγραμματισμού C#, Microsoft .NET Framework 4 και επικοινωνεί με βάση δεδομένων Microsoft Sql Server 2012, η οποία σχεδιάστηκε με το Microsoft Sql Server 2012 Management Studio.

Όλες οι τεχνολογίες, βιβλιοθήκες και μεθοδολογίες που χρησιμοποιήθηκαν είναι οι παρακάτω:

- Γλώσσα προγραμματισμού C#, Microsoft .NET Framework 4, που είναι μια γλώσσα προγραμματισμού της Microsoft με πάρα πολλές δυνατότητες και ευκολίες προς τους προγραμματιστές.
- WPF (Windows Presentation Foundation), που είναι η τεχνολογία της Microsoft για ανάπτυξη παραθυρικών εφαρμογών. Είναι η νεότερη τεχνολογία της Microsoft για τέτοιες εφαρμογές και ουσιαστικά αντικαθιστά τα Windows Forms Applications.
- Microsoft Sql Server 2012, που είναι server για σχεσιακές βάσεις δεδομένων, όπου αποθηκεύονται και ανακτόνται όλα τα δεδομένα του συστήματος.
- PersistentLib2009, που είναι μια βιβλιοθήκη για ORM (Object Relational Mapping) και επικοινωνία των εφαρμογών με σχεσιακές βάσεις δεδομένων οποιασδήποτε εταιρίας (Sql Server, MySql, Oracle Sql, IBM DB2 κα.).
- Model – View – ViewModel (MVVM), που είναι ένα design pattern (μεθοδολογία ανάπτυξης κώδικα) το οποίο προάγεται και ουσιαστικά είναι ενσωματωμένο στην τεχνολογία WPF.
- Inversion of Control (IoC), που είναι ένα design pattern για ανάπτυξη επεκτάσιμων και συντηρήσιμων εφαρμογών, το οποίο εδώ και κάποια χρόνια είναι πολύ δημοφιλές τις προγραμματιστικές κοινότητες και χρησιμοποιείται από πάρα πολλούς οργανισμούς ανάπτυξης λογισμικού.

Παρακάτω θα αναλύσουμε με περισσότερη λεπτομέρεια μερικά από τα παραπάνω, σχετικά με τις βασικές τους αρχές και την υλοποίηση τους στην εφαρμογή μας.

4.1 C#

Η γλώσσα στην οποία αναπτύχθηκε η εφαρμογή είναι η C# (C Sharp), η οποία υποστηρίζεται από την τεχνολογία παραθυρικών εφαρμογών της Microsoft, το WPF. Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού μέσα στην πλατφόρμα του Microsoft .NET. Ο στόχος της είναι να είναι μια απλή, μοντέρνα, γενικού σκοπού αντικειμενοστραφής γλώσσα, η οποία έχει σχεδιαστεί για το Microsoft CLI (Common Language Infrastructure).

Είναι μια γλώσσα με πάρα πολλές δυνατότητες και οι προγραμματιστές που τη χρησιμοποιούν μπορούν με ευκολία να υλοποιήσουν λειτουργίες που σε άλλες γλώσσες θα ήταν επίπονες, μέσω των πολλών βιβλιοθηκών που προσφέρονται.

Παρακάτω θα αναφέρουμε τρία χαρακτηριστικά που προσφέρει η C#, .NET Framework 4 τα οποία χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής και είναι πολύ χρήσιμα στον προγραμματισμό συντηρήσιμων εφαρμογών καθώς και απλά παραδείγματα κώδικα της εφαρμογής. Αυτά είναι τα παρακάτω:

- Extension Methdos
- Async feature
- Linq

4.1.1 Extension Methods

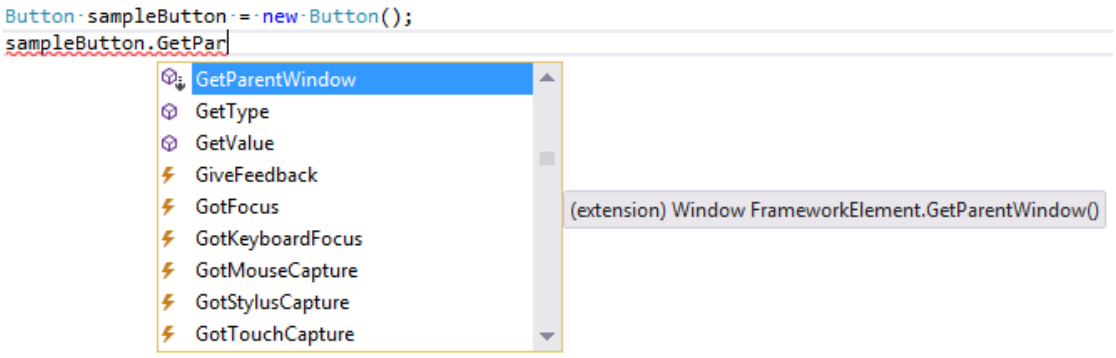
Όταν οριστεί ένας τύπος (πχ κλάση) σε κάποια βιβλιοθήκη .NET, τότε ο ορισμός αυτού του τύπου είναι τελικός. Ο μόνος τρόπος να προσθέσουμε νέα πράγματα είναι να να ξαναγράψουμε τον κώδικα και να τον κάνουμε πάλι compile σε μια νέα πλέον βιβλιοθήκη.

Από την έκδοση 3.5 του .NET Framework και μετά, μπορούμε να ορίσουμε extension methods (μεθόδους επέκτασης). Οι extension methods ουσιαστικά επιτρέπουν σε τύπους που έχουν ήδη γίνει compile σε κάποια βιβλιοθήκη, να αποκτήσουν νέες λειτουργικότητες χωρίς να πειραχτεί άμεσα ο τύπος που επεκτείνεται. Αυτή η τεχνική είναι πολύ χρήσιμη για τύπους που δεν έχουμε τον πηγαίο κώδικα και δεν μπορούμε να τους μεταβάλουμε πρωτογενώς. Αυτό γίνεται με έναν πολύ απλό τρόπο και δίνει την αίσθηση ότι αυτή η λειτουργικότητα υπήρχε στον τύπο από την αρχή. Παρακάτω παραθέτουμε ένα απλό παράδειγμα που χρησιμοποιήθηκε στην εφαρμογή μας:

```
public static class Helper
{
     public static Window GetParentWindow(this FrameworkElement element)
     {
         while (true)
         {
             if (element.Parent == null) { return null; }
             element = element.Parent as FrameworkElement;
             if (element is Window)
             {
                 return element as Window;
             }
         }
     }
     ……..
}
```

Μια extension method είναι μια στατική (static) μέθοδος η οποία πρέπει να οριστεί μέσα σε μία στατική κλάση. Ο τύπος που θέλουμε να προσθέσουμε την επιπλέον λειτουργικότητα θα πρέπει να περάσει μέσα στις παραμέτρους της μεθόδου και μπροστά να έχει το keyword «this».

Συγκεκριμένα στον κώδικά μας, η στατική κλάση λέγεται Helper, η extension method λέγεται GetParentWindow και ο τύπος στον οποίο προσθέτουμε την επιπλέον λειτουργικότητα είναι το FrameworkElement που ανήκει στη βιβλιοθήκη PresentationFramework.dll της Microsoft. Το FrameworkElement είναι ουσιαστικά ένα control στο user interface της εφαρμογής, δηλαδή όλα τα controls πάνω σε μια οθόνη είναι FrameworkElements. Ο κώδικας της μεθόδου ψάχνει επαναληπτικά τον πατέρα του FrameworkElement μέχρι να βρει πατέρα που είναι Window. Δηλαδή αυτό που κάνει ο κώδικας είναι ότι βρίσκει το παράθυρο (Window) στο οποίο ανήκει ένα control. Αφού οριστεί αυτή η extension method, τότε μπορεί να χρησιμοποιηθεί σε οποιοδήποτε σημείο του project μας, το οποίο έχει αναφορά στο namespace που έχει οριστεί η κλάση Helper, όπως φαίνεται παρακάτω.



Εικόνα 46: Extension Method στο Microsoft Visual Studio

4.1.2 Async

Το async είναι ένα χαρακτηριστικό του compiler της γλώσσας C# που προστέθηκε στην έκδοση 5.0. Αυτό το χαρακτηριστικό κάνει τον ασύγχρονο προγραμματισμό πολύ ευκολότερο, αφαιρώντας την ανάγκη για περίπλοκο κώδικα που χρειαζόταν παλιότερα. Ο ασύγχρονος προγραμματισμός ήταν πάντα δυνατόν να γίνει στη C#, αλλά χρειαζόταν πάρα πολύς κώδικας από τους προγραμματιστές.

Ο λόγος που χρησιμοποιούνται ασύγχρονες λειτουργίες στην εφαρμογή, είναι για να μην κολλάει το user interface όταν το πρόγραμμα κάνει εργασίες που δεν φαίνονται στο χρήστη. Με το async είναι πλέον πολύ εύκολο να υλοποιηθούν ασύγχρονες λειτουργίες με ασφάλεια, χωρίς να χρειάζεται μεγάλη προσπάθεια. Παρακάτω παραθέτεται παράδειγμα που χρησιμοποιούμε το async στην εφαρμογή.

```
void ExtendedWindow_LocationChanged(object sender, EventArgs e)
{
    if(IsHelpMode)
    {
        AsyncRelocatePopups();
    }
}

private async void AsyncRelocatePopups()
{
    Task t = Task.Factory.StartNew(() =>
    {
        Thread.Sleep(800);
    });

    await t;

    if ((_lastTop != Top || _lastLeft != Left ||
        _lastWidth != ActualWidth || _lastHeight != ActualHeight ||
        _lastWindowState != WindowState) && IsHelpMode)
    {
        PopulatePopups();
    }
}
```

Για να τρέξει μια μέθοδος ασύγχρονα πρέπει να έχει μπροστά το keyword async και κάπου μέσα στο σώμα της να έχει το keyword await. Αυτό που κάνει ο compiler είναι ότι ανοίγει αυτή την μέθοδο άλλο thread από το βασικό thread της εφαρμογής και όταν τελειώσει, εκτελεί τον κώδικα που βρίσκεται μετά το await στο βασικό thread.

Το συγκεκριμένο παράδειγμα είναι στον κώδικα του παραθύρου της εφαρμογής και αναλαμβάνει να μετακινήσει τα popup controls (αναδυόμενα στοιχεία) του παραθύρου, όταν το παράθυρο κινείται με το ποντίκι από το χρήστη. Τα popup controls είναι οι οδηγίες προς τον χρήστη όταν είναι σε λειτουργία βοήθειας. Η ασύγχρονη μέθοδος, περιμένει 800 χιλιοστά του δευτερολέπτου για να μετακινήσει τα popups και ελέγχει αν το παράθυρο έχει αλλάξει πάλι θέση μέσα σε αυτό το χρόνο. Αυτό συμβαίνει διότι ο χρήστης μπορεί να έχει ξαναλλάξει τη θέση του παραθύρου σε αυτό το χρόνο με αποτέλεσμα, όταν ο χρήστης σύρει με το ποντίκι το παράθυρο, να κολλάει.

4.1.3 Linq

Το Linq (Language-Integrated Query) είναι μια τεχνολογία της Microsoft για την απλοποίηση και ενοποίηση της πρόσβασης σε δεδομένα (data access) από οποιαδήποτε πηγή δεδομένων. Με το Linq μπορούμε να γράψουμε πιο κομψό και ευέλικτο κώδικα, όχι μόνο για να επικοινωνήσουμε με βάσεις και αρχεία, αλλά και για να χειριστούμε δομές δεδομένων.

Το Linq ενοποιεί τον τρόπο με τον οποίο τα δεδομένα μπορούν να αντληθούν από οποιοδήποτε αντικείμενο που υλοποιεί το interface IEnumerable<T> (δηλαδή οποιουδήποτε τύπου λίστα). Πίνακες, συλλογές, σχεσιακά δεδομένα, XML και Json μπορεί να είναι πιθανές πηγές δεδομένων. Τα κύρια τρία κομμάτια του Linq είναι τα παρακάτω:

- Linq σε αντικείμενα, που είναι ένα API που προσφέρει μεθόδους που ανακτούν δεδομένα από οποιοδήποτε αντικείμενο IEnumerable<T>. Αυτά τα «queries» γίνονται πάνω σε δεδομένα της μνήμης του προγράμματος.
- Linq στο ADO.NET, που προσφέρει κάποιες βασικές λειτουργίες από queries (Standard Query Operations) για σχεσιακά δεδομένα.
- Linq σε SQL, που χρησιμοποιείται για ερωτήματα σε σχεσιακές βάσεις δεδομένων όπως ο Microsoft Sql Server.
- Linq σε XML που υποστηρίζει βασικές λειτουργίες ερωτημάτων πάνω σε δεδομένα που είναι σε μορφή XML.

Στην εφαρμογή μας χρησιμοποιήσαμε μόνο το Linq σε αντικείμενα, για ερωτήματα πάνω σε λίστες που είχαμε στο μοντέλο δεδομένων μας. Παρακάτω φαίνεται ένα πολύ απλό ερώτημα σε Linq που χρησιμοποιήθηκε στην εφαρμογή.

```
public List<IEntityType_Ex> FindMany(string name)
{
    var res = from et in _EntityTypes
              where et._Name.ToLower().Equals(name.ToLower())
              select et;
    return res.ToList();
}
```

Εδώ ορίζεται μια μέθοδος που παίρνει σαν όρισμα ένα string και κάνει ερώτημα σε μια λίστα από αντικείμενα IEntityType_Ex για να βρει τα αντικείμενα των οποίων το Name είναι αυτό που έχει περαστεί στο όρισμα name της μεθόδου, χωρίς να ελέγχει για ταυτοποίηση μικρών και κεφαλαίων χαρακτήρων.

4.2 WPF

Το WPF (Windows Presentation Foundation) είναι ένα νέο σύστημα γραφικής αναπαράστασης για τα Windows. Είναι φτιαγμένο για το Microsoft .NET και είναι επηρεασμένο από τις μοντέρνες τεχνικές αναπαράστασης όπως το HTML και Flash και χρησιμοποιεί hardware acceleration. Είναι η πιο ριζική αλλαγή στο user interface των Windows από την εποχή των Windows 95.

Πριν το WPF, ένας προγραμματιστής μπορούσε να φτιάξει παραθυρικές εφαρμογές Windows χρησιμοποιώντας τα Windows Forms Applications. Το WPF είναι ουσιαστικά ο αντικαταστάτης των Windows Forms Applications. Μερικά σημαντικά πλεονεκτήματα του WPF σε σχέση με τα Window Forms φαίνονται παρακάτω:

- Μειώνεται κατά πολύ ο κώδικας που πρέπει να γραφτεί. Με το WPF Data Binding και το XAML που θα δούμε παρακάτω, ο κώδικας μειώνεται πάρα πολύ, καθώς ο προγραμματιστής δεν θα πρέπει να γράφει σε κάθε οθόνη το mapping των τιμών των μεταβλητών πάνω σε controls του user interface.
- Το user interface και ο κώδικας μπορούν να διαχωριστούν. Έτσι μπορούν οι προγραμματιστές και οι σχεδιαστές του user interface μπορούν να δουλεύουν παράλληλα και ανεξάρτητα.
- Τα Routed Commands που προσφέρει το WPF βοηθάνε την εύκολη διαχείριση των λειτουργιών του user interface.
- Το XAML, το styling των controls και τα control templates επιτρέπουν την εύκολη παρουσίαση περίπλοκων δομών δεδομένων και δημιουργία προσαρμοσμένων control, όπως επιθυμεί ο προγραμματιστής.

Στη συνέχεια θα πούμε μερικά λόγια για το XAML και το Data Binding που είναι πολύ σημαντικά στοιχεία του WPF καθώς και απλά παραδείγματα που χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής.

4.2.1 XAML

Το XAML (Extensible Application Markup Language) είναι μια markup γλώσσα για τη δημιουργία user interface (διεπαφή χρήστη). Με άλλα λόγια, το XAML περιγράφει την τοποθέτηση των panel, των πινάκων, των κουμπιών και γενικότερα όλων των control που είναι τα συστατικά στοιχεία ενός παραθύρου. Με το XAML μπορεί να γίνει εύκολα ο διαχωρισμός του user interface με την προγραμματιστική λογική που υπάρχει στο παρασκήνιο, πράγμα πολύ σημαντικό για μεγάλες εφαρμογές με πλούσια user interfaces.

4.2.2 Data Binding

Το Data Binding είναι μια παραδοσιακή τεχνική όπου αντλούμε δεδομένα από κάποιο αντικείμενο και τα δείχνουμε στο user interface μιας εφαρμογής, χωρίς να χρειάζεται να γράψουμε μεγάλα κομμάτια κώδικα που κάνουν το mapping μεταξύ των πεδίων του αντικειμένου και των control του user interface.

Στο WPF μπορεί λοιπόν να γίνει το mapping των τιμών των πεδίων των αντικειμένων με το user interface. Οι τιμές των πεδίων που παρουσιάζονται μπορούν να ενημερώνονται αυτόματα με πιθανές αλλαγές, αμφίδρομα. Το Data Binding στο WPF έχει πάρα πολλές δυνατότητες και μπορεί να δώσει πολύ αποτελεσματικές λύσεις στην αναπαράσταση δεδομένων. Το WPF περιέχει επίσης ένα σύνολο από controls για λίστες, που μπορεί να χειριστεί συλλογές δεδομένων και να μες επιτρέψει να περιηγηθούμε σε αυτές με μεγάλη ευκολία.

Παρακάτω φαίνεται ένα απλό παράδειγμα XAML κώδικά που περιγράφει το user interface για την αναπαράσταση και διαχείριση ενός λογαριασμού.

```
<UserControl x:Class="myBank.EditAccountControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:local="clr-namespace:myBank"
>
    <Grid>
        <Grid.Resources>
            <Style TargetType="TextBlock">
                <Setter Property="FontWeight" Value="Bold"/>
                <Setter Property="VerticalAlignment" Value="Center"/>
            </Style>
        </Grid.Resources>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="4"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="4"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="4"/>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <TextBlock Grid.Row="0" Grid.Column="0" Text="Πελάτης"/>
        <TextBlock Grid.Row="2" Grid.Column="0" Text="Τύπος Λογαριασμού"/>
        <TextBlock Grid.Row="4" Grid.Column="0" Text="Κατάστημα Λογαριασμού"/>

        <local:CustomerSearch Grid.Row="0" Grid.Column="2" Owner="{Binding
Path=.}" IsEnabled="{Binding Path=CanEditCustomer, RelativeSource={RelativeSource
AncestorType=local:EditAccountWindow}}"/>
        <ComboBox Grid.Row="2" Grid.Column="2" ItemsSource="{Binding _Types}"
DisplayMemberPath="_Value"
        SelectedValue="{Binding _AccountType}" SelectedValuePath="_Code"/>
        <ComboBox Grid.Row="4" Grid.Column="2" ItemsSource="{Binding _Branches}"
DisplayMemberPath="_FullName"
        SelectedValue="{Binding _Branch}" SelectedValuePath="_BranchID"
IsEnabled="False"/>
    </Grid>
</UserControl>
```

Όπως φαίνεται στον παραπάνω κώδικα, ορίζουμε ένα νέο UserControl με όνομα EditAccountControl, που είναι ένα control ορισμένο από εμάς για την αναπαράσταση του λογαριασμού. Αυτό το control μπορεί να χρησιμοποιηθεί σε οποιοδήποτε παράθυρο που θέλουμε να δείξουμε το λογαριασμό. Αρχικά μέσα στο UserControl ορίζουμε ένα Grid, που είναι ένα control το οποίο μας επιτρέπει να στοιχίσουμε τα controls που θα χρησιμοποιήσουμε. Αυτό το Grid έχει 5 γραμμές, 3 εκ των οποίων είναι για τα controls και 2 για τα κενά μεταξύ τους. Το Grid έχει και 3 στήλες, η πρώτη για τις ετικέτες, η δεύτερη για τα κενά μεταξύ των control και η τρίτη για τα controls.

Στη συνέχεια ορίζουμε τις ετικέτες Πελάτης, Τύπος Λογαριασμού και Κατάστημα και τα βάζουμε στις γραμμές 0 , 2 και 4 αντίστοιχα (Grid.Row=”0”). Όλες οι ετικέτες πηγαίνουν στην στήλη 0 του Grid (Grid.Column=”0”). Αντίστοιχα στη στήλη 2, βάζουμε ένα user control «Customer Search» που είναι το control που διαχειρίζεται την επιλογή πελάτη και χρησιμοποιείται σε όλα τα σημεία της εφαρμογής που θέλουμε να επιλέξουμε πελάτες. Επίσης βάζουμε και δύο ComboBox, το ένα για την επιλογή τύπου λογαριασμού και το άλλο για το κατάστημα λογαριασμού (το κατάστημα λογαριασμού είναι read only).

Όπως βλέπουμε στο πρώτο ComboBox έχουμε ορίσει στο πεδίο ItemsSource ένα Binding (ItemsSource="{Binding _Types}"). Ουσιαστικά αυτό χρησιμοποιεί το Data Binding και αναφέρεται σε μία λίστα από τύπους λογαριασμού με όνομα Types. Αν κοιτάξουμε την κλάση του λογαριασμού Account_Ex, αυτή ορίζει μια λίστα από AccountTypes που είναι οι διαθέσιμοι τύποι λογαριασμού. Έτσι, γεμίζει αυτόματα το ComboBox με τις τιμές που διαβάζει από την λίστα αυτή. Επίσης, έχουμε ορίσει στο πεδίο SelectedValue ένα Binding (SelectedValue="{Binding _AccountType}"). Αυτό είναι ένα Data Binding που ορίζει πως η επιλεγμένη τιμή από το ComboBox θα αποθηκεύεται στο πεδίο _AccountType της κλάσης του λογαριασμού (Account_Ex).

Τέλος, να αναφέρουμε πως στην δήλωση του Grid έχουμε ορίσει Grid.Resources και μέσα σε αυτό έχουμε βάλει ένα Style για TextBlock (ετικέτες). Αυτό δημιουργεί ένα Style που εφαρμόζεται σε όλα τα TextBlock του Grid και ουσιαστικά ορίζουμε δύο πεδία του TextBlock, την κάθετη στοίχιση (VerticalAlignment) και το FontWeight. Με αυτό τον τρόπο μπορούμε να ορίζουμε styles για συγκεκριμένους τύπους control και να τα χρησιμοποιούμε για να διαμορφώνουμε όπως θέλουμε του «στυλ» αυτών των control σε κάποια κλίμακα, είτε σε όλη την εφαρμογή.

4.3 Βάση Δεδομένων

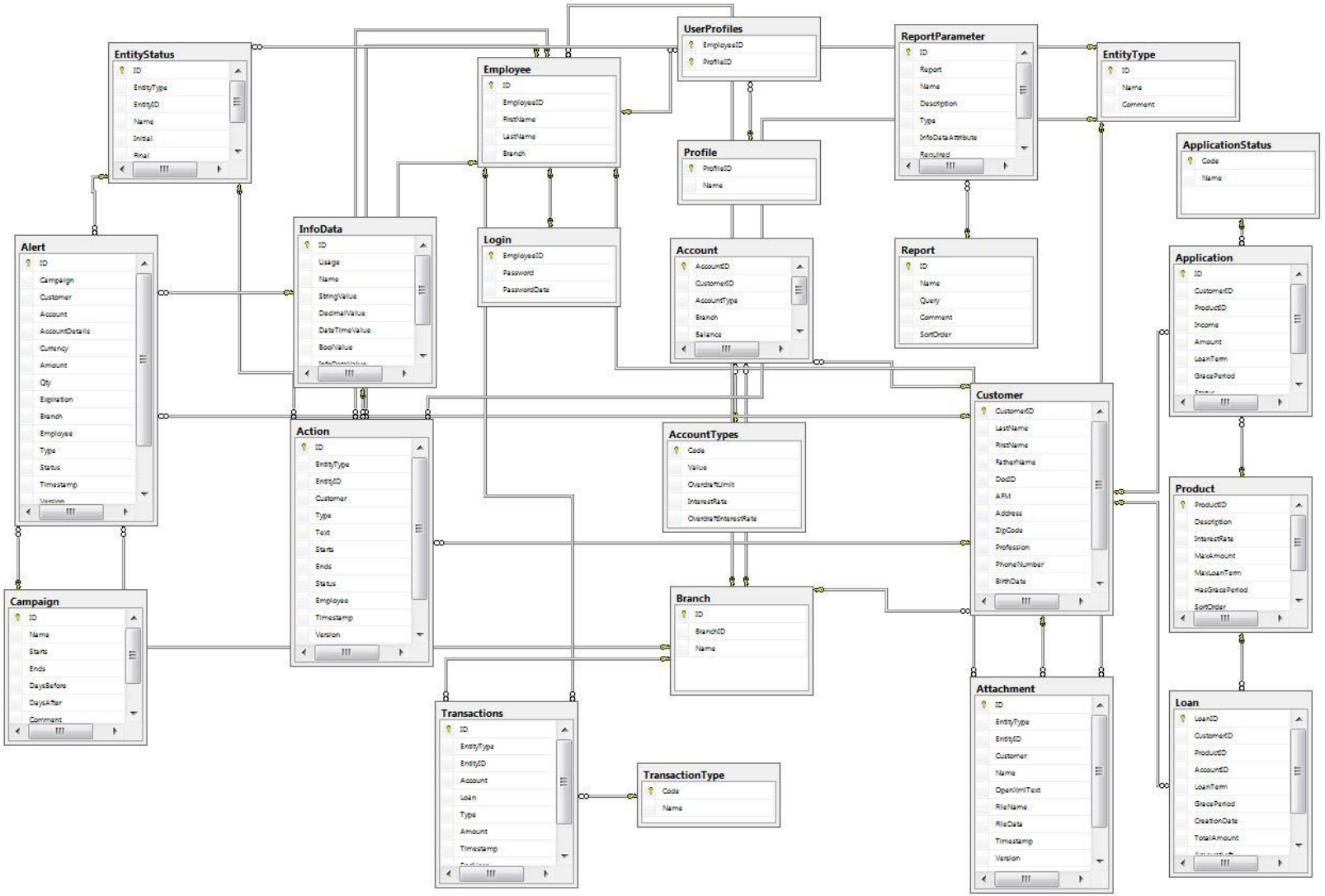
Για την υλοποίηση του συστήματος χρησιμοποιήσαμε τον Microsoft Sql Server 2012, ο οποίος είναι ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS). Ένα RBDMS δεν είναι απλά ένας χώρος αποθήκευσης των δεδομένων ενός συστήματος. Αν αυτή ήταν η μοναδική μας ανάγκη, τότε θα μπορούσαμε να χρησιμοποιήσουμε ένα απλό σύστημα αρχείων (file system).

Τα σύγχρονα RDBMS διαχειρίζονται τα δεδομένα ενός συστήματος, περιορίζουν τα δεδομένα που μπορεί κάποιος να διαβάσει και βοηθάει την άμεση πρόσβαση των δεδομένων. Επίσης ένα σύγχρονο RDBMS μας επιτρέπει να ορίσουμε το πώς θα πρέπει να είναι τα δεδομένα και τους πιθανούς επιχειρηματικούς κανόνες που μπορεί να υπάρχουν πάνω στα δεδομένα. Ένα τέτοιο σύστημα λοιπόν, είναι κατάλληλο για την υποστήριξη ενός συστήματος σαν το δικό μας.

Παρακάτω παραθέτουμε τα βασικά πράγματα που μας προσφέρει ο Microsof Sql Server:

- Πίνακες
- Indexes (ευρετήρια), για γρήγορη προσπέλαση δεδομένων σε μεγάλους όγκους πληροφοριών
- Διαγράμματα
- Views (Όψεις)
- Stored Procedures
- User-defined functions, συναρτήσεις ορισμένες από το χρήστη
- Sequences
- Χρήστες
- Roles
- Reports
- Full-text catalogs, για υποστήριξη αναζήτησης κειμένου μέσω full-text search
- User-defined data types, δομές δεδομένων ορισμένες από το χρήση
- Transaction log, log συναλλαγών

Στην εφαρμογή μας χρησιμοποιήσαμε πίνακες, indexes, διαγράμματα, stored procedures. Στη συνέχεια θα δούμε το Διάγραμμα Οντοτήτων – Συσχετίσεων του συστήματός μας καθώς και τη βασική δομή των σημαντικότερων πινάκων.



Εικόνα 47:Διάγραμμα Οντοτήτων - Συσχετίσεων myBank

Στο παραπάνω σχήμα φαίνεται το διάγραμμα οντοτήτων συσχετίσεων της βάσης δεδομένων του συστήματός μας. Σε αυτό το διάγραμμα αποτυπώνονται όλες οι οντότητες του συστήματος. Παρακάτω θα πούμε λίγα πράγματα για τις βασικές οντότητες (πίνακες) του μοντέλου δεδομένων του συστήματος. Αυτές είναι:

- Πελάτης (Customer)
- Λογαριασμός (Account)
- Αίτηση Δανείου (Application)
- Δάνειο (Loan)
- Κατάστημα (Branch)
- Χρήστης (Employee)
- Καμπάνια – Ειδοποίηση (Campaign – Alert)
- Ραντεβού – Ενέργεια (Action)
- Επισυναπτόμενο – Σχόλιο (Attachment)
- Αναφορά (Report)

4.3.1 Πελάτης

Ο πρώτος πίνακας που θα δούμε είναι ο πίνακας Customer, όπου αποθηκεύονται τα βασικά στοιχεία του πελάτη, μαζί με έναν κωδικό ο οποίος είναι η ταυτότητα της εγγραφής (Primary Key). Είναι ο βασικότερος πίνακας του συστήματος, αφού το σύστημά μας είναι πελατοκεντρικό και υπάρχουν πολλοί άλλοι πίνακες που έχουν foreign key πάνω στην ταυτότητα του πελάτη (CutomerID).

Κατά τη δημιουργία ενός πελάτη, κρατείται στην κολώνα CreationBranch ο κωδικός του καταστήματος στο οποίο δημιουργήθηκε και αυτό αποτελεί το κατάστημα πελατείας του πελάτη. Αυτό θα είναι και το κατάστημα που θα διαχειρίζεται τον χρήση (κατάστημα εξυπηρέτησης).

4.3.2 Λογαριαμός

Στον πίνακα Account κρατούνται όλοι οι λογαριασμοί καταθέσεων του συστήματος. Τα βασικά πεδία του πίνακα είναι ο πελάτης (CustomerID) που είναι foreing key στον πίνακα Customer, το AccountType που είναι foreign key στον πίνακα AccountTypes και δηλώνει τον τύπο λογαριασμού και το κατάστημα που είναι το κατάστημα που ανήκει ο λογαριασμός (foreign key στον πίνακα Branch). Επίσης υπάρχουν τα πεδία Balance που είναι το διαθέσιμο υπόλοιπο του λογαριασμού και CreationDate που είναι η ημερομηνία και ώρα δημιουργίας του λογαριασμού.

4.3.3 Αίτηση Δανείου

Ο πίνακας Application συλλέγει όλες τις πληροφορίες που έχουν να κάνουν με μια αίτηση ενός πελάτη για ένα νέο δάνειο. Τα βασικότερα από αυτά είναι τα παρακάτω.

- CustomerID, κωδικός πελάτη ο οποίος κάνει την αίτηση
- ProductID, κωδικός προϊόντος που θέλει να πάρει ο πελάτης
- Income, ετήσιο εισόδημα πελάτη
- Amount, αιτούμενο ποσό δανείου
- Status, κωδικός κατάστασης αίτησης, που μπορεί να είναι εκκρεμής, προς έγκριση, εγκεκριμένη, ολοκληρωμένη

Ανάλογα με την φάση έγκρισης που βρίσκεται η αίτηση του πελάτη, η κάθε εγγραφή στον πίνακα Application έχει και το αντίστοιχο Status, που είναι foreign key στον πίνακα ApplicationStatus που έχει τις περιγραφές των καταστάσεων αίτησης. Τέλος, το ProductID είναι foreign key στον πίνακα Product που έχει όλες τις παραμέτρους των διαθέσιμων δανειακών προϊόντων.

4.3.4 Δάνειο

Στον πίνακα Loan αποθηκεύονται οι πληροφορίες που αφορούν τα δάνεια. Οι βασικές κολώνες του πίνακα είναι οι παρακάτω:

- CustomerID, κωδικός πελάτη στον οποίο ανήκει το δάνειο, foreign key στον πίνακα Customer
- ProductID, κωδικός προϊόντος δανείου που προσδιορίζει ποιο είναι το δανειακό προϊόν, foreign key στον πίνακα Product
- AccountID, λογαριασμός εξυπηρέτησης που είναι ο λογαριασμός στον οποίο γίνεται η εκταμίευση του δανείου.
- LoanTerm, διάρκεια δανείου σε μήνες
- CreationDate, ημερομηνία ανοίγματος δανείου
- AmountLeft, υπόλοιπο ποσό κεφαλαίου που δεν έχει εξοφλήσει ο πελάτης
- Deposited, ένδειξη αν έχει γίνει εκταμίευση του δανείου στον λογαριασμό εξυπηρέτησης.

4.3.5 Κατάστημα

Στον πίνακα Branch υπάρχουν όλα τα καταστήματα της τράπεζας. Αυτός ο πίνακας έχει foreign key αναφορές από πολλούς άλλους πίνακες όπως είναι ο Employee, ο Customer, ο Account και άλλα.

4.3.6 Χρήστης

Στον πίνακα Employee είναι αποθηκευμένα τα βασικά στοιχεία του κάθε χρήστη, όπως το ονοματεπώνυμο και το κατάστημα στο οποίο ανήκει. Ο κωδικός πελάτη έχει πολλές αναφορές από άλλους πίνακες. Στον πίνακα Login αποθηκεύονται τα δεδομένα ταυτοποίησης του χρήστη (κωδικός). Στον πίνακα UserProfiles (σχέση 1 προς n) αποθηκεύονται όλες οι προσβάσεις που έχει ο χρήστης.

4.3.7 Καμπάνια – Ειδοποίηση

Στους πίνακες Campaign (καμπάνια) και Alert (ειδοποίηση) γίνεται η υλοποίηση του μοντέλου των προωθητικών καμπανιών του συστήματός μας.

Στον πίνακα Campaign αποθηκεύονται όλες οι παραμέτρους μιας καμπάνιας που έχουμε αναφέρει σε άλλες ενότητες. Τα βασικά πεδία του πίνακα Campaign είναι:

- Name, ονομασία καμπάνιας
- Starts, ημερομηνία έναρξης
- Ends, ημερομηνία λήξης
- DaysBefore, μέρες πριν την λήξη μιας ειδοποίησης αυτής της καμπάνιας που θα είναι ορατή στους χρήστες
- DaysAfter, μέρες μετά την λήξη μιας ειδοποίησης αυτής της καμπάνιας, που θα είναι ορατή στους χρήστες
- Comment, σχόλιο – οδηγία προς τους χρήστες για την καμπάνια
- Στον πίνακα Alert αποθηκεύονται οι ειδοποιήσεις όλων των καμπανιών με τις παραμέτρους τους. Παρακάτω είναι τα βασικά πεδία του πίνακα Alert:
- Campaign, κωδικός καμπάνιας που αφορά η ειδοποίηση
- Customer, κωδικός πελάτη που αφορά η ειδοποίηση
- Account, αριθμός λογαριασμού που μπορεί να αφορά η ειδοποίηση
- AccountDetails, επιπλέον πληροφορίες που μπορεί να αφορούν το λογαριασμό της ειδοποίησης

- Expiration, ημερομηνία λήξης ειδοποίησης. Αυτό καθορίζει το διάστημα που θα εμφανίζεται η ειδοποίηση στους χρήστες, ανάλογα με τα πεδία DaysBefore και DaysAfter της καμπάνιας της ειδοποίησης
- Status, κατάσταση ειδοποίησης. Είναι foreign key που κοιτά στον πίνακα EntityStatus. Ο EntityStatus είναι ένας πίνακας που έχει τις περιγραφές των καταστάσεων για διάφορες οντότητες (ειδοποιήσεις, ενέργειες - ραντεβού). Οι πιθανές καταστάσεις που μπορούν να πάρουν οι ειδοποιήσεις μιας συγκεκριμένης καμπάνιας, ορίζονται στην οθόνη επεξεργασίας καμπάνιας.
- Employee, είναι ο χρήστης που χειρίζεται την ειδοποίηση. Σε περίπτωση που κάποιος χρήστης επεξεργάζεται μια ειδοποίηση, τότε σε αυτό το πεδίο αποθηκεύεται ο κωδικός του ώστε να μην μπλέκουν οι διάφοροι χρήστες του καταστήματος τις ειδοποιήσεις

4.3.8 Ραντεβού – Ενέργεια

Στον πίνακα Action αποθηκεύονται τα στοιχεία όλων των ραντεβού – ενεργειών που έχουν αποθηκεύσει οι χρήστες, τα οποία εμφανίζονται και στον ημερολόγιό τους. Τα βασικά πεδία του πίνακα Action είναι τα παρακάτω:

- Customer, κωδικός πελάτη που αφορά η ενέργεια, foreign key στον πίνακα Customer
- Type, που είναι ο τύπος της ενέργειας (συνάντηση στο κατάστημα, τηλεφωνική επικοινωνία)
- Text, σχόλιο ενέργειας
- Starts, ημερομηνία και ώρα έναρξης
- Ends, ημερομηνία και ώρα λήξης
- Status, κατάσταση ενέργειας (εκκρεμής, αναβλήθηκε, ολοκληρώθηκε)
- Employee, κωδικός χρήστη που αφορά η ενέργεια
- VersionTime, ημερομηνία και ώρα τελευταίας επεξεργασίας
- EntityType, τύπος συσχετιζόμενης οντότητας. Εδώ αποθηκεύεται ο κωδικός του τύπου οντότητας και είναι foreign key στον πίνακα EntityType. Στον πίνακα EntityType υπάρχουν οι περιγραφές όλων των οντοτήτων της βάσης. Αυτό το πεδίο υπάρχει ώστε να μπορούμε να συσχετίσουμε μια ενέργεια με κάποια άλλη οντότητα, πέραν του Customer που ήδη υπάρχει κολώνα γι αυτό. Αυτό χρησιμοποιείται για να συσχετίσουμε την ενέργεια με κάποια ειδοποίηση (Alert). Σε μελλοντική επέκταση του συστήματος θα μπορούσαμε να συσχετίσουμε μια ενέργεια με κάποια άλλη οντότητα, έχοντας ήδη την απαραίτητη υποδομή.
- EntityID, κωδικός συσχετιζόμενης οντότητας. Είναι ο κωδικός της συσχετιζόμενης οντότητας με την ενέργεια, ο τύπος της οποίας φαίνεται από το πεδίο EntityType. Για τις ενέργειές μας, έχουμε τις ταυτότητες εγγραφής των ειδοποιήσεων που μπορεί να αφορά η ενέργεια. Αν δεν συσχετίζεται με κάποια οντότητα η ενέργεια, τότε τα πεδία EntityType και EntityID είναι NULL

Η τεχνική με το EntityType και EntityID για συσχετισμό μιας οντότητας με διαφόρων τύπων άλλες οντότητες είναι μια τεχνική που χρησιμοποιείται σε σύγχρονα μοντέλα σχεσιακών βάσεων δεδομένων όπου σκοπός είναι να εξυπηρετηθούν νέες απαιτήσεις σε ένα σύστημα για αποθήκευση δεδομένων, χωρίς να χρειάζεται αλλαγή του σχήματος της βάσης.

4.3.9 Επισυναπτόμενο – Σχόλιο

Στον πίνακα Attachment αποθηκεύονται όλα τα σχόλια και έγγραφα που ο χρήστης μπορεί να βάλει πάνω σε έναν πελάτη ή σε μια ειδοποίηση. Ο πίνακας αυτός έχει EntityType και EntityID όπως είδαμε και στον πίνακα Action, για συσχέτιση Attachment με άλλες οντότητες. Τα βασικά πεδία ενός Attachment είναι τα παρακάτω:

- Customer, κωδικός πελάτη, foreign key στον πίνακα Customer
- OpenXmlText, σχόλιο που εισάγει ο χρήστης, που είναι μορφοποιημένο κείμενο. Αυτό αποθηκεύεται σε μορφή OpenXml που υποστηρίζεται από το Microsoft Word και ουσιαστικά είναι XML text.
- FileName, όνομα επισυναπτόμενου αρχείου
- FileData, επισυναπτόμενο αρχείο, που αποθηκεύεται σε μορφή bytes

4.3.10 Αναφορά

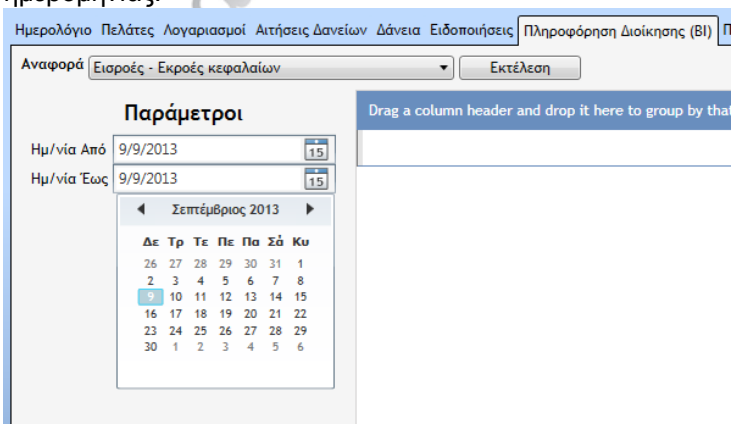
Στον πίνακα Report αποθηκεύονται όλες οι διαθέσιμες αναφορές που έχει ο χρήστης με πρόσβαση «Προσωπικό Επιχειρηματικού Χώρου», στην οθόνη Διοικητική Πληροφόρηση (BI). Αυτή είναι μια πολύ ενδιαφέρουσα υλοποίηση για custom reporting services γιατί δεν απαιτείται καμία αλλαγή στον κώδικα της εφαρμογής για προσθήκη νέων αναφορών ή αλλαγή πάνω σε υπάρχουσα αναφορά. Το ερώτημα (query) που εκτελείται στη βάση αποθηκεύεται στον πίνακα Report και ο κώδικας της εφαρμογής, προσαρμόζει έξυπνα το user interface για την εμφάνιση των αποτελεσμάτων.

Τα βασικά πεδία του πίνακα Report είναι:

- Name, ονομασία αναφοράς
- Query, sql query (ερώτημα) αναφοράς, εδώ αποθηκεύεται ο sql κώδικας του ερωτήματος.

Για τις αναφορές που απαιτούν κάποιες παραμέτρους υπάρχει ο πίνακας ReportParameter του οποίου τα βασικά πεδία είναι τα εξής:

- Report, κωδικός αναφοράς που αφορά η παράμετρος
- Name, Όνομα παραμέτρου
- Description, περιγραφή παραμέτρου που εμφανίζεται στον χρήστη
- Type, τύπος παραμέτρου. Σε αυτό το πεδίο κρατάμε τον τύπο της παραμέτρου (int, string, decimal, DateTime, Boolean κα). Το user interface ανάλογα με τον τύπο της κάθε παραμέτρου εμφανίζει το αντίστοιχο control στον χρήστη για να εισάγει τιμή. Πχ αν μια παράμετρος είναι ημερομηνία τότε εμφανίζει έναν DatePicker για επιλογή ημερομηνίας.



Εικόνα 48: Εισαγωγή τιμής παραμέτρου σε αναφορά

- Required, ένδειξη αν η παράμετρος είναι υποχρεωτική για την εκτέλεση της αναφοράς

4.4 PersistentLib2009 (ORM)

Για την επικοινωνία της εφαρμογής με την βάση δεδομένων, χρησιμοποιήσαμε την βιβλιοθήκη PersistentLib2009, που είναι μια βιβλιοθήκη ανοικτού κώδικα και ουσιαστικά είναι ένα εργαλείο Object-Relational Mapping (ORM). Το ORM είναι μια προγραμματιστική τεχνική για μετατροπή δεδομένων μεταξύ συστημάτων με ασύμβατους τύπους δεδομένων σε μια αντικειμενοστραφή γλώσσα προγραμματισμού.

Υπάρχουν πολλές διαθέσιμες βιβλιοθήκες ανοιχτού και κλειστού κώδικα για ORM, όπως είναι το Entity Framework της Microsoft, όμως πολλοί προγραμματιστές προτιμούν να χρησιμοποιούν δικά τους εργαλεία ORM. Στην περίπτωση της επικοινωνίας μιας σχεσιακής βάσης δεδομένων και ενός αντικειμενοστραφούς προγράμματος, ένα ORM αυτό που κάνει είναι να αντιστοιχεί όλες τις οντότητες της βάσεις (πίνακες κτλ) σε κλάσεις, έτσι μπορεί το πρόγραμμα να διαχειριστεί αποτελεσματικά τα δεδομένα που αντλεί από τη βάση. Επίσης, τα εργαλεία ORM δημιουργούν ένα επίπεδο αφαίρεσης, το οποίο κρύβει από τον προγραμματιστή τις λεπτομέρειες της επικοινωνίας του προγράμματος με τη βάση δεδομένων.

Συγκριτικά με τις παραδοσιακές τεχνικές επικοινωνίας μεταξύ ενός αντικειμενοστραφούς προγράμματος και μιας σχεσιακής βάσης, τα εργαλεία ORM συνήθως μειώνουν το μέγεθος του κώδικα που πρέπει να γραφτεί. Το αρνητικό αυτών των εργαλείων είναι ότι, δημιουργώντας αυτό το υψηλό επίπεδο αφαίρεσης στην επικοινωνία της βάσης με το πρόγραμμα, κρύβονται όλες οι λεπτομέρειες σχετικά με το τι πραγματικά συμβαίνει κατά την επικοινωνία.

Το PersistentLib2009 είναι λοιπόν ένα εργαλείο ORM. Για την επικοινωνία με τη βάση, το PersistentLib2009 κάνει παραγωγή κώδικα (code generation) όλων των κλάσεων για τις οντότητες (πίνακες) που πρέπει να διαχειριστεί η εκάστοτε εφαρμογή. Στις κλάσεις που παράγονται, υπάρχουν όλα τα πεδία του κάθε πίνακα που συμπεριλαμβάνεται στο μοντέλο, καθώς επίσης και η απαραίτητη λειτουργικότητα για insert, update, delete και select για πρόσβαση στα δεδομένα. Η επικοινωνία με τη βάση γίνεται με τον κλασσικό ADO.NET τρόπο που γίνεται σε όλα τα προγράμματα γραμμένα σε C#. Επίσης υπάρχει η δυνατότητα να αποτυπωθούν και οι σχέσεις μεταξύ των πινάκων.

Το PersistenLib2009 έχει γραφτεί με τέτοιο τρόπο ώστε να μπορεί να εξυπηρετήσει την επικοινωνία με οποιουδήποτε τύπου σχεσιακή βάση δεδομένων (Sql Server, MySql, Oracle Sql, IBM DB2 κτλ). Αυτό μπορεί να γίνει, δημιουργώντας απλές κλάσεις οι οποίες υλοποιούν συγκεκριμένα interfaces, οι οποίες ουσιαστικά περιγράφουν τις ιδιαιτερότητες της κάθε βάσης δεδομένων. Η σημαντικότερη ιδιαιτερότητα που πρέπει να περιγραφεί, είναι ουσιαστικά η μετατροπή των τύπων δεδομένων που έχει η κάθε βάση, σε τύπους δεδομένων της γλώσσας C# και αντίστροφα.

Τέλος, να αναφέρουμε ότι το PersistentLib2009 έχει και υποδομή για εξυπηρέτηση 3-tier εφαρμογών (εφαρμογές τριών επιπέδων), δηλαδή Client – Server – Database. Μπορεί να παράγει κώδικα αυτόματα για εξυπηρέτηση κλήσεων σε Web Services πάνω σε κάποιον Server που έχει αναλάβει την επικοινωνία με τη βάση, όπως δηλαδή είναι στημένη η αρχιτεκτονική των κλασσικών πληροφοριακών συστημάτων. Έτσι ο προγραμματιστής μπορεί να φτιάξει 3-tier εφαρμογές πολύ εύκολα και με πολύ λιγότερο κώδικα.

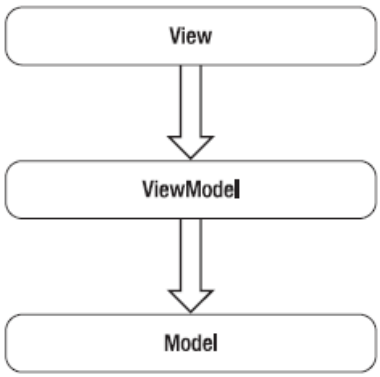
4.5 Model – View – ViewModel (MVVM)

Το MVVM είναι μια μεθοδολογία ανάπτυξης κώδικα (design pattern) για ανάπτυξη εφαρμογών σε WPF ή Silverlight. Ένας από τους κύριους στόχους του είναι να ξεχωρίσει το View, που είναι το user interface, από την λογική της παρουσίασης (presentation logic). Επιπλέον στόχοι είναι να γίνει η λογική παρουσίασης επαναχρησιμοποιήσιμη (reusable) για διαφορετικά user interfaces, μειώνοντας την εξάρτηση του user interface με τον κώδικα, και να επιτρέψει στους σχεδιαστές του user interface να εργάζονται πιο ανεξάρτητα.

Δομικά, μια εφαρμογή MVVM έχει τρία βασικά κομμάτια: το Model (Μοντέλο), το View (Όψη) και το ViewModel.

- Το Model είναι η οντότητα που αντιπροσωπεύει την επιχειρηματική λογική. Μπορεί να είναι οτιδήποτε, από μια απλή οντότητα πελάτη μέχρι μια περίπλοκη οντότητα ανταλλαγής μετοχών.
- Το View είναι το γραφικό control ή σύνολο από control που είναι υπεύθυνο να δείχνει τα δεδομένα του μοντέλου (Model) στην οθόνη. Το View μπορεί να είναι ένα WPF παράθυρο, μια σελίδα Silverlight ή ένα XML data template control.
- Το ViewModel είναι η «μαγεία» πίσω απ' όλα. Το ViewModel περιέχει την λογική του UI, τα commands, τα events και τις αναφορές στο μοντέλο (Model). Στο MVVM, το ViewModel δεν είναι υπεύθυνο για την ενημέρωση των δεδομένων που παρουσιάζονται στο UI, εξ αιτίας του πολύ ισχυρού data binding μηχανισμού του WPF και του Silverlight, και δεν χρειάζεται να το κάνει. Αυτό γίνεται επειδή το View είναι απλά ένας παρατηρητής του ViewModel, οπότε τη στιγμή που αλλάζει το ViewModel, το UI ενημερώνεται αυτόματα.

Ουσιαστικά λοιπόν, το Model είναι η επιχειρηματική λογική, δηλαδή ο κώδικας που χειρίζεται το μοντέλο των δεδομένων που χειρίζεται η εφαρμογή. Είναι υπεύθυνο για την ανάκτηση των δεδομένων, την ενημέρωσή τους και γενικά για την διαχείρισή τους. Το View είναι το UI, δηλαδή τα παράθυρα και διάφορα controls που έχουμε στην εφαρμογή. Το ViewModel είναι αυτός που είναι υπεύθυνος για την λογική του UI και συνδέει το View με το Model.



Εικόνα 49: Τα επίπεδα του MVVM

Το ViewModel που έχουμε υλοποιήσει στην εφαρμογή μας, είναι κλάσεις που περιέχουν όλη τη λογική του user interface. Αυτές οι κλάσεις υλοποιούν το interface με όνομα INotifyPropertyChanged. Αυτό γίνεται για να μπορεί το View να ενημερώνεται αυτόματα το View με τη βοήθεια του μηχανισμού του WPF. Το INotifyPropertyChanged interface, στην πραγματικότητα προδιαγράφει πως η κλάση που θα το υλοποιήσει, θα πρέπει να έχει ένα event το οποίο θα ενεργοποιείται (θα γίνεται fire) κάθε φορά που αλλάζει κάτι στο ViewModel.

Αυτό που κάνει λοιπόν ο μηχανισμός του WPF είναι ότι καταλάβει πως ένα UI έχει κάποιο ViewModel, τότε κοιτάει αυτό το αντικείμενο και αν αυτό υλοποιεί το INotifyPropertyChanged και τότε «κουμπώνει» πάνω σε αυτό το even που προδιαγράφεται, με αποτέλεσμα να μπορεί να «ακούει» τις αλλαγές του UI. Έτσι έχει τη δυνατότητα να μπορεί να ενημερώνει αυτόματα το UI.

Παρακάτω θα δούμε ένα κομμάτι της βασικής ViewModel κλάσης της εφαρμογής που λέγεται Workspace.

```
partial class Workspace : IWorkspace : INotifyPropertyChanged
{
    private string _p_StatusMessage;
    public string _StatusMessage { get { return _p_StatusMessage; } set {
_p_StatusMessage = value; SendPropertyChanged("_StatusMessage"); } }

    private IEmployee_Ex _p_User;
    public IEmployee_Ex _User { get { return _p_User; } set { _p_User = value;
SendPropertyChanged("_User"); } }

    private bool _p_HasConnection;
    public bool _HasConnection
    {
        get { return _p_HasConnection; }
        set { _p_HasConnection = value; SendPropertyChanged("_HasConnection"); }
    }

    public bool UserLogin(int username, string pwd)
    {
        _User = this.ResolveValue<IEmployeeResolver>().Find(username);
        if (_User != null && _User._LoginDetails != null)
        {
            if (_User._LoginDetails._Password == pwd)
            {
                if (_User._Profiles == null || _User._Profiles.Count == 0)
                {
                    _HasConnection = false;
                    _StatusMessage = "Ο χρήστης δεν έχει καμία πρόσβαση στο
σύστημα. Δεν είναι δυνατή η σύνδεση. Στοιχεία χρήστη: " + _User._UserID + ".";
                    _User = null;
                    return false;
                }
                _HasConnection = true;
                _StatusMessage = null;
                ResetUserProfiles();
                BankLib.IoC.IoCContainer.RegisterResolver<IEmployee_Ex>(_User);
                return true;
            }
        }

        _User = null;
        _HasConnection = false;
        _StatusMessage = "Λάθος Όνομα Χρήστη ή Κωδικός";
        return false;
    }

    public event PropertyChangedEventHandler PropertyChanged;
    public void SendPropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
     ……
    }
}
```

Όπως βλέπουμε στον παραπάνω κώδικα, η κλάση μας έχει το event PropertyChanged και επίσης έχουμε ορίσει μια μέθοδο SendPropertyChanged, που κάνει fire αυτό το event. Επίσης μπορούμε να δούμε μερικά από τα properties που έχουν οριστεί στην κλάση όπως το

_HasConnection που είναι η ένδειξη αν ο χρήστης έχει συνδεθεί. Παρατηρούμε ότι στον setter του _HasConnection property, όπως και σε όλα τα properties, γίνεται η κλήση της μεθόδου SendPropertyChanged για να γίνει fire το event και να καταλάβει το UI τις αλλαγές του ViewModel. Τέλος, μπορούμε να δούμε τη λογική του UI που αφορά την είσοδο του χρήστη στο σύστημα, που υλοποιείται από τη μέθοδο UserLogin, όπου ελέγχεται αν υπάρχει ο χρήστης και αν ο κωδικός είναι σωστός και βάζει τον χρήστη στο σύστημα.

4.6 Inversion of Control (IoC)

Το IoC είναι μια μεθοδολογία ανάπτυξης κώδικα (design pattern) που βασίζεται στο Dependency Inversion Principle (αρχή αντιστροφής των εξαρτήσεων). Για την υλοποίηση του IoC σε μια εφαρμογή, χρησιμοποιείται μια τεχνική που λέγεται Dependency Injection (DI). Επειδή υπάρχει μεγάλη σύγχυση με τις ονοματολογίες που χρησιμοποιούνται για να περιγράψουν αυτές τις έννοιες, πολλοί νομίζουν πώς το Inversion of Control, το Dependency Injection και το Dependency Inversion Principle είναι το ίδιο πράγμα. Για καταλάβουμε τη διαφορά μεταξύ τους θα δώσουμε μια απλή περιγραφή στο καθένα που θα ξεκαθαρίσει τα πράγματα:

- Το Dependency Inversion Principle είναι αρχές που χρησιμοποιούνται για την αρχιτεκτονική ανάπτυξης εφαρμογών
- Το Inversion of Control είναι ένα συγκεκριμένο design pattern για την αντιστροφή των εξαρτήσεων
- Το Dependency Injection είναι μια υλοποίηση του IoC για την αντιστροφή των εξαρτήσεων

Στη συνέχεια θα πούμε μερικά λόγια για το Dependency Injection (DI) που θα μας βοηθήσει να καταλάβουμε το IoC design pattern. Το DI δεν είναι ένας στόχος, αλλά είναι ένα μέσο που βοηθάει να πετύχουμε τον στόχο μας. Το DI επιτρέπει την χαλαρή σύνδεση (loose coupling) μεταξύ των διάφορων οντοτήτων μιας εφαρμογής. Το loose coupling κάνει τον κώδικά μας πιο συντηρήσιμο και αυτός είναι ο κύριος στόχος μας. Τα πλεονεκτήματα του loose coupling στον κώδικά μας είναι τα παρακάτω:

- Late binding, οι διάφορες υπηρεσίες μπορούν να αντικατασταθούν με άλλες υπηρεσίες, πράγμα πολύ χρήσιμο σε μεγάλες εφαρμογές όπου το περιβάλλον που τρέχουν είναι σαφώς καθορισμένο
- Επεκτασιμότητα, ο κώδικας μπορεί να επεκταθεί και να επαναχρησιμοποιηθεί με τρόπους που δεν έχουν προγραμματιστεί ή προβλεφτεί από την αρχή
- Παράλληλη ανάπτυξη, ο κώδικας μπορεί να αναπτυχθεί παράλληλα από διαφορετικούς προγραμματιστές, πράγμα πολύ χρήσιμο σε πολύ μεγάλες και περίπλοκες εφαρμογές
- Συντηρησιμότητα (maintainability), κλάσεις με καθαρά ορισμένες ευθύνες είναι ευκολότερο να συντηρηθούν
- Δυνατότητα testing (testability), οι κλάσεις μπορούν να δοκιμαστούν με unit tests

Τι είναι όμως πραγματικά το Dependency Injection; Στον ιστότοπο Stack Overflow παρουσιάζονται απαντήσεις στην ερώτηση «Πώς να εξηγήσεις το Dependency Injection σε ένα παιδί 5 χρονών». Η πιο δημοφιλής απάντηση δόθηκε από τον John Munsh, ο οποίος έκανε μια πολύ καλή παρομοίωση: «Όταν θέλεις να πάρεις κάτι από το ψυγείο μόνος σου, αυτό μπορεί να δημιουργήσει διάφορα προβλήματα. Μπορεί να αφήσεις την πόρτα του ψυγείου ανοιχτή, μπορεί να βρεις κάτι που δεν θέλουν οι γονείς σου να πάρεις. Μπορεί να ψάχνεις για κάτι που δεν υπάρχει ή κάτι που έχει λήξει. Αυτό που πρέπει να κάνεις είναι να δηλώσεις την ανάγκη σου, πχ θέλω κάτι να πιω μαζί με το μεσημεριανό, και εμείς θα φροντίσουμε να έχεις κάτι να πιεις όταν κάτσεις να φας μεσημεριανό».

Αυτό που θέλει να πει, σχετικά με την ανάπτυξη κώδικα σε αντικειμενοστραφείς γλώσσες, είναι ότι οι συνεργαζόμενες κλάσεις (τα παιδιά 5 χρονών) θα πρέπει να βασίζονται σε υποδομές (τους γονείς) για να τους προσφέρουν την υπηρεσία που χρειάζονται.

Όπως είπαμε λοιπόν, το loose coupling κάνει τον κώδικά μας συντηρήσιμο. Αυτό όμως είναι το εύκολο κομμάτι. Το να προγραμματίζουμε με interfaces αντί με πραγματικές υλοποιήσεις (κλάσεις) είναι εύκολο και είναι προαπαιτούμενο για ακολουθήσουμε αυτή την μεθοδολογία. Το πρόβλημα είναι από πού θα προέρχονται τα αντικείμενά που χρησιμοποιεί ο κώδικας. Δεν μπορούμε να φτιάξουμε ένα νέο αντικείμενο έχοντας μόνο ένα interface, χωρίς να κάνουμε άμεση αναφορά σε κάποια υλοποίησή του (κλάση που το υλοποιεί).

Γι' αυτό το λόγο χρησιμοποιούμε μια τεχνική που λέγεται IoC Containers (μπορεί να βρεθεί κι ως DI Containers). Οι IoC Containers είναι βιβλιοθήκες που αυτοματοποιούν τις λειτουργίες που έχουν να κάνουν στη δημιουργία νέων αντικειμένων. Δεν περιμένουμε από έναν IoC Container να κάνει τον κώδικά μας loosely coupled με μαγικό τρόπο. Ένας IoC Container μπορεί να κάνει τη χρήση του Dependency Injection πιο αποτελεσματική αλλά η εφαρμογή μας πρέπει πρώτα να έχει σχεδιαστεί έχοντας στο μυαλό μας τις τεχνικές του DI.

Υπάρχουν πολλές βιβλιοθήκες οι οποίες προσφέρουν υποδομή για IoC Containers και μπορούμε να χρησιμοποιήσουμε στον κώδικά μας. Μερικές από αυτές είναι το Unity της Microsoft, το Castle Windsor και το Ninject. Στην υλοποίησή της εφαρμογής μας χρησιμοποιήσαμε μια προσαρμοσμένη λύση για IoC Containers που διαφέρει λίγο από τους κλασσικούς IoC Containers αλλά τηρεί τις αρχές που έχουμε περιγράψει παραπάνω.

Το πρώτο πράγμα που πρέπει να κάνουμε όταν χρησιμοποιούμε έναν IoC Container είναι στην αρχή της εφαρμογής (entry point) να καταχωρήσουμε ποιες συγκεκριμένες υπηρεσίες θα προσφέρει ο IoC Container για την κάθε λειτουργία. Σε αυτό το σημείο να αναφέρουμε πως για να σπάσουμε την πολυπλοκότητα του IoC Container σε ανεξάρτητα κομμάτια, έχει υλοποιηθεί ένας κεντρικός IoC Container ο οποίος προσφέρει όλους τους επιμέρους IoC Containers για την επίλυση των εξαρτήσεων. Η κάθε οντότητα του κώδικα έχει έναν δικό της IoC Container που είναι υπεύθυνος να προμηθεύσει όποιον χρειάζεται με τα κατάλληλα αντικείμενα. Οι κλάσεις θα ζητούν κάποια λειτουργικότητα και ο container θα τους γυρίζει την κλάση που υλοποιεί αυτή την λειτουργικότητα σε μορφή interface. Στο entry point της εφαρμογής λοιπόν, καταχωρούμε στον IoC Container όλες τις εξαρτήσεις που χρειάζεται να γνωρίζει για να μπορεί να τροφοδοτεί τις κλάσεις με αυτά που χρειάζονται. Παρακάτω παραθέτεται κομμάτι κώδικα που δείχνει αυτή τη λειτουργία.

```
public static void RegisterResolvers(ISqlFactory SqlFactory)
{
    ……
    RegisterResolver<ICustomerResolver>(new CustomerResolver(SqlFactory));
    RegisterResolver<IAccountResolver>(new AccountResolver(SqlFactory));
    RegisterResolver<IApplicationResolver>(new ApplicationResolver(SqlFactory));
    RegisterResolver<ILoanResolver>(new LoanResolver(SqlFactory));
    RegisterResolver<IAccountTypeResolver>(new AccountTypeResolver(SqlFactory));
    RegisterResolver<IBranchResolver>(new BranchResolver(SqlFactory));
    RegisterResolver<IEmployeeResolver>(new EmployeeResolver(SqlFactory));
    RegisterResolver<IAlertResolver>(new AlertResolver(SqlFactory));
    RegisterResolver<IActionResolver>(new ActionResolver(SqlFactory));
    RegisterResolver<IAttachmentResolver>(new AttachmentResolver(SqlFactory));
   …
}
```

Όπως βλέπουμε παραπάνω, καταχωρούνται όλοι οι επιμέρους containers, που τους ονομάζουμε resolvers, που θα χρειαστούμε. Ορίζουμε πως όταν κάποιος θέλει έναν ICustomerResolver τότε θα του επιστρέφεται ένας CustomerResolver που είναι μια υλοποίηση του ICustomerResolver, ο οποίος τροφοδοτεί με νέα αντικείμενα που υλοποιούν το interface ICustomer_Ex. Οπότε, όταν θέλουμε να δημιουργήσουμε έναν νέο πελάτη, θα πρέπει να κάνουμε resolve του ICustomerResolver ο οποίος προδιαγράφει τη μέθοδο Create που μας επιστρέφει έναν νέο πελάτη. Αυτό ουσιαστικά υλοποιείται στο βασικό ViewModel της εφαρμογής που είδαμε στην προηγούμενη ενότητα (MVVM), όπως βλέπουμε παρακάτω.

```
public void AddNewCustomer()
{
    ICustomer_Ex c = this.ResolveValue<ICustomerResolver>().Create();
    if (_Customers == null) _Customers = new
ObservableCollection<ICustomer_Ex>();
    _Customers.Add(c);
    _SelectedCustomer = c;
}
```

Όπως βλέπουμε, το ViewModel ζητά από τον κεντρικό container να του φέρει τον container (resolver) του Customer (ICustomerResolver) και αυτός να του δημιουργήσει έναν πελάτη ICustomer_Ex. Η κλάση CustomerResolver υλοποιεί το ICustomerResolver και αυτή ουσιαστικά φτιάχνει το νέο αντικείμενο. Παρακάτω φαίνεται κομμάτι κώδικα του CustomerResolver.

```
public ICustomer_Ex Create()
{
    var c = new Customer_Ex();
    c.SqlFactory = SqlFactory;
    return c;
}
```

Όπως βλέπουμε, εδώ γίνεται η αναφορά του Customer, και ο CustomerResolver είναι υπεύθυνος να λύσει την εξάρτηση του ViewModel με τον Customer. Αυτό κάνει τον κώδικα επεκτάσιμο και συντηρήσιμο διότι δεν υπάρχει εξάρτηση μεταξύ των οντοτήτων. Αν αλλάξει το ViewModel δεν χρειάζεται να αλλάξει ο Customer και αντίστροφα. Με αυτόν τον τρόπο λύνονται στην εφαρμογή όλες οι εξαρτήσεις μεταξύ των κλάσεων που ουσιαστικά έχουν όλη την υποδομή που χρειάζονται χωρίς να έχουν άμεση σχέση μεταξύ τους.

Τέλος, να πούμε πως προγραμματίζοντας με interfaces και ακλουθώντας το IoC, βοηθάμε την ανεξαρτητοποίηση του user interface από το μοντέλο δεδομένων μας. Η βιβλιοθήκη του UI δεν έχει καμία αναφορά σε τύπους (κλάσεις) που υλοποιούν κάποια λειτουργικότητα, παρά μόνο σε interfaces. Έτσι όταν αλλάξει το μοντέλο δεδομένων δεν θα χρειαστεί καμία αλλαγή στο UI και αυτός είναι και ο στόχος του IoC. Είναι μια μεθοδολογία που μπορεί να εφαρμοστεί σε οποιονδήποτε κώδικα. Μπορεί αρχικά να φαίνεται ότι προσθέτει overhead στον κώδικα μας, αλλά ουσιαστικά όσο μεγαλώνει η εφαρμογή αυτό το overhead δεν είναι τίποτα μπροστά στα πλεονεκτήματα που προσφέρει αυτή η μεθοδολογία.

5. Συμπεράσματα και Μελλοντικές Επεκτάσεις

Έχοντας κάνει ανασκόπηση του πεδίου της εφαρμογής, έχοντας δει τις λύσεις που υπάρχουν στην αγορά για συστήματα CRM και έχοντας δει την συνολικά την υλοποίηση του συστήματος αυτής της εργασίας μπορούμε εύκολα να καταλάβουμε ότι ένα σύστημα CRM μπορεί να βοηθήσει πολύ στην ανάπτυξη μιας οποιαδήποτε επιχείρησης. Ένα σύστημα CRM είναι απαραίτητο για μια επιχείρηση, πόσο μάλλον σε μια τράπεζα, ώστε να μπορεί να γνωρίζει τους πελάτες της και να καταστρώνει τις στρατηγικές της στην αγορά.

Όλες οι σύγχρονες επιχειρήσεις θα πρέπει να έχουν ένα σύστημα CRM, το οποίο θα είναι ικανό να συλλέξει όλες τις χρήσιμες πληροφορίες που αφορούν τους πελάτες της. Το κόστος όμως ενός τέτοιου συστήματος είναι πολύ μεγάλο και οι περισσότερες εταιρίες ανάπτυξης εφαρμογών που παρέχουν προϊόντα CRM στοχεύουν κυρίως σε μεγάλους οργανισμούς, με αποτέλεσμα μικρότερες επιχειρήσεις να μην μπορούν να τα αγοράσουν. Ο βασικός τρόπος κοστολόγησης των εταιριών είναι ανά χρήστη και αυτό αυξάνει ακόμη περισσότερο το κόστος για έναν οργανισμό όπως είναι μια τράπεζα που θέλει να το χρησιμοποιήσει σε όλα τα επίπεδα χρηστών.

Συνοψίζοντας λοιπόν, είδαμε πώς μπορεί να γίνει η υλοποίηση ενός τραπεζικού συστήματος CRM το οποίο μπορεί να ικανοποιήσει βασικές ανάγκες CRM σε μια τράπεζα με αποτελεσματικότητα. Το μεγάλο κόστος των CRM συστημάτων είναι ο λόγος που ωθεί πολλές επιχειρήσεις να αναπτύξουν δικά τους in house συστήματα CRM, τα οποία είναι ικανά να ικανοποιήσουν τις ανάγκες τους. Το κόστος ανάπτυξης και συντήρησης ενός τέτοιου συστήματος είναι σημαντικά μικρότερο από το κόστος αγοράς ενός συνδρομητικού συστήματος CRM, όπως αυτά που υπάρχουν στην αγορά. Αυτό δεν σημαίνει πως η ανάπτυξη ενός συστήματος CRM είναι εύκολη, αλλά είναι δυνατόν να γίνει με κατάλληλη μεθοδολογία και έρευνα.

Οι μελλοντικές επεκτάσεις που θα μπορούσαν να γίνουν στο σύστημά μας είναι πολλές. Στο τραπεζικό κομμάτι της εφαρμογής θα μπορούσαν να υλοποιηθούν επιπλέον υποσυστήματα που υπάρχουν στα τραπεζικά συστήματα, όπως:

- Υποσύστημα πιστωτικών καρτών
- Υποσύστημα επενδυτικών προϊόντων
- Υποσύστημα ασφαλιστικών προϊόντων

Στο κομμάτι του CRM, υπάρχουν διάφορα κομμάτια στα οποία θα μπορούσε να επεκταθεί η εφαρμογή. Μερικά από αυτά είναι τα εξής:

- Ενσωμάτωση Email, όπου το ημερολόγιο του χρήστη θα μπορούσε να συγχρονίζεται με κάποιον mail server (Microsoft Exchange Server κα) και οι χρήστες να μπορούν να στείλουν emails κατευθείαν μέσω της εφαρμογής
- Διαχείριση και πρόβλεψη πωλήσεων, όπου το προσωπικό επιχειρηματικού θα μπορούσε να δει αναλύσεις για την πρόοδο των πωλήσεων και προβλέψεις για μελλοντικές πωλήσεις
- Υποστήριξη στοχοθεσίας, όπου στην αρχή κάθε τριμήνου θα μπορούν να ορίζονται οι στόχοι τριμήνου για πωλήσεις προϊόντων και οι χρήστες θα μπορούν να βλέπουν την πρόοδο του στόχου τους και να επιβραβεύονται αν τον πετύχουν
- Ενσωμάτωση μέσων κοινωνικής δικτύωσης, για να μπορούν οι διοικούντες της τράπεζας να γνωρίζουν τι συζητιέται στο διαδίκτυο για την τράπεζά τους και να έχουν μια εικόνα για την εικόνα του οργανισμού προς τον κόσμο
- Πρόσβαση από άλλες συσκευές όπως smartphones και tablets, αλλά και πρόσβαση μέσω διαδικτύου, ώστε τα στελέχη να μπορούν να έχουν πρόσβαση στο σύστημα οπουδήποτε κι αν βρίσκονται
- Συνοπτική εικόνα του συστήματος μέσω ειδικών διαγραμμάτων και KPIs (Key Performance Indicators). Τα KPIs είναι μετρήσιμα μεγέθη για σημαντικές οντότητες για την επιχείρηση, οι οποίες αποτυπώνονται σε απλά νούμερα και ο χρήστης μπορεί να

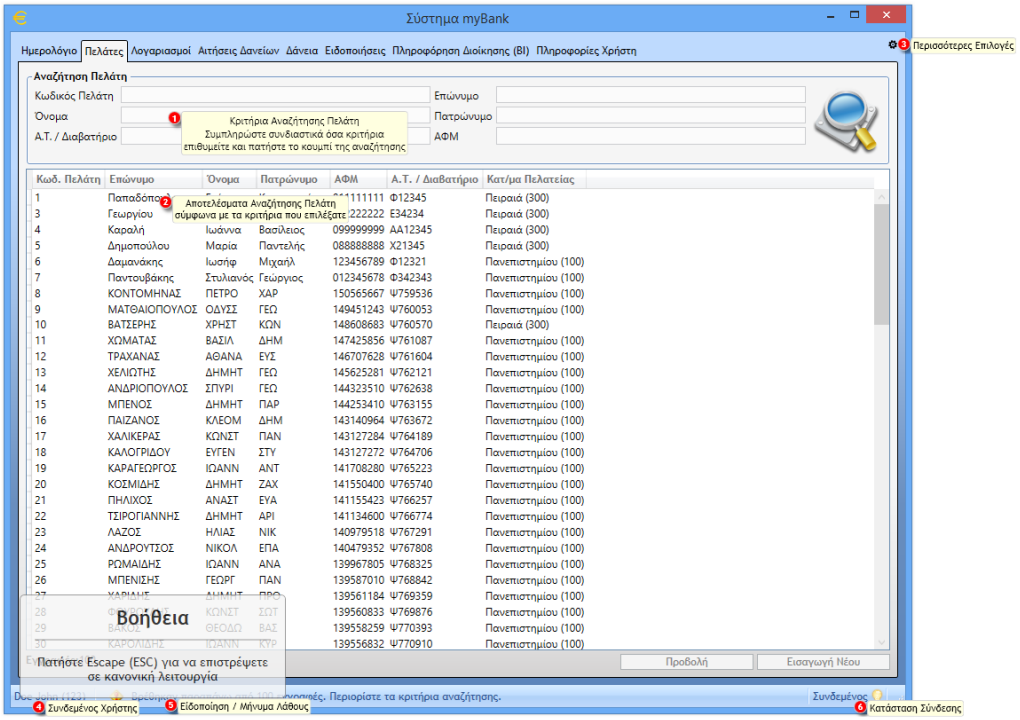
καταλάβει την εικόνα ενός συγκεκριμένου υποσυστήματος της εφαρμογής με μια απλή ματιά. Πχ ένα KPI είναι πόσοι νέοι πελάτες ανοίχτηκαν στο σύστημα τον τελευταίο μήνα ή τρίμηνο, ή το συνολικό ποσό εισροών κεφαλαίων στο σύστημα για κάποιο διάστημα.

Όπως μπορούμε να καταλάβουμε, οι επεκτάσεις ενός τέτοιου συστήματος είναι πάρα πολλές και με την πάροδο του χρόνου αυξάνονται οι ανάγκες των επιχειρήσεων οι οποίες επιβάλλουν την συνεχή επέκταση του συστήματος. Ταυτόχρονα όμως, ωριμάζουν τα συστήματα και μπορούν να είναι πολύ χρήσιμα εργαλεία για μια επιχείρηση, τα οποία είναι σε θέση να εξυπηρετήσουν οποιαδήποτε μηχανογραφική ανάγκη προκύψει.

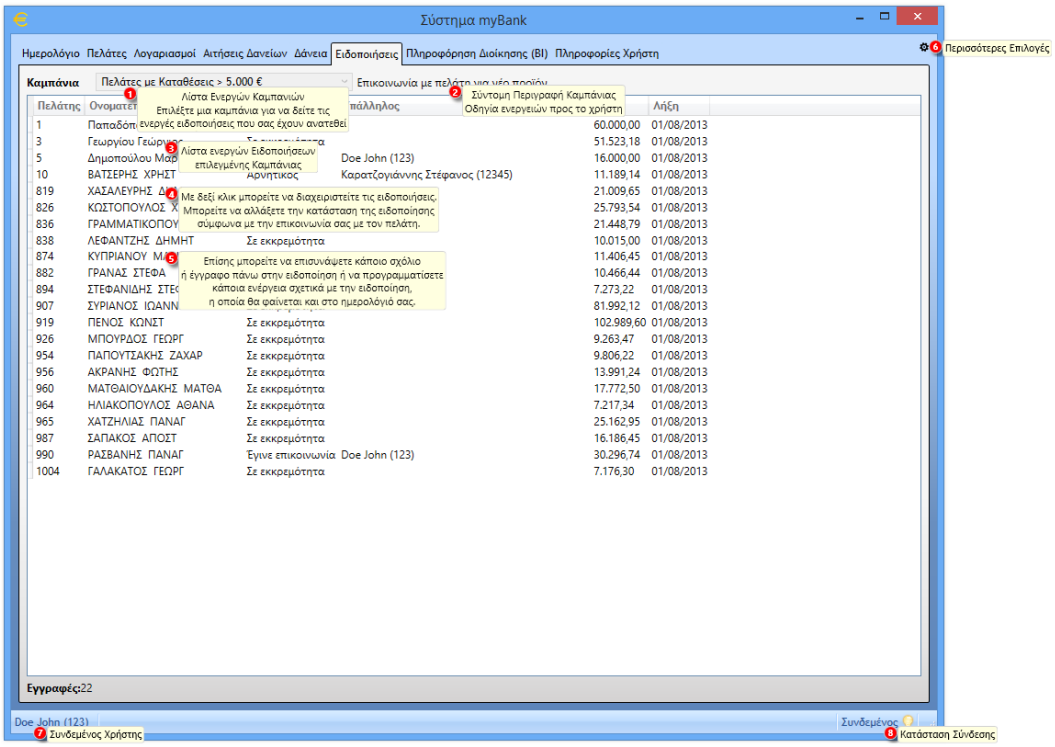
Το προσωπικό που αναπτύσσει αυτά τα συστήματα, θα πρέπει να είναι καταρτισμένο όχι μόνο τεχνολογικά αλλά να μπορεί να καταλάβει και τις επιχειρηματικές ανάγκες των επιχειρήσεων, ώστε να είναι σε θέση να τις ικανοποιήσει, επιλέγοντας πάντα τις κατάλληλες τεχνολογίες που προσφέρουν τη δυνατότητα υλοποίησης κατάλληλων διαδικασιών που θα υποστηρίξουν την επιχείρηση. Όλα αυτά απαιτούν μεγάλη προσπάθεια, συνεχή έρευνα και εμπειρία από τους προγραμματιστές, γι’ αυτό το λόγο ο ρόλος τους στη διαδικασία ανάπτυξης είναι πολύ δύσκολος και απαιτεί πολλές ικανότητες.

6. Παραρτήματα

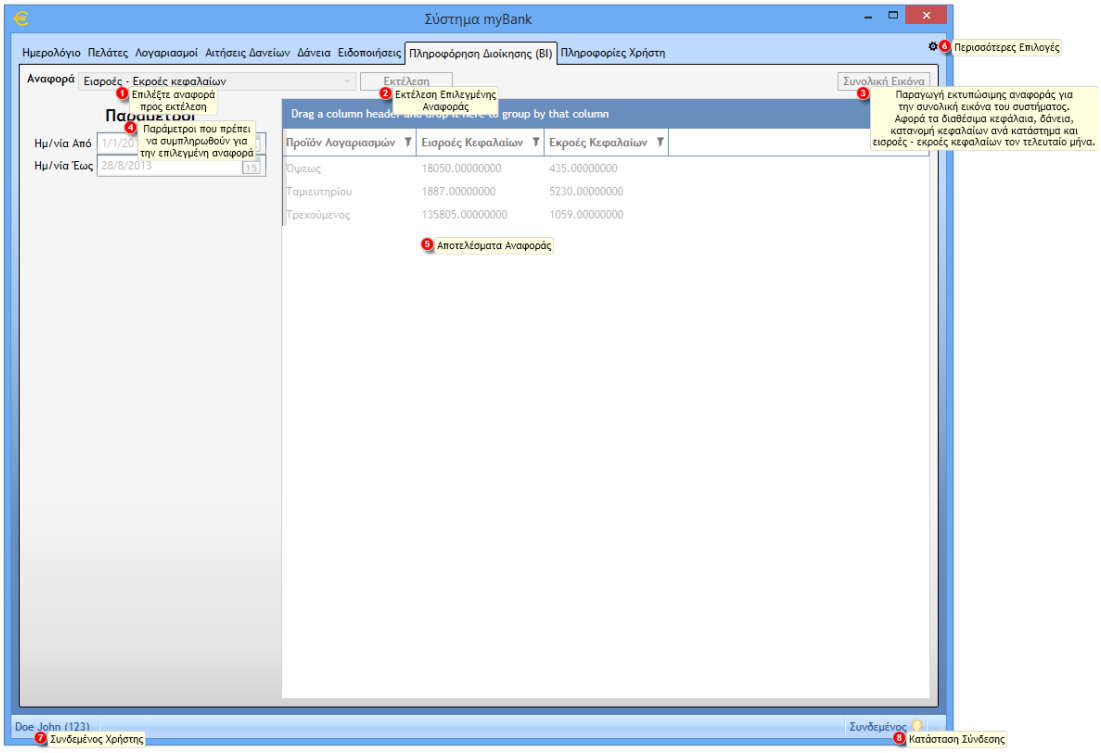
6.1 Οθόνες που υποστηρίζουν λειτουργία βοήθειας



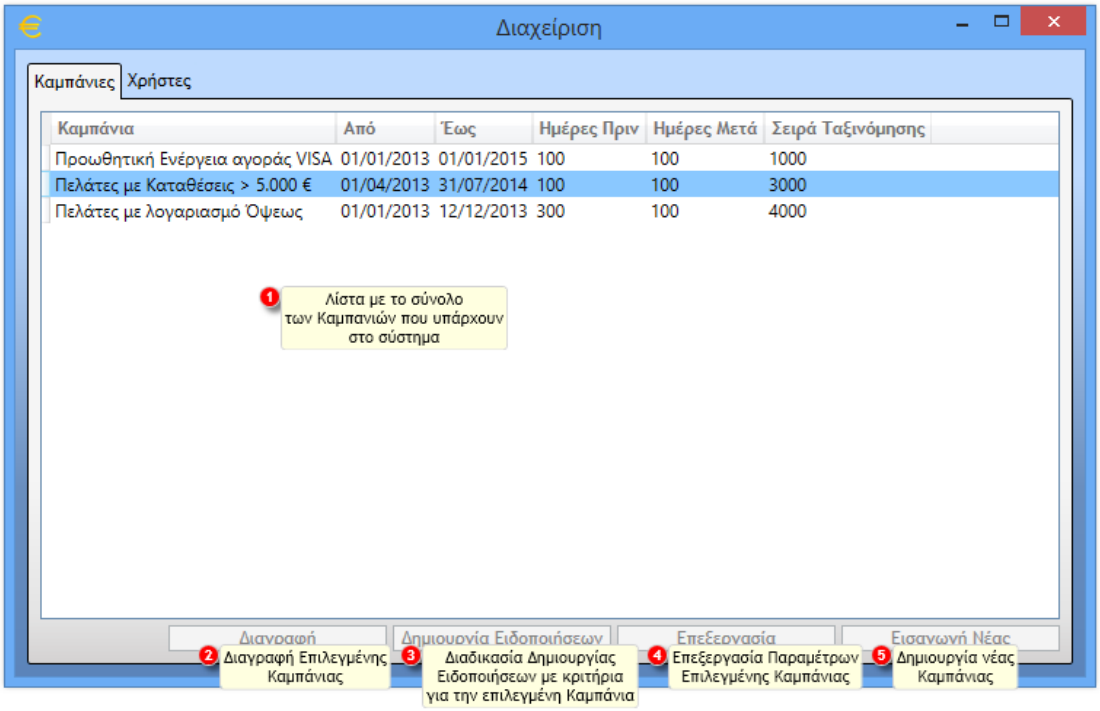
Εικόνα 50: Λειτουργία Βοήθειας Χρήστη Οθόνης Πελατών



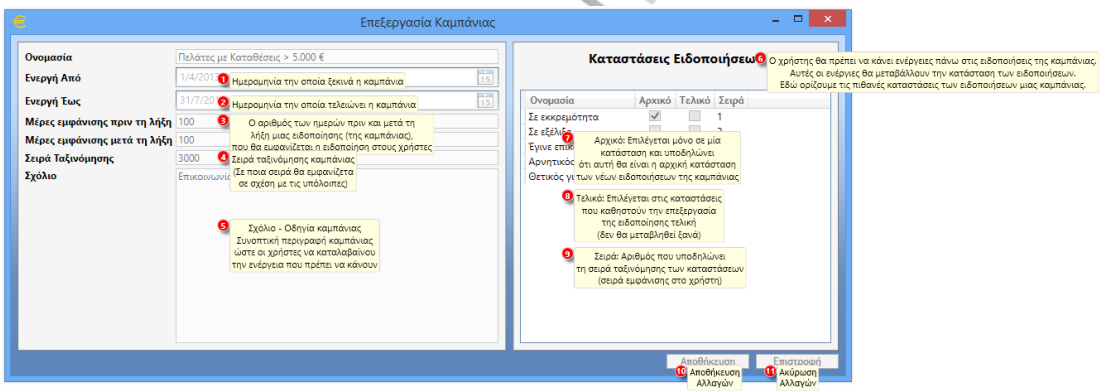
Εικόνα 51: Λειτουργία Βοήθειας Χρήστη Οθόνης Ειδοποιήσεων



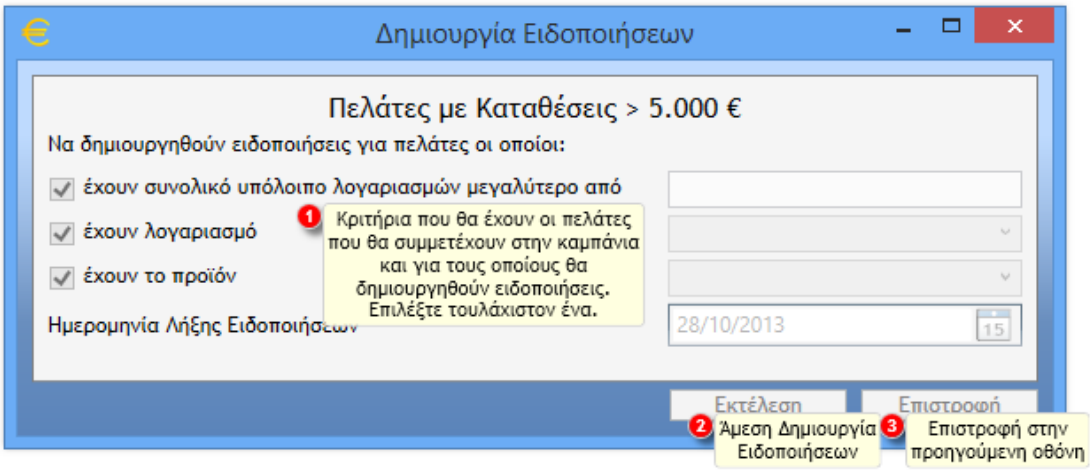
Εικόνα 52: Λειτουργία Βοήθειας Χρήστη Οθόνης Διοικητικής Πληροφόρησης



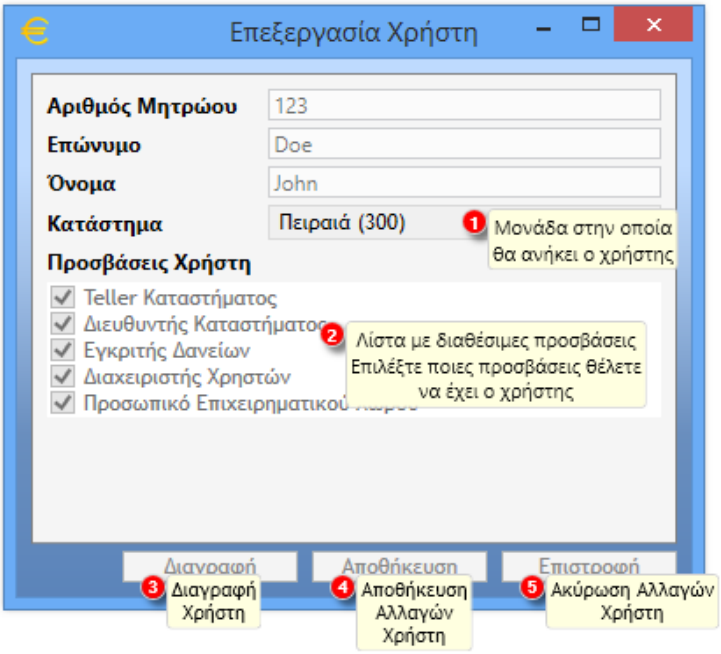
Εικόνα 53: Λειτουργία Βοήθειας Χρήστη Οθόνης Διαχείρισης Καμπανιών



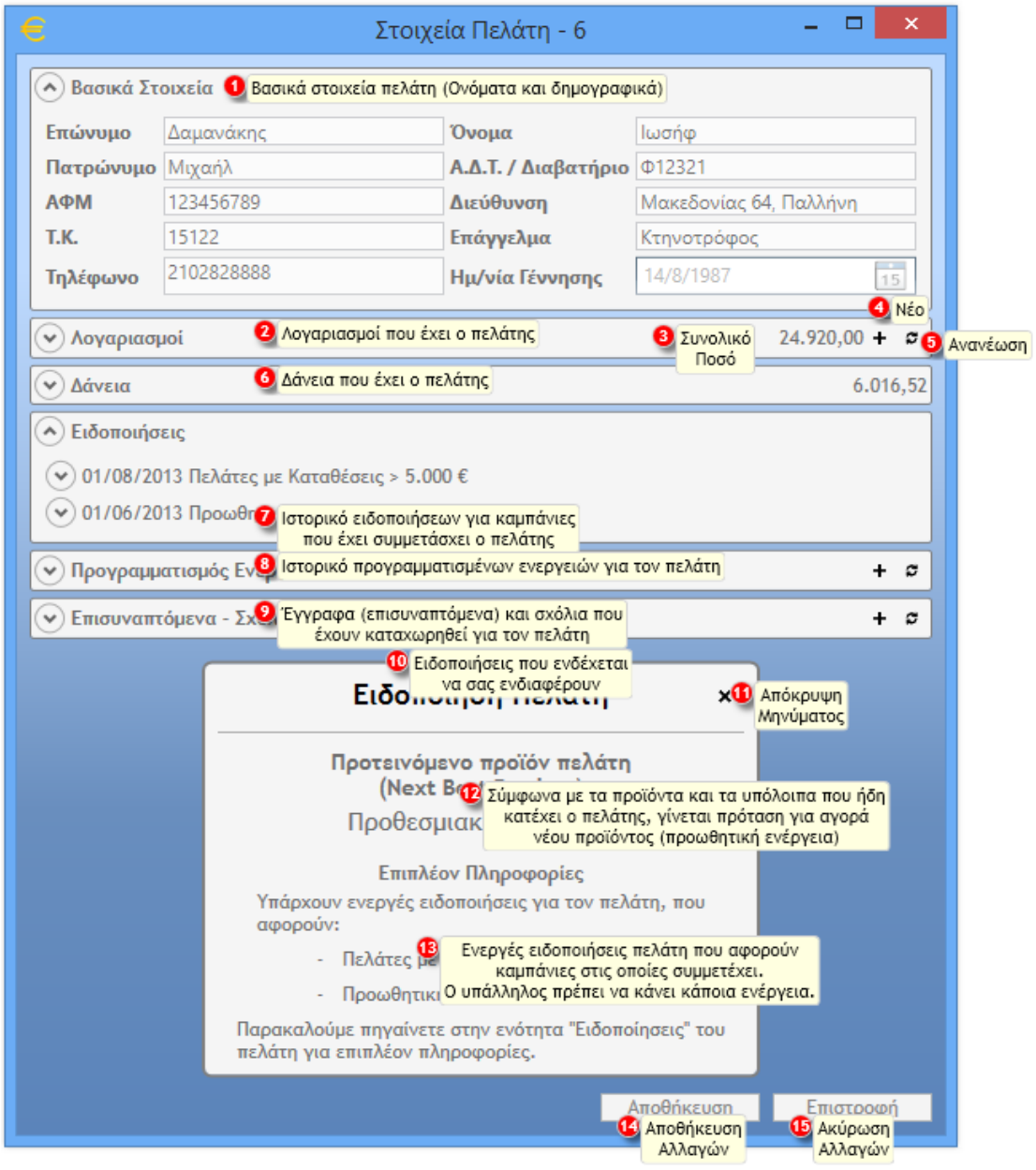
Εικόνα 54: Λειτουργία Βοήθειας Χρήστη Οθόνης Επεξεργασίας Καμπάνιας



Εικόνα 55: Λειτουργία Βοήθειας Χρήστη Οθόνης Δημιουργίας Ειδοποιήσεων



Εικόνα 56: Λειτουργία Βοήθειας Χρήστη Οθόνης Επεξεργασίας Χρήστη και Προσβάσεων



Εικόνα 57: Λειτουργία Βοήθειας Χρήστη Οθόνης Πελάτη

7. Βιβλιογραφία

- Andrew Troelsen (2007), Pro C# 2008 and the .NET 3.5 Platform, Fourth Edition, Apress
- Alex Mackey (2010), Introducing .NET 4.0 With Visual Studio 2010, Apress
- Fabio Claudio Ferrachiati (2008), LINQ for Visual C#, Apress
- Vijay P. Mehta (2008), Pro LINQ Object Relational Mapping with C# 2008, Apress
- Alex Davies (2012), Async in C# 5.0, O' Reilly Media
- Matthew MacDonald (2008), Pro WPF in C# 2008: Windows Presentation Foundation with .NET 3.5, Second Edition, Apress
- Raffaele Garofalo (2011), Applied WPF 4 in Context, Apress
- Applied WPF 4 in Context (2005), Programming Windows Presentation Foundation, O' Reilly Media
- Vijay P. Mehta (2008), Pro LINQ Object Relational Mapping with C# 2008, Apress
- Julia Lerman (2010), Programming Entity Framework, Second Edition, O' Reilly Media
- Raffaele Garofalo (2011), Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern, O' Reilly Media
- Gary McLean Hall (2010), Pro WPF and Silverlight MVVM: Effective Application Development with Model-View-ViewModel, Apress
- Mark Seemann (2012), Depedency Injection in .NET, Manning Publications Co.
- Dhanji R. Prasanna (2009), Dependency Injection, Manning Publications Co.
- Paul Atkinson, Robert Vieira (2012), Beginning Microsoft SQL Server 2012 Programming, John Wiley & Sons, Inc.
- Itzik Ben-Gan (2012), Microsoft SQL Server 2012 T-SQL Fundamentals, O' Reilly Media
- Scott Kostojohn, Mathew Johnson, Brian Paulen (2011), CRM Fundamentals, Apress
- John Metric (2011), Building on SugarCRM, O' Reilly Media
- Mike Snyder, Jim Steger, Brendan Landers (2011), Microsoft Dynamics CRM 2011 Step by Step, Microsoft Press
- CRM (http://www.salesboom.com/products/intro-intro.html)
- CRM (http://en.wikipedia.org/wiki/Customer_relationship_management)
- SugarCRM (http://www.sugarcrm.com)
- Object-relational Mapping (http://en.wikipedia.org/wiki/Object-relational_mapping)
- MVVM Light (http://channel9.msdn.com/blogs/kreekman/techdays-2010-understanding-the-model-view-viewmodel-pattern)
- Dependency Injection video tutorial (http://pluralsight.com/training/Courses/TableOfContents/inversion-of-control)
- SAP CRM (http://www.sap.com/greece/solutions/business-suite/crm/index.epx)
- T24 Temenos (http://www.temenos.com/products/t24/)