Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

# Σχεδίαση και Υλοποιήση RIA Client και κατάλληλης υποδομής σε Εξυπηρετητή για την Διαχείριση Έργων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Δαμόπουλου Δημήτριου

**Επιβλέπων : Ιωάννης Ψαρράς**

Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2010

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

# Σχεδίαση και Υλοποιήση RIA Client και κατάλληλης υποδομής σε Εξυπηρετητή για την Διαχείριση Έργων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Δαμόπουλου Δημήτριου

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

..... ..... .....
.

**Επιβλέπων : Ιωάννης Ψαρράς**

Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2010

……..

**Δαμόπουλος Δημήτριος**

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

**ΠΕΡΙΛΗΨΗ**

Στόχος της παρούσης διπλωματικής εργασίας είναι η σχεδίαση και ανάπτυξη υπολογιστικής εφαρμογής που υλοποιεί μία συγκεκριμένη μεθοδολογία Διαχείρισης Έργου.

Η μεθοδολογία αυτή τυποποιείται από την τεκμηρίωση και την λειτουργικότητα προϋπάρχουσας εφαρμογής που είχε αναπτυχθεί στο παρελθόν για παρόμοιο σκοπό.

Η εφαρμογή ακολουθεί την αρχιτεκτονική πελάτη – εξυπηρετητή: οι τελικοί χρήστες χρησιμοποιούν ένα πρόγραμμα που εκτελείται στον web browser του υπολογιστή τους και το οποίο επικοινωνεί για λογαριασμό τους με έναν εξυπηρετητή για την πραγματοποίηση εργασιών διαχείρισης. Ο εξυπηρετητής αποτελείται από μια βάση δεδομένων με αποθηκευμένες τις πληροφορίες διαχείρισης έργων και από Web Services για την αλληλεπίδραση των πελατών με την βάση δεδομένων.

Κατά την σχεδίαση της εφαρμογής, δώσαμε έμφαση στην θωράκιση της επικοινωνίας πελάτη – εξυπηρετητή από προσπάθειες υποκλοπής των πληροφοριών των έργων, στην πιστή υλοποίηση της καθορισμένης μεθοδολογίας Διαχείρισης Έργων και στην εξασφάλιση συμβατότητας της βάσης δεδομένων με την βάση δεδομένων που χρησιμοποιεί η προϋπάρχουσας εφαρμογή.

**Λέξεις κλειδιά:**

διαχείριση έργων, ανάπτυξη εφαρμογής, RIA, .NET, Silverlight

**ABSTRACT**

Main objective of the present diplomatic thesis is the design and implementation of a software system that applies a certain methodology to the Management of Projects.

This methodology is standardized by the documentation and the functionality of a preexisting system that had been developed in the past for a similar purpose.

The application that we developed follows the server – client architecture: End Users use a web browser plugin which fulfils their management requests by commucating with a server. The server consists of a databased collection of the management information of the projects and of a set of Web Services that the clients use to interact with the database.

The design puts emphasis on the protection of server – client communication from attackers who try to intercept information about the projects, on the exact implementation of the specified Project Management methodology and on the ensuring that the databases of our system and of the preexisting one are compatible.

**Keywords:**

project management, application development, RIA, .NET, Silverlight

# 1. Εισαγωγή

## 1.1 Γενικά Για Την Διαχείριση Έργου

*Η Διαχείριση Έργου είναι το πεδίο μελέτης που ασχολείται με τον σχεδιασμό, την οργάνωση, την εξασφάλιση και την διαχείριση των πόρων ώστε να επιτευχθούν οι αντικειμενικοί στόχοι ενός συγκεκριμένου έργου.*

Όσον αφορά το τι αποτελεί έργο, ένας σαφής και διαδεδομένος ορισμός εργασίας είναι ο εξής: *Έργο είναι ένα προσωρινό εγχείρημα που στοχεύει στην δημιουργία ενός πρωτότυπου προϊόντος ή υπηρεσίας* (1).

Υπονοείται ήδη από τον αρχικό αυτό ορισμό του έργου πως η διαχείρισή του έχει να λάβει υπόψιν της κάποιους χρονικούς περιορισμούς. Άλλης φύσεως περιορισμοί προκύπτουν από το μέγεθος του οικονομικού προϋπολογισμού που διατίθεται για τις ανάγκες του έργου, καθώς βέβαια και από το συγκεκριμένο αντικείμενο του έργου.

Οι τρεις αυτές γενικές κατηγορίες περιορισμών αναφέρονται μαζί ως τρίγωνο διαχείρισης του έργου, όπου κάθε πλευρά αντιπροσωπεύει έναν περιορισμό (2). Αλλαγή στη μια πλευρά του τριγώνου μπορεί να προκαλέσει αλλαγή στους περιορισμούς που σχετίζονται με τους άλλους παράγοντες. Έτσι, αλλαγή στο αντικείμενο των εργασιών του έργου αναπροσδιορίζει τους περιορισμούς του χρόνου και του κόστους πχ αύξηση της διάρκειας του έργου, ή αύξηση του προϋπολογισμού.

Λαμβάνοντας υπόψιν αυτές τις παρατηρήσεις, οδηγούμαστε σε έναν αναλυτικότερο ορισμό του έργου:

*Έργο είναι ένα εγχείρημα κατά το οποίο ανθρώπινοι πόροι, μηχανές, οικονομικοί πόροι και πρώτες ύλες οργανώνονται κατά καινοφανή τρόπο, με στόχο την ανάληψη συγκεκριμένου αντικειμένου εργασιών που έχουν ορισμένες προδιαγραφές και υπόκεινται σε δεδομένους κοστολογικούς και χρονικούς περιορισμούς, ώστε να παραχθεί μια επωφελής μεταβολή η οποία ορίζεται μέσω ποσοτικών και ποιοτικών στόχων* (3) .

## 1.2 Διαδεδομένες Μεθοδολογίες Διαχείρισης Έργου

Στην ενότητα αυτή γίνεται μια σύντομη αναφορά σε μερικές από τις πλέον δημοφιλείς στην πράξη μεθοδολογίες που έχουν προταθεί για την διαχείριση έργου. Καθεμιά τους προσεγγίζει το πρόβλημα από διαφορετική σκοπιά,

προτείνει την δική της φιλοσοφία και επιλέγει να τονίσει περισσότερο ή λιγότερο διαφορετικά ζητήματα της διαχείρισης.

Γενικά, η πληθώρα των επιλογών, των οποίων η αξία δοκιμάζεται συνεχώς στην πράξη και η καταλληλότητα τους φαίνεται πως εξαρτάται πάρα πολύ από την συγκεκριμένη φύση του έργου, δείχνει πως το πρόβλημα της διαχείρισης είναι δύσκολο και σύνθετο.

Να παρατηρήσουμε πως πρόκειται για γενικά μοντέλα διαχείρισης, τα οποία γνωρίζουν αρκετές παραλλαγές και εξειδικεύσεις για επιμέρους κατηγορίες έργων. Μιας και για μας έχει ιδιαίτερο ενδιαφέρον η διαχείριση έργων λογισμικού, θα αναφερθούμε ιδιαίτερα στον τρόπο που εφαρμόζονται οι μεθοδολογίες σε τέτοια έργα.

### 1.2.1 Παραδοσιακή Προσέγγιση

Η παραδοσιακή προσέγγιση –παραδοσιακή τόσο υπό την έννοια πως τυποποιήθηκε πριν από τις υπόλοιπες όσο και ως προς το γεγονός πως βασίζεται σε παλιές ιδέες και τεχνικές, δίνει έμφαση στις φάσεις από τις οποίες περνάει η κατασκευή ενός έργου και στην σειρά μετάβασης από την μία φάση στην άλλη.

Διακρίνει 5 φάσεις στην ανάπτυξη ενός έργου:

1) Αρχικοποίηση έργου (Initiation)
2) Σχεδιασμός (Planning and Design)
3) Εκτέλεση (Execution)
4) Έλεγχος και Αξιολόγηση (Monitoring and Controlling)
5) Ολοκλήρωση (Completion)

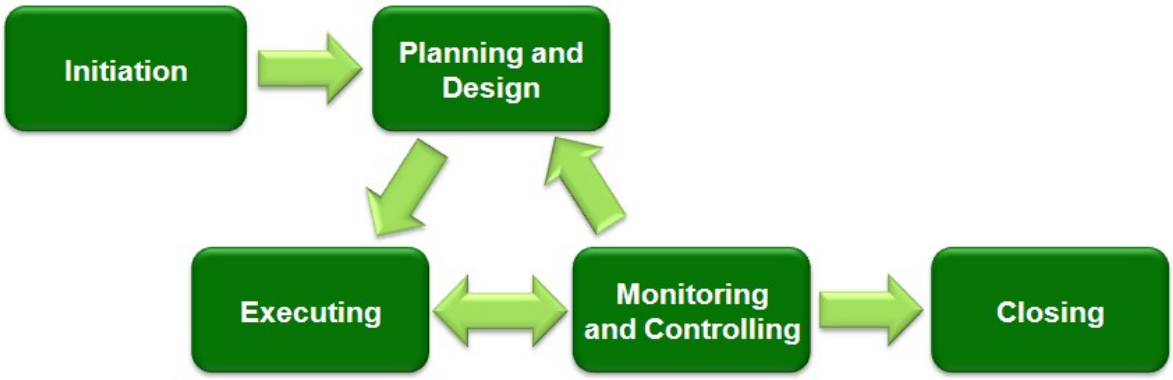
Η τέταρτη φάση, αυτή του ελέγχου και της αξιολόγησης, μπορεί να οδηγήσει είτε στην ολοκλήρωση του έργου (πέμπτη φάση), είτε στον επανασχεδιασμό του (δεύτερη φάση) είτε στην εκ νέου εκτέλεσή του (τρίτη φάση).

Η μεθοδολογία αυτή προέρχεται από πρακτικές που εφαρμόζονται με επιτυχία στις βιομηχανίες κατασκευής βιομηχανικών έργων. Η εφαρμογή της σε έργα λογισμικού έχει καθιερωθεί να καλείται μοντέλο του Καταρράκτη (Waterfall Model).

Έχει ενδιαφέρον πως η πρώτη περιγραφή του μοντέλου του καταρράκτη γίνεται σε ένα άρθρο του 1970, στην οποία όμως ο συγγραφέας ούτε χρησιμοποιεί τον όρο Waterfall Model, ούτε το αναφέρει σαν μια

ενδεικνυόμενη στρατηγική για την διαχείριση έργου. Γενικότερα, στην βιβλιογραφία το μοντέλο αυτό αναφέρεται μόνο σαν κριτική σε διαδεδομένες πρακτικές[1] .

Η κριτική που δέχεται αυτή η προσέγγιση επικεντρώνεται κυρίως στην δυσκολία του αρχικού σχεδιασμού του έργου, ειδικά στις περιπτώσεις εκτενών έργων και έργων των οποίων το τελικό προϊόν έχει ιδιότητες που δεν μπορούν να είναι γνωστές με σαφήνεια από την αρχή. Κάτι τέτοιο είναι αρκετά συνηθισμένο στην βιομηχανία λογισμικού, μιας και εκεί η ανάπτυξη έχει συχνά στόχο την δημιουργία ενός πρωτότυπου προϊόντος. Η μερική ασάφεια τόσο όσον αφορά τα χαρακτηριστικά της τελικής εφαρμογής που θα αναπτυχθεί, έχει σαν συνέπεια μια πολύ μεγαλύτερη ασάφεια στον αρχικό σχεδιασμό. Για να πετύχει η μεθοδολογία σε τέτοια έργα, είναι απαραίτητος ένας πολύ προσεκτικός ορισμός των απαιτήσεων του προϊόντος (διαδικασία που αναφέρεται ως Διαχείριση Απαιτήσεων, Requirements Management).




Εικόνα 1

### 1.2.2 Κρίσιμη Αλυσίδα (Critical Chain)

Η προσέγγιση αυτή επισημαίνει πως οι πόροι της οντότητας (για παράδειγμα, μιας επιχείρησης) που καλείται να ολοκληρώσει ένα έργο, ανθρώπινοι και υλικοί, είναι περιορισμένοι σε αριθμό και σε δυνατότητες (το (4) είναι η καθιερωμένη αναφορά).

Βασική στη μεθοδολογία αυτή είναι η έννοια της κρίσιμης αλυσίδας: πρόκειται για την πολύ βασική ακολουθία των εργασιών του έργου, χωρίς την

[1] Τόσο από το ιδιαίτερης επιρροής άρθρο του Winston Royce που αναφέραμε στην αρχή (16), όσο και μεταγενέστερες αναφορές σε αυτό, για παράδειγμα στα (18) και (17).

εκτέλεση των οποίων όχι μόνο δεν είναι δυνατή η ολοκλήρωση του έργου, αλλά ούτε και η εξέλιξή του προς επόμενα στάδια.

Η κύρια παρατήρηση της μεθοδολογίας είναι πως η διεκπεραίωση των εργασιών της κρίσιμης αλυσίδας απαιτεί την απασχόληση ενός συγκεκριμένου αριθμού πόρων, των οποίων η διαθεσιμότητα θα είναι επομένως περιορισμένη για όσο χρονικό διάστημα διαρκεί η εργασία. Το μοντέλο λοιπόν, καταπιάνεται με το πώς θα διατεθούν οι πόροι στις εργασίες ούτως ώστε να μεγιστοποιηθεί ο μέσος αριθμός των κρίσιμων εργασιών που ολοκληρώνονται ανά πάσας στιγμή, και προτείνει μια αρκετά λεπτομερή μεθοδολογία για την επίτευξη αυτής της βελτιστοποίησης.

### 1.2.3 Agile Management

Η μεθοδολογία αυτή γεννήθηκε από την προσπάθεια να ξεπεραστούν οι αδυναμίες των παραδοσιακών προσεγγίσεων στις περιπτώσεις που οι απαιτήσεις του έργου αλλάζουν διαρκώς και με ταχύτητα. Κάτι τέτοιο σίγουρα συμβαίνει σε αρκετές κατηγορίες σύγχρονων προϊόντων λογισμικού.

Καταρχήν, η μεθοδολογία αυτή δεν δίνει τόσο έμφαση στην ολοκλήρωση του προϊόντος όσο στην συνεχή εξέλιξη του με την προσθήκη καινούργιων χαρακτηριστικών και την βελτίωσή του σε διαδοχικές εκδόσεις. Σε σύγκριση με παρόμοιες μεθοδολογίες που επίσης προτείνουν την επαναληπτική εξέλιξη του έργου μέσα από διαδοχικές εκδόσεις, η agile φιλοσοφία διαφέρει στο ότι προωθεί την πραγματικά πολύ γρήγορη δημιουργία διαδοχικών εκδόσεων, της τάξης των μερικών εβδομάδων αντί για αρκετών μηνών.

Το μοντέλο αυτό, όπως παρατηρήσαμε, ταιριάζει ιδιαίτερα σε αρκετά έργα λογισμικού. Στις περιπτώσεις τέτοιων έργων, το μοντέλο είναι γνωστό ως Extreme Programming. Μιας και συγκεντρώνει αρκετό ενδιαφέρον τα τελευταία χρόνια, θα κάνουμε εδώ μια μικρή αναφορά σε μερικές ενδεικτικές προτάσεις που κάνει για την διαχείριση έργων λογισμικού:

- Η συγγραφή κώδικα σε ζευγάρια προγραμματιστών. Αντί να εργάζεται ο καθένας μόνος του, λειτουργεί υπό την επίβλεψη ενός δεύτερου (με τον οποίον πιθανώς πιο μετά μπορεί να εναλλάξουν ρόλους), ο οποίος απλώς παρακολουθεί την δουλειά του πρώτου και συμβουλεύει και βοηθά όποτε το κρίνει απαραίτητο. Έχει διαπιστωθεί πως αυτή η πρακτική έχει πολλά θετικά αποτελέσματα στην ποιότητα του κώδικα που παράγεται.
- Η συστηματική συγγραφή unit tests του κώδικα (Εικόνα 2). Αυτά είναι κομμάτια κώδικα επιπλέον αυτού που παρέχει την πραγματική

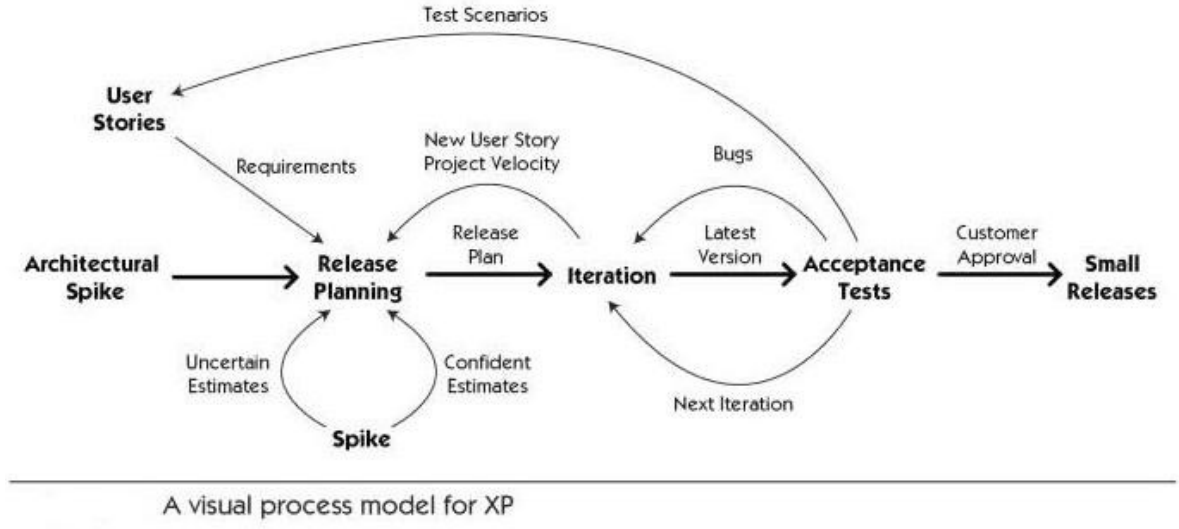
λειτουργικότητα στο τελικό προϊόν, τα οποία αναλαμβάνουν να ελέγξουν, στο βαθμό που είναι εφικτό και χρονικά συμφέρον, πως κάποια συγκεκριμένη προγραμματιστική μονάδα (μια κλάση ή μια συνάρτηση) συμπεριφέρεται ακριβώς με τον τρόπο που προβλέπεται. Το βασικό είναι πως τα unit tests θα πρέπει να γραφτούν πριν από τον κώδικα που καλούνται να ελέγξουν και να κρατούνται σε συγχρονισμό με τις αλλαγές στις απαιτήσεις.

- Υλοποίηση χαρακτηριστικών όταν αυτά απαιτηθούν, όχι νωρίτερα.
- Επίπεδη (δηλαδή, όχι ιεραρχική) δομή των ρόλων των μελών της ομάδας που εκπονεί το έργο. Για παράδειγμα, οι προγραμματιστές δεν ασχολούνται μόνο με την συγγραφή κώδικα, αλλά μπορεί να έχουν λόγο και στην σχεδίαση της εφαρμογής.
- Απλός και σαφής κώδικας.
- Ετοιμότητα για σχεδιαστικές αλλαγές, όταν οι απαιτήσεις αλλάζουν και το πρόβλημα γίνεται καλύτερα κατανοητό.
- Συχνή επικοινωνία μεταξύ των μελών της ομάδας και με τον πελάτη.

Οι φάσεις της διαχείρισης έργου με μια Agile μεθοδολογία φαίνονται διαγραμματικά στην Εικόνα 3.[2]


Εικόνα 2

[2]Υπάρχουν αρκετές πηγές που περιγράφουν την μεθοδολογία. Γενικά για την Agile διαχείριση έργου, γνωστό είναι το (8). Για το Extreme Programming, οι Online αναφορές είναι πραγματικά πολλές· προτείνουμε το κατατοπιστικό άρθρο του Martin Fowler (9).




Εικόνα 3

## 1.2.4 Event Chain

Η μεθοδολογία Event Chain είναι εξέλιξη της Κρίσιμης Αλυσίδας. Χαρακτηριστικό της είναι πως επιχειρεί να μοντελοποιήσει την αβεβαιότητα που υπάρχει στις εκτιμήσεις που γίνονται στην διαχείριση έργου.
Η μεθοδολογία αναγνωρίζει πως οι εργασίες ενός έργου δεν είναι συνεχείς ομοιόμορφες διαδικασίες, αλλά επηρεάζονται από εξωτερικά γεγονότα, τα οποία αλλάζουν την κατάσταση των εργασιών. Το πότε θα συμβούν αυτά τα γεγονότα αντιμετωπίζεται στα πλαίσια της μεθοδολογίας σαν μια τυχαία μεταβλητή. Επίσης, πέρα από το να αλλάζουν την κατάσταση των εργασιών, τα γεγονότα μπορούν να πυροδοτήσουν νέα γεγονότα, δημιουργώντας αλυσίδες γεγονότων.
Μόλις οριστούν με ακρίβεια τα γεγονότα ενός έργου και οι αλυσίδες που σχηματίζουν, μπορεί να εκτελεστεί μια υπολογιστική προσομοίωση ώστε να εκτιμηθούν τα συνολικά αποτελέσματα των γεγονότων στις εργασίες του έργου.
Χρησιμοποιώντας την προσομοίωση και εξετάζοντας τις συσχετίσεις μεταξύ των βασικών παραμέτρων του έργου, μπορούν να εντοπιστούν εκείνα τα μεμονωμένα γεγονότα και εκείνες οι αλυσίδες γεγονότων που επηρεάζουν περισσότερο τις καταστάσεις των εργασιών του έργου. Αυτά καλούνται κρίσιμα γεγονότα και κρίσιμες αλυσίδες γεγονότων.
Η μεθοδολογία προβλέπει μια τεχνική για τον μετριασμό των ανεπιθύμητων επιπτώσεων των γεγονότων, ιδιαίτερα των κρίσιμων. Συγκεκριμένα, συνδέει κάθε τέτοιο γεγονός με ένα σχέδιο αλλαγής του χρονοπρογραμματισμού των εργασιών του έργου, ώστε η εκτέλεση των εργασιών να επηρεάζεται λιγότερο από την εμφάνιση του γεγονότος (ή της αλυσίδας γεγονότων). Αυτά τα σχέδια

μεταβολής του χρονοπρογραμματισμού μπορούν επίσης να μοντελοποιηθούν μαζί τα γεγονότα, και να οδηγήσουν σε μια υπολογιστική πρόβλεψη της πορείας του έργου.
Η μεθοδολογία περιγράφεται με λεπτομέρεια στο (5).

### 1.2.5 PRINCE2

Το PRINCE2 είναι ένα πολύ λεπτομερές πρότυπο διαχείρισης, ελέγχου και οργάνωσης έργων, το οποίο είναι αρκετά διαδεδομένο σε χώρες της Κεντρικής Ευρώπης.

Η μεθοδολογία προσεγγίζει την διαχείριση του έργου με δομημένο τρόπο, αναλύοντας το σε καλά ορισμένες δραστηριότητες. Κάθε δραστηριότητα ορίζεται επακριβώς τεκμηριώνοντας τις βασικές εισόδους της και εξόδους της, τους στόχους της και τα επιμέρους βήματα που πρέπει να ακολουθηθούν για την ολοκλήρωσή της.

Συνολικά, το PRINCE2 ορίζει 40 ξεχωριστές δραστηριότητες, οργανωμένες σε 7 διαδικασίες: την έναρξη του έργου, την αρχικοποίησή του, την διεύθυνσή του, των έλεγχο των καταστάσεων του έργου, την διαχείριση του έργου κατά την μετάβασή του από τη μια κατάσταση στην άλλη, την διαχείριση της παράδοσης του τελικού προϊόντος, και του τερματισμού του έργου.

Επίσης, το PRINCE2 τυποποιεί μεθόδους για την οργάνωση του ανθρώπινου δυναμικού και διαδικασίες για την περίπτωση που κάτι στην ανάπτυξη του έργου εξελιχθεί διαφορετικά από ότι αναμενόταν.

Η πιο πρόσφατη έκδοση του PRINCE2, δημοσιευμένη το 2009 προτυποποιείται από το (6). Σημαντικό είναι και το έγγραφο (7).

## 1.3 Εργασίες Έργου

*Η Εργασία Έργου είναι μια δραστηριότητα που απαιτείται να ολοκληρωθεί μέσα σε ένα ορισμένο χρονικό διάστημα.* Οι εργασίες ενός έργου μπορούν να θεωρηθούν ως τα επιμέρους «εγχειρήματα» στα οποία αναλύεται ένα έργο και των οποίων η σωστή και έγκαιρη διευθέτηση οδηγεί στην επιτυχή ολοκλήρωση του έργου.

Όπως είδαμε προηγουμένως, η διαχείριση ειδικά των εργασιών διαδραματίζει πολύ βασικό ρόλο σε αρκετές μεθοδολογίες διαχείρισης έργου.

Στην διαχείριση, το ζητούμενο για κάθε εργασία είναι ο σαφής ορισμός τους και η καταγραφής τους.

Αυτό, σε πρώτη φάση, περιλαμβάνει για κάθε εργασία την συλλογή πληροφοριών για τον στόχο της και τους πόρους που απαιτεί για την εκτέλεσή της.

Σε δεύτερη φάση, προσδιορίζεται το χρονικό διάστημα μέσα στο οποία αναμένεται να εκτελεστεί, και προγραμματίζεται η ανάθεσή της σε συγκεκριμένα μέλη της ομάδας του έργου.

## 1.4 Παραδοτέα

*Ως όρος παραδοτέο αναφέρεται σε οποιαδήποτε απτό ή αφηρημένο αντικείμενο που παράγεται κατά την πορεία ενός έργου και που προορίζεται να προς τον πελάτη.*

Παραδοτέο μπορεί να είναι μια αναφορά, ένα κείμενο τεκμηρίωσης ή κάποιο δομικό κομμάτι του συνολικού έργου. Ένα πιο αφηρημένο παραδοτέο, μπορεί να είναι για παράδειγμα η αύξηση του κέρδους μιας εταιρίας.

Σε έργα ανάπτυξης λογισμικού, τα παραδοτέα μπορούν να κατηγοριοποιηθούν σε παραδοτέα software, hardware και τεχνικών αναφορών.

## 1.5 Στελέχωση

Η εκπόνηση των εργασιών ενός έργου απαιτεί την απασχόληση και τον συντονισμό κάποιου αριθμού ατόμων. Σε μεγάλα έργα, πιθανώς να χρειάζεται μάλιστα η συμμετοχή αρκετών από αυτά στην διαχείριση.

Γενικά, στην στελέχωση ενός έργου η σπουδαιότερη εργασία είναι ο ορισμός των ρόλων που μπορεί να έχουν τα μέλη της ομάδας του έργου, και ανάθεση τους στον καθέναν.

## 1.6 Ορισμός του Προβλήματος

Θέμα της παρούσας διπλωματικής εργασίας είναι η υλοποίηση μιας συγκεκριμένης μεθοδολογίας Διαχείρισης Έργων Λογισμικο, όπως αυτή περιγράφεται από την τεκμηρίωση και την λειτουργικότητα μιας προϋπάρχουσας εφαρμογής που είχε αναπτυχθεί στο παρελθόν για τον ίδιο σκοπό. Αναλυτική παρουσίαση της εν λόγω μεθοδολογίας γίνεται στο επόμενο

κεφάλαιο. Στη συνέχεια αυτής της ενότητας, θα απαριθμήσουμε τις λειτουργικές και μη απαιτήσεις τις εφαρμογής και θα περιγράψουμε σύντομα την προϋπάρχουσα εφαρμογή.

### 1.6.1 Απαιτήσεις Λογισμικού

Γενική περιγραφή της εφαρμογής[3]

- *Σύνοψη:*

  Η εφαρμογή θα χρησιμοποιείται στην διαχείριση έργων. Η μεθοδολογία διαχείρισης είναι ορισμένη εκ των προτέρων, και περιγράφεται από την λειτουργικότητα προϋπάρχοντος συστήματος που χρησιμοποιείται για τον ίδιο σκοπό και από την τεκμηρίωσή του.

  Η τελική εφαρμογή θα πρέπει να αποτελείται από τρία μέρη:

  1) Μία βάση δεδομένων στην οποία αποθηκεύονται στοιχεία για διάφορα έργα και διαχειριστικές πληροφορίες για αυτά.
  2) Ένα σύνολο υπηρεσιών για την απομακρυσμένη προσπέλαση της βάσης δεδομένων, ομαδοποιημένων με τρόπο που να βοηθά την διαχείριση των έργων σύμφωνα με την συγκεκριμένη μεθοδολογία.
  3) Μια διαδραστική ξεχωριστή εφαρμογή, η οποία θα εκτελείται στον web browser του τελικού χρήστη, και θα του επιτρέπει να επικοινωνεί με τις προηγούμενες υπηρεσίες και να πραγματοποιεί τις εργασίες διαχείρισης που είναι εξουσιοδοτημένος να κάνει. Το user interface θα πρέπει να είναι παραπλήσιο με εκείνο του προϋπάρχοντος συστήματος.

- *Γενικές απαιτήσεις ως προς τις λειτουργίες:*

  Οι λειτουργίες της εφαρμογή πελάτη θα πρέπει να αυτές της προϋπάρχουσας εφαρμογής.

- *Χαρακτηριστικά χρηστών:*

[3] Στην παρουσίαση των απαιτήσεων, ακολουθούμε την δομή που προτείνει το (13)

Οι χρήστες της εφαρμογής είναι τα μέλη των ομάδων διεκπεραίωσης των έργων. Κάποιοι από αυτούς έχουν λιγότερα ή περισσότερα δικαιώματα διαχείρισης των έργων, ανάλογα με τον ρόλο τους στο εκάστοτε έργο, και τον γενικό ρόλο τους σε όλο το σύστημα.

Απαιτήσεις ως προς τις εξωτερικές διασυνδέσεις της εφαρμογής

- *Αλληλεπίδραση με τους χρήστες*:

  Οι χρήστες της εφαρμογής είναι τα μέλη των ομάδων διεκπεραίωσης των έργων. Ο καθένας από αυτούς θα πρέπει να έχει πρόσβαση και, για τα έργα στα οποία συμμετέχει, να μπορεί πραγματοποιεί τις εργασίες διαχείρισης για τις οποίες είναι εξουσιοδοτημένος.

- *Hardwares:*

  Οι χρήστες θα πρέπει να μπορούν να συνδεθούν στο σύστημα από τον προσωπικό τους υπολογιστή.

- *Το Software περιβάλλον εκτέλεσης της εφαρμογής:*

  Το λειτουργικό σύστημα των χρηστών θεωρείται πως είναι μια αρκετά πρόσφατη έκδοση είτε των Microsoft Windows (τουλάχιστον Windows 2000) είτε Mac OS Intel. Η εφαρμογή θα πρέπει να μπορεί να εκτελεστεί σε όλες τις πρόσφατες εκδόσεις των διαδεδομένων browser σε αυτά τα συστήματα. Πιο συγκεκριμένα: Internet Explorer 7/8 σε Windows (ειδικά για τα Windows 2000 μόνο ο Internet Explorer 6, μιας και δεν τρέχουν ομαλά οι επόμενες εκδόσεις του IE σε αυτό το λειτουργικό), Safari, Firefox 3, Chrome ανεξαρτήτου λειτουργικού.

- *Διασύνδεση Δικτύου:*

  Η επικοινωνία πελάτη (browser) – εξυπηρετητή θα πρέπει να μπορεί να γίνεται από οποιεσδήποτε θέσεις στο διαδίκτυο. Η βάση δεδομένων και ο εξυπηρετητής βρίσκονται σε τοπικό ασφαλές δίκτυο.

Χαρακτηριστικά εφαρμογής

- *Διεπαφή χρήστη:*

  Το User Interface θα πρέπει να είναι άμεσο, να βοηθά την γρήγορη διαχείριση των έργων, και να αξιοποιεί όσο περισσότερο χώρο στο παράθυρο του browser είναι εφικτό.

- *Σχεδίαση:*

  Η σχεδίαση του λογισμικού θα πρέπει να είναι αρκετά σαφής ώστε να είναι εύκολη η επέκτασή του από προγραμματιστή που δεν συμμετέχει στην αρχική ανάπτυξη της εφαρμογής. Ειδικότερα, η προσθήκη νέων σελίδων, η τροποποίηση των φορμών διαχείρισης και η προσθήκη νέων υπηρεσιών στον εξυπηρετητή, θα πρέπει να γίνεται γρήγορα και με συστηματικό τρόπο.

Βασικές Ιδιότητες του Συστήματος

- *Αξιοπιστία:*

  Η βάση δεδομένων θα πρέπει κάτω από οποιεσδήποτε συνθήκες να βρίσκεται σε έγκυρη κατάσταση. Η εφαρμογή πελάτη θα πρέπει να ανταποκρίνεται στον χρήστη ακόμη και όταν το έργο περιλαμβάνει μεγάλο όγκο πληροφορίας.

- *Διαθεσιμότητα:*

  Η βάση δεδομένων και οι υπηρεσίες θα πρέπει να είναι συνεχώς διαθέσιμες και προσπελάσιμες από πολλούς χρήστες ταυτόχρονα.

- *Ασφάλεια:*

  Το σύστημα θα πρέπει να διαθέτει ένα αρκετά ισχυρό σχήμα ασφάλειας, το οποίο να είναι σε θέση να:

  1) απαιτεί από τους χρήστες να δίνουν username και password για την είσοδο τους,
  2) να εγγυάται πως δεν θα είναι δυνατή η υποκλοπή των στοιχείων που μεταφέρονται μεταξύ πελάτη και εξυπηρετητή, συμπεριλαμβανομένου των username και password
  3) να περιορίζει τους χρήστες στις εργασίες διαχείρισης που είναι εξουσιοδοτημένοι να κάνουν
  4) να απαγορεύει οποιαδήποτε πρόσβαση στην βάση δεδομένων πέρα από την έμμεση που μπορεί να γίνει μέσω των υπηρεσιών.

- *Συντηρησιμότητα:*

  Η εφαρμογή θα πρέπει να επιτρέπει την χρήση εργαλείων εποπτείας της λειτουργίας της, ώστε να εντοπίζεται γρήγορα η αιτία τυχόν σφαλμάτων, και να τεκμηριώνει τους τρόπους με τους οποίους μπορούν να επεκταθούν οι λειτουργίες της.

- *Απόδοση:*

  Η αλληλεπίδραση του χρήστη με τον πελάτη θα πρέπει να είναι συνεχής και να μην διακόπτεται ακόμη και όταν ο δεύτερος επικοινωνεί με τον εξυπηρετητή. Η βάση δεδομένων και οι υπηρεσίες θα πρέπει να είναι σε θέση να εξυπηρετούν με ταχύτητα αιτήματα από πολλούς πελάτες ταυτόχρονα.

- *Βάση Δεδομένων:*

  Θα πρέπει να είναι δυνατή η χρήση της βάσης δεδομένων του προϋπάρχοντος συστήματος, χωρίς την παραμικρή μετατροπή, από την νέα εφαρμογή.

### 1.6.2 Προϋπάρχουσα Εφαρμογή

Το προϋπάρχον σύστημα χρησιμοποιεί έναν Microsoft Sql Server για την βάση δεδομένων. Σε αυτήν αποθηκεύονται όλα τα στοιχεία που που αφορούν τα έργα, την διαχείρισή τους, καθώς και πληροφορίες για τους χρήστες του συστήματος και τους ρόλους τους σε αυτό.

Οι χρήστες αλληλεπιδρούν με το σύστημα από τον web browser του προσωπικού τους υπολογιστή, μέσω ASP.NET σελίδων. Ο κώδικας στις Web Forms, δηλαδή στον Visual Basic κώδικα που βρίσκεται στις ASP.NET σελίδες, αναλαμβάνει τόσο την σχεδίαση των σελίδων όσο και την επικοινωνία με την βάση δεδομένων.

## 1.7 Διάρθρωση του Κειμένου

Τα επόμενα κεφάλαια οργανώνονται ως εξής:

- *Κεφάλαιο 2$^{ο}$ : Επιθυμητή Μέθοδος Διαχείρισης Έργου*

  Γίνεται μια αναλυτική παρουσίαση της μεθοδολογίας διαχείρισης έργου που εφαρμόζει η προϋπάρχουσα εφαρμογή που και υλοποιεί και η δική μας.

- *Κεφάλαιο 3$^{ο}$ : Εναλλακτικές Τεχνολογίες*

  Οι λειτουργικές απαιτήσεις μας οδήγησαν στην απόφαση της χρήσης RIA τεχνολογιών για την σχεδίαση του πελάτη. Προτού παρουσιάσουμε τα εργαλεία που χρησιμοποιήσαμε, κάνουμε στο κεφάλαιο αυτό μια σύντομη αναφορά σε εναλλακτικές επιλογές και περιγράφουμε γενικά τα χαρακτηριστικά των RIA εφαρμογών.

- *Κεφάλαιο 4$^{ο}$ : Χρησιμοποιούμενη Τεχνολογία*

Οι σπουδαιότερες τεχνολογίες και τα εργαλεία που αξιοποιήσαμε στην υλοποίηση της εφαρμογής και παρουσίαση των λόγων που καταλήξαμε στις συγκεκριμένες επιλογές.

- *Κεφάλαιο 5ο : Υλοποίηση*

  Αναλυτική περιγραφή της βάσης δεδομένων και περιγραφή της αρχιτεκτονικής του εξυπηρετητή και του πελάτη. Γίνεται επίσης επεξήγηση μερικών βασικών σημείων του κώδικα της εφαρμογής.

- *Κεφάλαιο 6ο : Παρουσίαση*

  Οι βασικές οθόνες της διεπαφής της εφαρμογής, όπως τις βλέπει ο τελικός χρήστης από τον browser του.

- *Κεφάλαιο 7ο : Επίλογος*

  Συμπεράσματα, προτάσεις για πιθανές επεκτάσεις του συστήματος

- *Βιβλιογραφία*

- *Επιλεγμένα Τμήματα Του Κώδικα Της Εφαρμογής*

# 2. Επιθυμητή Μέθοδος Διαχείρισης Έργου

Παρακάτω περιγράφεται η μεθοδολογία που ακολουθείται για την διαχείριση των έργων. Η μεθοδολογία αυτή θα υλοποιηθεί με τα σύγχρονα εργαλεία που αναφέρονται στην ενότητα 4.

## 2.1 Έντυπο Έργου (Ε1)

Σκοπός αυτού του εντύπου είναι ο πλήρης προσδιορισμός των στοιχείων που χαρακτηρίζουν ένα έργο. Το έντυπο συμπληρώνεται από τον υπεύθυνο σύμβασης με την έναρξη του έργου και ενημερώνεται αν μεταβληθεί κάποιο στοιχείο κατά την διάρκεια αυτού.

Ο κωδικός του έργου (πεδίο 1), το ακρωνύμιο του (πεδίο 2), η επιστημονική περιοχή στην οποία ανήκει (πεδίο 3), ο αριθμός συμβολαίου (πεδίο 4) και ο τίτλος του (πεδίο 5) είναι τα στοιχεία που χρειάζονται τα μέλη μιας εταιρεία για να προσδιορίζουν το έργο στο οποίο αναφέρεται το παρόν έντυπο. Στην συνέχεια, και για λόγους πληρότητας του εντύπου, στα πεδία 6 και 7 δηλώνονται οι υπεύθυνοι του συγκεκριμένου έργου.

Επειδή τα μεγάλα έργα απαιτούν κοινοπραξίες εταιριών για να μπορέσουν να υλοποιηθούν, το έντυπο προβλέπει αυτή την περίπτωση και ενώ το πεδίο 8 και πεδίο 9 φαίνονται να είναι πλεονασμός, την πράξη οι ανθρωπομήνες της ομάδας (δηλαδή των μελών της εταιρίας) είναι σχεδόν πάντα γνήσιο υποσύνολο του συνόλου των ανθρωπομηνών.

Ο εντολέας (πεδίο 10) είναι το άτομο ή ο οργανισμός με το οποίο συνεννοήθηκε ο υπεύθυνος σύμβασης για την ανάληψη του έργου. Στην συνέχεια του εντύπου εμφανίζονται οι συμμετέχοντες στο έργο και ο ρόλος τους σε αυτή την κοινοπραξία (πεδίο 11), οι αποδέκτες του έργου (πεδίο 12), μια σύντομη περιγραφή του έργου (πεδίο 13), οι λέξεις κλειδιά (πεδίο 14) για την διευκόλυνση αναζήτησης παρεμφερή έργων, οι κρίσιμες ημερομηνίες

(πεδία 15 έως 17), το πλήθος των εργασιών που πρέπει να γίνουν για την ολοκλήρωση του έργου (πεδία 18 έως 20) και στο τέλος μπαίνει το όνομα αυτού που συμπλήρωσε το έντυπο (δηλαδή του υπεύθυνου σύμβασης) και η ημερομηνία συμπλήρωσης αυτού.

## E1. Κάρτα έργου

| | |
|---|---|
| 1. Κωδικός | 2. Ακρωνύμιο |
| 3. Επιστημονική Περιοχή | 4. Αριθμός Συμβολαίου |
| 5. Τίτλος | |
| 6. Υπεύθυνος Σύμβασης | 7. Υπεύθυνος Έργου |
| 8. Σύνολο Ανθρωπομηνών | 9 Ανθρωπομήνες Ομάδας |
| 10. Εντολέας | |

| 11. Συμμετέχοντες | Ρόλος | | |
|---|---|---|---|
| | Αρχηγός Κοινοπραξίας | Συνεργάτης | Υπεργολάβος |
| | | | |
| | | | |
| | | | |

| |
|---|
| 12. Αποδέκτες (Χώρες / Οργανισμοί) |

| 13. Σύντομη Περιγραφή (Σκοπός / Αντικείμενο) |
|---|
| |

| 14. Λέξεις Κλειδιά |
|---|
| |

| | Προγραμματισμένη | Πραγματοποιηθείσα |
|---|---|---|
| 15. Ημερομηνία Έναρξης | | |
| 16. Ημερομηνία Λήξης | | |
| αναθεώρηση | | |
| 17. Διάρκεια (μήνες) | | |
| αναθεώρηση | | |

| 18. Πλήθος Πακέτων Εργασίας | | 19. Πλήθος Εργασιών | |
|---|---|---|---|

| 20. Πλήθος Παραδοτέων | |
|---|---|

| 21. Συντάκτης: | 22 Τελευταία Ενημέρωση |
|---|---|
| | |

## 2.2 Έντυπο Πόρων Έργου (Ε2)

Το έντυπο Ε2 αποτελεί μια σύνοψη των πόρων που θα χρειαστεί το έργο για να ολοκληρωθεί. Συμπληρώνεται και πάλι από τον υπεύθυνο σύμβασης και ενημερώνεται μόνο εάν μεταβληθεί ουσιαστικά. Τα πρώτα τέσσερα πεδία του εντύπου είναι τα ίδια με το προηγούμενο έντυπο έτσι ώστε να εξασφαλίζεται η αυτονομία των εντύπων.

Όσον αφορά τον προϋπολογισμό που καλείται να συμπληρώσει ο υπεύθυνος σύμβασης, χωρίζεται σε δύο κύριες κατηγορίες. Στην χρηματοδότηση του έργου και την χρηματοδότηση της ομάδας. Το έργο απαιτεί κάποιους πόρους για να μπορέσει να υλοποιηθεί και τα μέλη της εταιρίας που θα δουλέψουν στο συγκεκριμένο έργο πρέπει να ανταμειφθούν για τις εργασίες τους. Τα χρήματα μπορεί να προέρχονται είτε από τον εντολέα ή από άλλες πηγές όπως είναι οι επιχορηγήσεις, τα ιδία χρήματα ή κάποιο δάνειο.

Το ανθρώπινο δυναμικό αποτελεί την πιο χρήσιμη πληροφορία για το σύστημα στην υπάρχουσα έκδοση του. Στο πεδίο 6 πρέπει να συμπληρωθούν τα άτομα που θα εργαστούν στο συγκεκριμένο έργο, η θέση τους στο έργο και ο αριθμός των ανθρωποημερών που θα ασχοληθούν με αυτό. Μια εταιρεία μπορεί να αναθέσει εργασία σε τρεις κατηγορίες εργαζομένων. Σε αυτούς που είναι μέλη της ομάδας εργασίας, στους εξωτερικούς συνεργάτες τις ομάδας και στο προσωπικό διοικητικής και γραμματειακής υποστήριξης.

Στην συνέχεια, το έντυπο δίνει την δυνατότητα στον υπεύθυνο σύμβασης να συμπληρώσει τον χώρο στον οποίο θα γίνουν οι εργασίες, τον εξοπλισμός και το λογισμικό που θα χρησιμοποιήσει η ομάδα εργασίας και την λοιπή υποδομή που χρειάζεται για την υλοποίηση του έργου.

Η ημερομηνία συμπλήρωσης που πρέπει να συμπληρωθεί σε όλα τα έντυπα υπάρχει για λόγους ταξινόμησης των εγγράφων.

## E2. Κάρτα πόρων έργου

| | | | |
|---|---|---|---|
| 1. | Κωδικός | 2. | Ακρωνύμιο |
| 3. | Επιστημονική Περιοχή | 4. | Αριθμός Συμβολαίου |

| | | | |
|---|---|---|---|
| 4. | Τίτλος Έργου | | |
| 5. | Προϋπολογισμός | | |
| | | Έργου | Ομάδας |
| | Χρηματοδότηση Εντολέα | | |
| | Χρηματοδότηση από άλλες Πηγές (ιδία χρημ., κτλ) | | |
| | Συνολικός Προϋπολογισμός | | |

| | | | | |
|---|---|---|---|---|
| 6. | Ανθρώπινο Δυναμικό | | | |
| A. | Μέλη Ομάδας | | | |
| | ΑΑ | Θέση στο Έργο | Όνομα | Απασχόληση (αν.ημ.) |
| | 1. | | | |
| | 2. | | | |
| | **Σύνολο** | | | |
| B. | Εξωτερικοί Συνεργάτες Ομάδας | | | |
| | A | Θέση στο Έργο | Όνομα | Απασχόληση |

| | | | | |
|---|---|---|---|---|
| | Α | | | (αν.ημ.) |
| | 3. | | | |
| | 4. | | | |
| | **Σύνολο** | | | |
| Γ. | Προσωπικό Διοικητικής/ Γραμματειακής Υποστήριξης | | | |
| | Α Α | Θέση στο Έργο | Όνομα | Απασχόληση (αν.ημ.) |
| | 5. | | | |
| | **Σύνολο** | | | |

| |
|---|
| 7. Χώρος |

| |
|---|
| 8. Εξοπλισμός / Λογισμικό |

| |
|---|
| 9. Λοιπή Υποδομή |

| |
|---|
| 10. Ημερομηνία συμπλήρωσης |

## 2.3 Κάρτες Οικονομικών και Τεχνικών Παραδοτέων (Ε3)

Με την σύναψη ενός συμβολαίου για την εκπόνηση ενός έργου, καθορίζονται τόσο τα τεχνικά όσο και τα οικονομικά παραδοτέα. Συνήθως, τα

τεχνικά παραδοτέα είναι αυτά που έχει αναλάβει η εταιρία να παραδώσει, ενώ τα οικονομικά αποτελούν τα χρήματα που έχει δεσμευτεί ο εντολέας να πληρώσει στους συμμετέχοντες. Το έντυπο συμπληρώνεται από τον υπεύθυνο σύμβασης.

## Ε3. Κάρτα Συμβατικών Παραδοτέων

| | |
|---|---|
| 1. Κωδικός | 2. Ακρωνύμιο |
| 3. Επιστημονική Περιοχή | 4. Αριθμός Συμβολαίου |

5 Όνομα

6 ΤΕΧΝΙΚΑ ΠΑΡΑΔΟΤΕΑ

| ΑΑ | Τίτλος | Ημερομηνία Παράδοσης | |
|---|---|---|---|
| | | Προγραμματισμένη | Πραγματοποιηθείσα |
| 1. | | | |
| 2. | | | |
| 3. | | | |
| 4. | | | |
| 5. | | | |
| 6. | | | |
| 7. | | | |

7. ΟΙΚΟΝΟΜΙΚΑ ΠΑΡΑΔΟΤΕΑ

| ΑΑ | Τίτλος | Ημερομηνία Παράδοσης | |
|---|---|---|---|
| | | Προγραμματισμένη | Πραγματοποιηθείσα |
| 1. | | | |
| 2. | | | |
| 3. | | | |
| 4. | | | |
| 5. | | | |
| 6. | | | |
| 7. | | | |

8. Ημερομηνία συμπλήρωσης

## 2.4 Κάρτες Ορισμού Εργασιών (Ε4)

Ενώ όλα τα παραπάνω έντυπα είχαν σκοπό τον προσδιορισμού του έργου και των απαιτήσεων του, αυτό το έντυπο περιέχει χρήσιμες πληροφορίες όσον αφορά τον προγραμματισμό που θα γίνει. Το παρόν έντυπο συμπληρώνεται από τον υπεύθυνο έργου και ανανεώνεται όταν υπάρχουν σημαντικές αλλαγές.

Όπως φαίνεται και παρακάτω, το έντυπο αυτό απαρτίζεται από τρεις πίνακες. Οι πρώτοι δύο περιέχουν την ίδια πληροφορία προσφέροντας την με διαφορετική οπτική γωνία. Ο πίνακας 4.1 περιέχει την ανάλυση των πόρων ανά αντικείμενο εργασίας, ενώ ο 4.2 αποτελεί την ανάλυση των αντικειμένων εργασίας ανά μέλος ομάδας εργασίας.

Με τον όρο αντικείμενο εργασίας ονοματίζουμε κάθε εργασία που πρέπει να γίνει για να ολοκληρωθεί ένα έργο. Αυτά τα αντικείμενα μπορεί να είναι υποσύνολο των παραδοτέων που ορίζονται στο έντυπο Ε3 ή να είναι τελείως ασυσχέτιστα.

Ο πίνακας 4.3 αποτελεί τον μηνιαίο χρονοπρογραμματισμό των πόρων που έχουν δεσμευτεί για το συγκεκριμένο έργο. Η πληροφορία που μας δίνει είναι αρκετά σημαντική στην πρόβλεψη πιθανής ανεπάρκειας πόρων που μπορεί να προκύψει σε μια εταιρία. Δεν πρέπει να ξεχνάμε ότι μια εταιρία σπανίως αναλαμβάνει μόνο ένα έργο. Συνεπώς όταν οι πόροι της εταιρίας ασχολούνται με πολλά έργα ταυτόχρονα, υπάρχει κίνδυνος να χρειαστεί το ίδιο άτομα σε πολλά έργα την ίδια χρονική περίοδο. Αυτά τα προβλήματα άλλωστε προσπαθεί να επιλύσει η εφαρμογή που αναπτύσσεται στην παρούσα διπλωματική.

Ε4. Κάρτα Χρονοπρογραμματισμού Πόρων

(συμπληρώνεται από τον Υπεύθυνο Έργου με την έναρξη του έργου. Ενημερώνεται αν υπάρξουν ουσιαστικές μεταβολές)

| 1. | Κωδικός | 2. | Ακρωνύμιο |
|---|---|---|---|
| | | | |

| 3. | Τίτλος Έργου |
|---|---|
| | |

| 4 | Ημερομηνία συμπλήρωσης |
|---|---|
| | |

**Πίνακας 4.1 Ανάλυση Πόρων ανά Αντικείμενο**

| Α/Α | Αντικείμενο | ΜΟΕ που συμμετέχουν | | |
|---|---|---|---|---|
| | | Α/Α | Ονοματεπώνυμο | Ανθρωποημέρες |
| 1 | | | | |
| | | | | |
| | | | | |
| | | Σύνολο | | |
| 2 | | | | |
| | | | | |
| | | | | |
| | | Σύνολο | | |
| 3 | | | | |
| | | | | |
| | | | | |
| | | Σύνολο | | |
| | | **ΣΥΝΟΛΟ** | | |

**Πίνακας 4.2 Ανάλυση Αντικειμένων ανά ΜΟΕ**

| Α/Α | Ονοματεπώνυμο ΜΟΕ | Αντικείμενο | | |
|---|---|---|---|---|
| | | Α/Α | Τίτλος | Ανθρωποημέρες |
| 1 | | | | |
| | | | | |
| | | | | |
| | | Σύνολο | | |
| 2 | | | | |
| | | | | |
| | | | | |
| | | Σύνολο | | |
| 3 | | | | |
| | | | | |
| | | | | |
| | | Σύνολο | | |
| | | **ΣΥΝΟΛΟ** | | |

**Πίνακας 4.3:** Μηνιαίος Χρονοπρογραμματισμός Πόρων

| Α/Α | Ονοματεπώνυμο ΜΟΕ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | |

## 2.5 Κάρτα Χρονοπρογραμματισμού και Ανάθεσης Εργασιών (Ε5)

Αυτό το έντυπο αποτελεί την αποτύπωση του εβδομαδιαίο προγραμματισμό του υπεύθυνου έργου. Αποτελεί εργαλείο προγραμματισμού (planning) και επίβλεψης εφόσον ο υπεύθυνος έργου συμπληρώνει τον απολογισμό της ομάδας εργασίας για την προηγούμενη εβδομάδα και στην συνέχεια τον προγραμματισμό για την επόμενη.

Ο υπεύθυνος έργου πρέπει να συμπληρώσει τα στοιχεία που χαρακτηρίζουν το έργο, δηλαδή τον κωδικό του, το ακρωνύμιο του και τον τίτλο του, καθώς επίσης και τον αύξοντα αριθμό της τρέχουσας εβδομάδας για το συγκεκριμένο έργο. Με αυτά τα στοιχεία προσδιορίζεται πλήρως η χρονική στιγμή στην οποία αναφέρεται το παρών έντυπο.

Έχοντας συμπληρώσει το έντυπο Ε4, ο υπεύθυνος έργου έχει ορίσει τον συνολικό αριθμό ανθρωποημερών που χρειάζεται κάθε αντικείμενο εργασίας για να ολοκληρωθεί. Συνεπώς, αν ένα αντικείμενο έχει δηλωθεί ότι χρειάζεται 50 ανθρωποημέρες και ένα μέλος ομάδας εργασίας ασχολήθηκε μαζί του την προηγούμενη εβδομάδα 5 ανθρωποημέρες, τότε εκπλήρωσε το $\frac{5}{50} * 100\% = 10\%$ του συνολικού αντικειμένου εργασίας. Αυτό το νούμερο πρέπει να συμπληρωθεί στο πεδίο '% ολοκλ. Εβδομάδας' που υπάρχει στο έντυπο Ε5. Στην συνέχεια υπάρχει το πεδίο '% ολοκλ. Σωρευτικό' το οποίο είναι το συνολικό ποσοστό ολοκλήρωσης του αντικειμένου εργασίας που έχει γίνει από όλα τα μέλη εργασίας μέχρι στιγμής.

| 1. | Κωδικός |
|---|---|
| | |

| 2. | Ακρωνύμιο |
|---|---|
| | |

| 3. | Τίτλος Έργου |
|---|---|
| | |

| 4 | Ημ/νία συμπλήρωσης |
|---|---|
| | |

| 5 | Α/Α τρέχουσας εβδομάδας |
|---|---|
| | |

| Α/Α | Όνομα, Επώνυμο ΜΟΕ | **Ανθρωποημέρες Εβδομάδας** | | **Αντικείμενα Τρέχουσας Εβδομάδας** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Τρέχουσας (απολογισμός)** | **Επόμενης (πρόβλεψη)** | **Τρέχουσας εβδομάδας (απολογιστικά)** | | | | **Επόμενης εβδομάδας (πρόβλεψη)** | | | |
| | | | | Α/Α | Τίτλος | %. Εβδομάδας | %. Σωρευτικό | Α/Α | Τίτλος | %. Εβδομάδας | %. Σωρευτικό |
| **1** | | | | | | | | | | | |
| | | | | | | | | | | | |
| **2** | | | | | | | | | | | |
| | | | | | | | | | | | |
| **3** | | | | | | | | | | | |
| | | | | | | | | | | | |
| **4** | | | | | | | | | | | |
| | | | | | | | | | | | |
| **5** | | | | | | | | | | | |
| | | | | | | | | | | | |

## 2.6 Κατηγορίες Μελών Έργου Και Τύποι Χρηστών

Οι τρεις επόμενες υποενότητες αντιστοιχούν η καθεμιά σε μια κατηγορία μελών ενός έργου. Για κάθε τύπο μέλους, κάνουμε πρώτα μια γενική περιγραφή του ρόλου του σε ένα έργο και μετά απαριθμούμε τα συγκεκριμένα δικαιώματά του στην μεθοδολογία που υλοποιεί η εφαρμογή.
Η ενότητα αυτή τελειώνει με μία επιπλέον τέταρτη υποενότητα, η οποία αναφέρεται στους ρόλους που μπορεί να έχουν οι χρήστες συνολικά στο σύστημα, ανεξάρτητα από τους ρόλους που πιθανώς έχουν στα επιμέρους έργα.

### 2.6.1 Διευθυντής Έργου

Γενικά, ο διευθυντής του έργου έχει την ευθύνη διαχείρισης της σύμβασης έργου για την οποία είναι υπεύθυνος και διασφάλισης της ποιότητας των παραδοτέων. Η ευθύνη αυτή αφορά σε όλο τον «κύκλο ζωής» της σύμβασης και εξειδικεύεται ως εξής:

- *Σχεδίαση συμβατικών χαρακτηριστικών του έργου*

  Συνεννόηση με όλους τους συνεργαζόμενους φορείς (εντολείς, υπεργολάβοι, εργολάβοι, παραλήπτες), συμφωνία επί των συμβατικών χαρακτηριστικών του έργου (ρόλοι, τεχνικό και οικονομικό αντικείμενο ανά συνεργαζόμενο φορέα κλπ).

- *Σύνταξη Σύμβασης*

- *Υπογραφή Σύμβασης*

- *Κατάρτιση / αναθεώρηση προϋπολογισμού έργου*

- *Δέσμευση πόρων έργου*

  Η εξασφάλιση των απαραίτητων πόρων (ανθρώπινο δυναμικό, χώροι, hardware, software, κλπ υποδομή) βάσει του συμφωνηθέντος προγραμματισμού

- *Κατάρτιση συμβατικών παραδοτέων έργου*

  Κατάρτιση προδιαγραφών ποιότητας των συμβατικών τεχνικών αναφορών του έργου και εμπρόθεσμη υποβολή των αναφορών αυτών

- *Προώθηση αναπτυξιακών δυνατοτήτων έργου*

  Δημιουργία κλίματος καλής συνεργασίας με τα στελέχη του Εντολέα, διερεύνηση και προώθηση δυνατοτήτων περαιτέρω συνεργασίας

Στην εφαρμογή, ο Διευθυντής έργου αναφέρεται ως Project Director, και επί της ουσίας έχει τον πλήρη έλεγχο σε ένα έργο. Μπορεί δηλαδή να προσπελάσει και να τροποποιήσει όλες τις κάρτες διαχείρισης έργου.

### 2.6.2 Υπεύθυνος Έργου

Αναφέρεται και ως Project Manager. Είναι υπεύθυνος για:

- την επίτευξη του τελικού στόχου του έργου με τους περιορισμούς των διαθέσιμων πόρων και μέσα στα προκαθορισμένα πλαίσια χρόνου, κόστους και απόδοσης,
- την λήψη των απαιτούμενων αποφάσεων για κάθε φάση και δραστηριότητα του έργου,
- την ομαλή επικοινωνία των υπευθύνων των επιμέρους λειτουργικών μονάδων.

Οι υπευθυνότητες του project manager μπορούν να ομαδοποιηθούν ως εξής:

- *διαχείριση σχέσεων (interface management),* με την έννοια ότι ο υπεύθυνος έργου θα πρέπει να διαχειρίζεται τις σχέσεις και αλληλεπιδράσεις (interface) τόσο στο εξωτερικό περιβάλλον της εταιρίας (με τους πελάτες), στο εξωτερικό περιβάλλον της εταιρίας αλλά στο εξωτερικό περιβάλλον του έργου (με τους υπεύθυνους των λειτουργικών μονάδων), αλλά και στο εσωτερικό περιβάλλον του έργου (με τα μέλη της ομάδας έργου).
- *Διαχείριση σχεδιασμού και ελέγχου (planning and control management),* δηλαδή ο σχεδιασμός, έλεγχος και αξιολόγηση της χρήσης του εξοπλισμού, η ελαχιστοποίηση των κινδύνων αποτυχίας, ο προσδιορισμός εναλλακτικών λύσεων σε προβλήματα κλπ
- *Διαχείριση πόρων (resource management),* αναφορικά με τον χρόνο, το ανθρώπινο δυναμικό, τον εξοπλισμό, την τεχνολογία πληροφορικής που χρησιμοποιεί το έργο κλπ.

Στο σύστημά μας, ο Project Manager μπορεί να προσπελάσει όλες τις κάρτες διαχείρισης έργου, αλλά μπορεί να τροποποιήσει μόνον εκείνες που αναφέρονται στον έλεγχο της ομάδας εργασίας. Με άλλα λόγια, μπορεί να τροποποιήσει τις E5, E4, E43, και την λίστα των ανθρώπινων πόρων της E2.

### 2.6.3 Ομάδα Εργασίας

Τα μέλη της ομάδας έργου αποτελούν τους κύριους πόρους του έργου. Σκοπός τους είναι η έγκαιρη πραγματοποίηση των αντικειμένων εργασίας τα οποία τους έχουν ανατεθεί από τους υπεύθυνους έργου και σύμβασης.

Όσοι έχουν αυτόν τον ρόλο στο έργο μπορούν μόνο να προσπελάσουν τις πληροφορίες που αναφέρονται ειδικά σε αυτούς. Δηλαδή, τα στοιχεία από τις λίστες των E3, E43 και E5 που περιγράφουν τις δικές τους εργασίες και υποχρεώσεις.

### 2.6.4 Τύποι Χρηστών Στο Σύστημα

Ένας χρήστης που συνδέεται στο σύστημα μέσω ενός πελάτη, ανήκει σε ακριβώς μία από τις παρακάτω κατηγορίες:

- *Επισκέπτης*
  Ο επισκέπτης είναι ο βασικός ρόλος που ανατίθεται σε όσους έχουν απλά λογαριασμό στο σύστημα. Αυτοί οι χρήστες έχουν δικαίωμα να δούνε μόνο την κεντρική σελίδα και την σελίδα όπου μπορούν να αλλάξουν τα στοιχεία τους.

- *Χρήστης*
  Ο χρήστης είναι ο ρόλος που πρέπει να έχουν όλα τα μέλη της ομάδας εργασίας. Έχοντας αυτόν τον ρόλο, ο χρήστης μπορεί να δει τα έργα στα οποία έχει κάποιον ρόλο.

- *Δημιουργός Έργου*
  Επιπρόσθετα των δικαιωμάτων της προηγούμενης κατηγορίας, χρήστες αυτής της κατηγορίας μπορούν να δημιουργούν νέα έργα.

- *Διαχειριστής*
  Αυτός είναι ο πιο πλέον σημαντικός ρόλος του συστήματος. Έχει τα δικαιώματα όλων των προηγούμενων κατηγοριών, και επιπλέον είναι υπεύθυνος για τους υπόλοιπους χρήστες του συστήματος: μπορεί να τροποποιήσει τα στοιχεία τους, να αλλάξει τους κωδικούς τους και τους ρόλους τους στο σύστημα, να αφαιρέσει ή να προσθέσει νέους. Επίσης, έχει δικαιώματα Διευθυντή Έργου σε όλα τα έργα. Όπως όλοι οι διαχειριστές σε όλα τα συστήματα, έτσι και εδώ αυτόν τον ρόλο πρέπει να τον έχουν έμπιστα άτομα τα οποία γνωρίζουν τις δυνατότητες του συστήματος.

# 3. Εναλλακτικές Τεχνολογίες για την Υλοποίηση της Εφαρμογής

Στα πρώτα χρόνια του Web, σχεδόν όλοι οι ιστότοποι αλληλεπιδρούσαν με τον χρήστη χρησιμοποιώντας μόνο HTML κώδικα. Αυτή η τεχνική είχε φανερά μειονεκτήματα: περιορισμένες επιλογές στην σχεδίαση του user interface, ανάγκη για συχνές ανανεώσεις ολόκληρου του περιεχομένου της σελίδας, ακόμη και για μικρές αλλαγές, αυξημένες υπολογιστικές απαιτήσεις προς τον εξυπηρετητή.

Προκειμένου να ξεπεραστούν αυτά τα προβλήματα, είναι απαραίτητο να αποκτήσει ο web πελάτης πιο δυναμικό ρόλο: θα πρέπει να είναι σε θέση να εκτελεί μόνος του τις λειτουργίες που δεν απαιτούν λογικά την μεσολάβηση του εξυπηρετητή, ενώ η επικοινωνία τους θα πρέπει να περιορίζεται σε ακριβώς εκείνα τα δεδομένα που χρειάζεται ο πελάτης, χωρίς να είναι απαραίτητη κάθε φορά η μεταφορά ολόκληρου του περιεχομένου της σελίδας. Μάλιστα, ακόμη και κατά την μεταφορά των δεδομένων, η εφαρμογή-πελάτη θα ήταν επιθυμητό να μπορεί να εκτελεί και άλλες λειτουργίες, αντί να διακόπτεται έως ότου ολοκληρωθεί η μεταφορά.

Τέτοιες σύνθετες εργασίες είναι δυνατό να εκτελεστούν μόνο αν ο πελάτης μπορεί να εκτελεί πραγματικό κώδικα που του στέλνει ο εξυπηρετητής και όχι μόνο απλή HTML. Ιστορικά, αυτό έγινε πρώτα εφικτό με την εμφάνιση των Java Applets, στην πρώτη έκδοση της Java το 1995. Ακολούθησαν σύντομα τα ActiveX Controls της Microsoft (1996), τα οποία όμως υποστηρίζονται επίσημα μόνο από τον Internet Explorer, και εκτελούνται σε πραγματικό κώδικα μηχανής, εισάγοντας πολλά κενά ασφαλείας στο μοντέλο προγραμματισμού.

Σχετικές τεχνολογίες που εμφανίστηκαν τα επόμενα χρόνια άρχισαν να προσφέρουν όλο και περισσότερες δυνατότητες στον προγραμματιστή, σε βαθμό που από μια στιγμή και μετά οι διαδικτυακές εφαρμογές να αρχίσουν να προσομοιάζουν στο user interface των desktop προγραμμάτων. Μια τέτοια εφαρμογή, ανεξάρτητα της συγκεκριμένης τεχνολογίας που χρησιμοποιεί, έχει καθιερωθεί να καλείται RIA, Rich Internet Application.

Η ικανοποίηση των απαιτήσεων που καταγράψαμε στην σχετική ενότητα της Εισαγωγής μπορεί να γίνει μόνο με την χρήση μιας τέτοιας

τεχνολογίας. Την συγκεκριμένη επιλογή που κάναμε, και τους λόγους που μας οδήγησαν σε αυτήν, την παρουσιάζουμε στο επόμενο κεφάλαιο. Σε αυτό το κεφάλαιο θα αναφερθούμε στις βασικότερες εναλλακτικές επιλογές.

## 3.1 Adobe Flash / AIR

Η προγραμματιστική πλατφόρμα Flash της Adobe, είναι η πλέον διαδεδομένη τεχνολογία που χρησιμοποιούν οι Web Developers για να εισάγουν βίντεο, animation και διαδραστικότητα σε ιστοχώρους.

Πρόσφατα (από τον Φεβρουάριο 2009) η Adobe άρχισε να προσθέτει στο Flash και δυνατότητες για την ανάπτυξη RIA εφαρμογών με χαρακτηριστικά παρόμοια των desktop εφαρμογών, καθιστώντας το μια βιώσιμη λύση για προγραμματιστικές ανάγκες όπως η δική μας.

Οι επιπλέον δυνατότητες, στις τελευταίες εκδόσεις του Adobe Flash προσφέρονται από το Adobe Integrated Runtime. Πρόκειται για ένα περιβάλλον εκτέλεσης που επιτρέπει σε κώδικα Flash (και όχι μόνο, επίσης HTML και Javascript) να χρησιμοποιηθεί για την κατασκευή είτε desktop εφαρμογών είτε εφαρμογών που τρέχουν σε browser, αλλά με χαρακτηριστικά παραδοσιακών desktop προγραμμάτων. Το RIA χρησιμοποιείται συνήθως σε συνδυασμό με το Adobe Flex, μια πλατφόρμα ανάπτυξης RIA εφαρμογών.

Ένα από τα προηγμένα χαρακτηριστικά του Adobe Flex, είναι η χρήση μιας εξειδικευμένης markup γλώσσας, καλούμενη MXML, για την σχεδίαση και υλοποίηση του user interface των εφαρμογών. Παρόμοιας λογικής γλώσσες διαθέτουν όλες οι κύριες ανταγωνιστικές τεχνολογίες (XAML για το Silverlight, JavaFX Script για την JavaFX).

Προκειμένου να εκτελεστεί μια εφαρμογή που αξιοποιεί το AIR, θα πρέπει στον υπολογιστή του χρήστη να είναι εγκατεστημένο το σχετικό περιβάλλον εκτέλεσης, μαζί βέβαια με το plug-in για το Flash. Και τα δύο είναι διαθέσιμα για Linux, Mac OS X και Windows. Χάρη στην επιθετική πολιτική της Adobe, είναι όντως εγκατεστημένο στην πλειονότητα των προσωπικών υπολογιστών.

## 3.2 Java

Από την πρώτη της κιόλας έκδοση, η Java έδινε την δυνατότητα στον web προγραμματιστή να εμπλουτίσει το περιεχόμενο των ιστοσελίδων μέσω των Java Applets. Όμως, τα Applets από μόνα τους δεν είναι κατάλληλα για την ανάπτυξη RIA εφαρμογών. Το κύριο πρόβλημα είναι πως η σχεδίαση του user interface με αυτά βασίζεται στις standard βιβλιοθήκες του Swing και AWT της Java, η οποίες είναι δύσχρηστες και πολύπλοκες για τον web designer.
Το μειονέκτημα αυτό προσπαθεί να αντιμετωπίσει μια νέα βιβλιοθήκη του Java Runtime Environment, η JavaFX, την οποία μάλιστα δίνει η Sun με ελεύθερου λογισμικού άδεια.
Όπως αναφέραμε παρενθετικά προηγουμένως, ο προγραμματισμός σε JavaFX γίνεται χρησιμοποιώντας την γλώσσα JavaFX Script, παρόμοιας λειτουργικότητας και λογικής με την MXML της Adobe και την XAML της Microsoft. Σε αντίθεση πάντως με τις τελευταίες δύο γλώσσες, η JavaFX Script δεν είναι XML γλώσσα, αλλά πρόκειται για compiled γλώσσα η οποία μεταφράζεται σε τυπικό java bytecode. Αυτό κάνει δυνατή την χρήση σε JavaFX Script κώδικα, πέρα των βιβλιοθηκών του JavaFX, και άλλων βιβλιοθηκών του Java Runtime Environment, όπως του Swing.
Στον σχεδιασμό της JavaFX η Sun έδωσε έμφαση στα εξής στοιχεία:

- Στην μεταφερσιμότητα του JavaFX κώδικα σε πολλές συσκευές (εκτός δηλαδή από προσωπικούς υπολογιστές, και σε αρκετά κινητά τηλέφωνα, για παράδειγμα) και σε αρκετά λειτουργικά συστήματα,
- Από πλευράς τελικού χρήστη, στην δυνατότητα της άμεσης drag-n-drop μεταφοράς της JavaFX εφαρμογής από τον browser στο desktop, ακόμη και την ώρα που αυτή εκτελείται,
- Στην υποστήριξη αρκετών πλατφορμών σχεδίασης γραφικών (SVG γραφικά, Adobe Photoshop, Adobe Illustrator).

## 3.3 Ajax-based

Ο όρος Ajax περιλαμβάνει ένα ευρύ σύνολο τεχνικών για την σχεδίαση διαδραστικών διαδικτυακών εφαρμογών που εκτελούνται στους browsers των τελικών χρηστών οι οποίες μπορούν να επικοινωνούν με κάποιον εξυπηρετητή χωρίς να παρεμβαίνουν στην τρέχουσα κατάσταση της σελίδας που βλέπει ο χρήστης . Το όνομά του είναι τα αρχικά των λέξεων Asynchronous JAvascript and Xml, αν και αυτές δεν συνοψίζουν ακριβώς τις Ajax τεχνικές.

Στην αρχική του περιγραφή, το Ajax αποτελούσε την σύζευξη πέντε τεχνολογιών:

- Την HTML ή XHTML με CSS για την παρουσίαση,
- Το Document Object Model (DOM) για την αλληλεπίδραση με τα δεδομένα που μεταφέρονται,
- XML για την μεταφορά των δεδομένων, και XSTL για τον άμεσο μετασχηματισμό τους σε HTML/XHTML,
- Την μέθοδο XMLHttpRequest για ασύγχρονη επικοινωνία. Πρόκειται για ένα API που διατίθεται στις βιβλιοθήκες των web browser scripting γλωσσών, όπως της Javascript, για την μεταφορά XML και άλλων δεδομένων από τον εξυπηρετητή στον πελάτη,
- Η γλώσσα Javascript.

Οι τεχνολογίες αυτές χρησιμοποιούνται ως εξής: Στην έναρξη μιας συνεδρίας του browser με έναν web server, αντί να φορτώνεται μια web σελίδα, ο browser αρχίζει να εκτελεί κώδικα Javascript (Ajax engine). Ο κώδικας αυτός είναι υπεύθυνος τόσο για την δημιουργία του interface που βλέπει ο χρήστης όσο και για την επικοινωνία με τον server. Αυτό επιτρέπει στην αλληλεπίδραση του χρήστη με το interface της εφαρμογής να γίνεται ασύγχρονα σε σχέση με την επικοινωνία με τον server. Έτσι δεν είναι απαραίτητο να διακόπτεται η πρώτη όσο διαρκεί η δεύτερη.

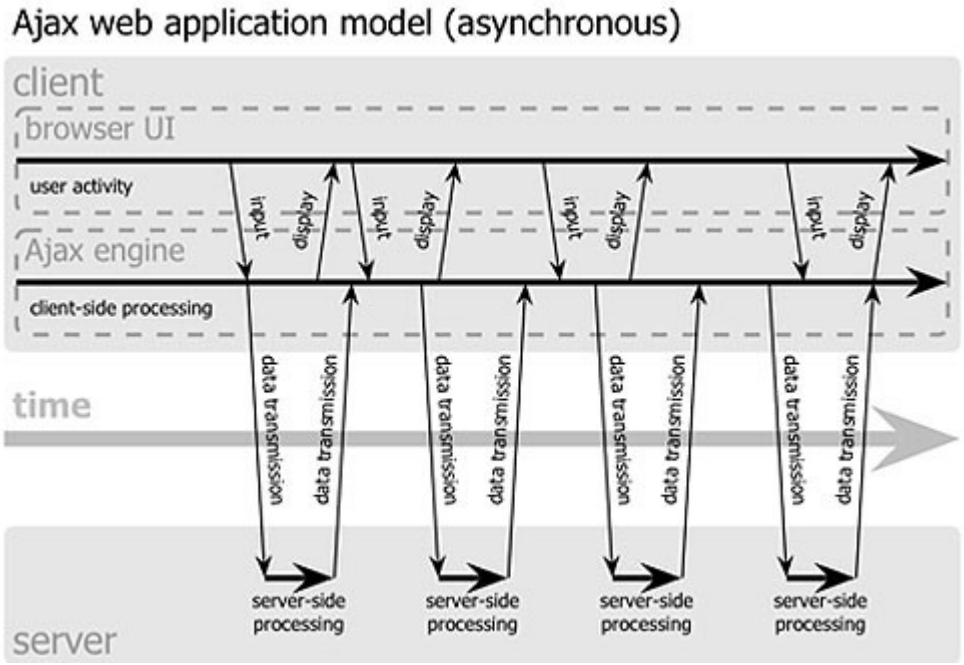
Κάθε ενέργεια του χρήστη η οποία στο παραδιασιακό μοντέλο web συνεδρίας θα οδηγούσε σε ένα HTTP αίτημα παίρνει την μορφή κλήσης μιας Javascript συνάρτησης στο Ajax engine. Όταν αυτό το αίτημα μπορεί να ικανοποιηθεί τοπικά χωρίς την μεσολάβηση του server, το Ajax engine το ικανοποιεί μόνο του. Εάν απαιτεί την μεταφορά καινούργιων δεδομένων, το engine λαμβάνει αυτά τα δεδομένα ασύγχρονα με XMLHttpRequest, χωρίς να διακόπτει τα υπόλοιπα τοπικά αιτήματα. Τα νέα δεδομένα, στην συνηθισμένη περίπτωση που αντιστοιχούν σε περιεχόμενο που πρέπει να απεικονιστεί στο interface, μπορούν να μετατραπούν άμεσα σε XHTML με έναν κατάλληλο XSTL μετασχηματισμό.

Ο όρος Ajax και οι βασικές αρχές του έγιναν γνωστές μέσω του άρθρου (8) Το παρακάτω διάγραμμα προέρχεται από αυτό το άρθρο και συνοψίζει το Ajax μοντέλο web συνεδρίας.

Μερικές από τις πέντε αυτές τεχνολογίες μπορούν να αντικατασταθούν σε κάποιες περιπτώσεις με άλλες παρόμοιας λειτουργικότητας. Αντί για Javascript μπορεί να χρησιμοποιηθεί μια άλλη scripting γλώσσα, αν και ομολογουμένως καμία άλλη δεν υποστηρίζεται τόσο καθολικά από τους browser όσο αυτή. Τα δεδομένα μπορούν να μεταφέρονται σε κάποια άλλη μορφή εκτός XML, η μορφή JSON είναι μια άλλη διαδεδομένη εναλλακτική επιλογή. Επίσης, σε κάποια σενάρια ίσως να είναι καταλληλότερη η σύγχρονη επικοινωνία.

Το Ajax διαφέρει από τις υπόλοιπες RIA τεχνολογίες που αναφερθήκαμε, ως προς το ότι δεν απαιτεί την εγκατάσταση κάποιου runtime στον υπολογιστή του χρήστη, μιας και η Javascript υποστηρίζεται εξαρχής από τους browser. Αυτή η διαφορά είναι αρκετή για πολλούς ώστε να το θεωρούν τεχνολογία ξεχωριστή των RIA. Μπορεί να αξιοποιηθεί πάντως στην επίλυση των ίδιων προβλημάτων.

Στην δική μας περίπτωση, μια λύση βασισμένη στο Ajax θα ήταν εφικτή με την χρήση κάποιας Ajax πλατφόρμας που θα προσέφερε τις UI δυνατότητες που ζητάμε. Τέτοιες πλατφόρμες υπάρχουν, για παράδειγμα η Ext και η Yahoo! UI Library.




Εικόνα 4

# 4. Χρησιμοποιούμενη Τεχνολογία

## 4.1 Microsoft Silverlight / XAML

Το Microsoft Silverlight είναι μια προγραμματιστική πλατφόρμα για την ανάπτυξη κυρίως διαδικτυακών εφαρμογών, με ευκολίες για την δημιουργία RIA ιστότοπων, χειρισμό πολυμεσικού υλικού, γραφικών και animation. Οι δυνατότητές του σε μεγάλο βαθμό ανταγωνίζονται αυτές του Adobe Flash.

Η βιβλιοθήκη του Silverlight βασίζεται σε ένα υποσύνολο της βιβλιοθήκης .NET, περιλαμβανομένο υ του CLR. Αυτό συνεπάγεται πως ο προγραμματισμός σε Silverlight μπορεί να γίνει χρησιμοποιώντας οποιαδήποτε CLI γλώσσα, με συνηθέστερες για αυτόν τον σκοπό τις Visual Basic .NET και C#. Η βιβλιοθήκη αυτή εγκαθίσταται στον υπολογιστή του τελικού χρήστη τυπικά σαν ένα browser plug-in. Οι εφαρμογές Silverlight μπορούν πάντως να εκτελεστούν και εκτός browser, ακόμη και ως Windows Sidebar εφαρμογές (9).

Ένα από τα σημαντικότερα χαρακτηριστικά του Silverlight είναι η χρήση της XAML για την περιγραφή των διεπαφών των εφαρμογών. Η XAML είναι μια XML γλώσσα που έχει φτιάξει επίσης η Microsoft ειδικά για τον σκοπό της σχεδίασης διεπαφής εφαρμογών. Επιτρέπει μεταξύ άλλων τον ορισμό των UI στοιχείων, την περιγραφή της σύζευξης των δεδομένων με την διεπαφή και της δήλωσης των event handlers. Εκτός από το Silverlight, χρησιμοποιείται και από το WPF, που ήταν και η αρχική χρήση της γλώσσας, και το WF.[4]

Τα πλεονεκτήματα της χρήσης μιας ξεχωριστής markup γλώσσας για την περιγραφή της διεπαφής της εφαρμογής, σε σύγκριση με την περιγραφή της απευθείας σε κάποια γενικού σκοπού γλώσσα, είναι αρκετά:

- Οδηγεί σε σαφέστερο και πιο δομημένο κώδικα, μιας και ανεξαρτητοποιείται σε μεγάλο βαθμό ο κώδικας της διεπαφής με τα υπόλοιπα τμήματα της εφαρμογής,

[4] Για το Silverlight, την XAML το WPF, το WP και γενικά όλα τα προγραμματιστικά περιβάλλοντα που αναπτύσσει η Microsoft, πολύ καλή τεκμηρίωση παρέχεται από το msdn.microsoft.com. Για παράδειγμα, μιας σύνοψη της XAML μπορεί να βρεθεί στο (24).

- Μια markup γλώσσα είναι απλούστερη από κάποια γενικού σκοπού όπως η C#, κάτι που κάνει πιο εύκολη την κωδικοποίηση της διεπαφής από developers που ειδικεύονται σε τέτοιες εργασίες (web designers). Υπάρχουν μάλιστα προγράμματα (για παράδειγμα, το Microsoft Expression Blend) που μπορούν να αναπαριστούν τον XAML κώδικα κατευθείαν σε UI στοιχεία και αντίστροφα, επιτρέποντας στον designer να σχεδιάσει την διεπαφή χωρίς να χρειαστεί να γράψει καθόλου κώδικα,
- Η κωδικοποίηση της διεπαφής γίνεται με αυτόν τον τρόπο ανεξάρτητη όχι μόνο από την γλώσσα που χρησιμοποιείται για τα υπόλοιπα τμήματα της εφαρμογής, αλλά ακόμη και από τις συγκεκριμένες τεχνικές απεικόνισης. Για παράδειγμα, ο ίδιος κώδικας XAML μπορεί σε μια περίπτωση να μεταφραστεί σε XHTML ώστε να παρουσιαστεί από έναν browser, και σε μια άλλη περίπτωση σε κώδικα μηχανής ώστε να αποτελέσει την διεπαφή μιας desktop εφαρμογής.
- Επιτρέπει την υιοθέτηση ευέλικτων και απλών αρχιτεκτονικών μοτίβων για την σχεδίαση της διεπαφής. Περισσότερα στοιχεία για τον τρόπο που μπορεί να γίνει αυτό δίνονται στην υποενότητα για το Model-View-ViewModel αρχιτεκτονικό μοτίβο, του επόμενου κεφαλαίου.

Η βάση χρηστών του Silverlight, ο αριθμός δηλαδή των προσωπικών υπολογιστών που έχουν το runtime εγκατεστημένο, δεν είναι ακόμη τόσο μεγάλη όσο αυτή του Adobe Flash, μεγαλώνει όμως συνεχώς. Αν και αρχικά ήταν δυνατή η εκτέλεσή του μόνο σε λειτουργικά της Microsoft, από τον Μάρτιο του 2009 είναι διαθέσιμη και μια υλοποίησή του για Linux και Unix συστήματα. Η υλοποίηση αυτή ονομάζεται Moonlight, αναπτύσσεται από την εταιρία ανοιχτού λογισμικού Novell και έχει την υποστήριξη της Microsoft. Το Moonlight δεν είναι ακόμη συμβατό με τις πλέον πρόσφατες εκδόσεις του Silverlight, παραμένει όμως μια προσπάθεια που ίσως να αποκτήσει ιδιαίτερη σημασία στο άμεσο μέλλον (υπάρχει η προσδοκία πως θα επιτρέψει η Apple την μεταφορά του Moonlight στο iPhone. Μέχρι στιγμής, η Apple απαγορεύει την εκτέλεση browser-based κώδικα στο iPhone, ακόμη και Flash).

## 4.2 Windows Communication Foundation (WCF)

Το WCF είναι ένα υποσύστημα της βιβλιοθήκης .NET που επιτρέπει την ανάπτυξη εφαρμογών προσανατολισμένων σε υπηρεσίες (Service Oriented Architecture, SOA). Σε αδρές γραμμές, στην ανάπτυξη τέτοιων εφαρμογών το ζητούμενο είναι η κατάτμηση της αρχιτεκτονικής σε ανεξάρτητα τμήματα-μονάδες, τα οποία αλληλεπιδρούν μόνο μέσω της χρήσης υπηρεσιών, χρησιμοποιώντας συγκεκριμένα πρωτόκολλα επικοινωνίας. Οι μονάδες αυτά είναι πραγματικά ανεξάρτητες μεταξύ τους, υπό την έννοια πως είναι δυνατό να βρίσκονται σε διαφορετικούς υπολογιστές και σε απομακρυσμένες θέσεις στο δίκτυο. Η καθεμιά εκθέτει την λειτουργικότητά της υπό τη μορφή υπηρεσιών.

Ιδιαίτερη σημασία έχει ο τρόπος που γίνεται η διάθεση αυτών των υπηρεσιών, η κλήση τους, η επιστροφή αποτελεσμάτων, η εξασφάλιση ασφάλειας στην επικοινωνία, η επίλυση προβλημάτων που προκύπτουν όταν τα τμήματα που επικοινωνούν βρίσκονται σε υπολογιστές με διαφορετικό λειτουργικό. Ζητήματα σαν αυτά διευθετεί το WCF.

Υπάρχουν και άλλες τεχνολογίες που καταπιάνονται με τα προβλήματα που προσπαθεί να επιλύσει το WCF[5]. Αναλυτική παρουσίαση των σχεδιαστικών μεθόδων που βρίσκουν εφαρμογή σε όλες τις σχετικές τεχνολογιές μπορεί να βρεθεί στο (10).

Επίσης, είναι πολύ βασική προϋπόθεση η υλοποίηση και ο ορισμός των υπηρεσιών να μην κάνουν καμιά υπόθεση όσον αφορά το ποια θα είναι και πώς θα λειτουργεί η συγκεκριμένη μονάδα που θα τις καλέσει και για ποιον σκοπό. Αυτό είναι βασικό για την επίτευξη πραγματικού διαχωρισμού μεταξύ των μονάδων.

Τα πλεονεκτήματα που μπορεί να προσφέρει η υιοθέτηση SOA αρχιτεκτονικής είναι σημαντικά:

- Προάγει την επαναχρησιμοποίηση των ξεχωριστών μονάδων που απαρτίζουν την εφαρμογή. Σε μια εφαρμογή που όντως υλοποιεί την SOA αρχιτεκτονική, οι υπηρεσίες που παρέχει η κάθε μονάδα δεν κάνουν υποθέσεις ως προς την φύση των πελατών της. Έτσι, οι υπηρεσίες μπορούν να χρησιμοποιηθούν από πολλούς διαφορετικούς πελάτες και με τρόπους που δεν είχαν κατ' ανάγκην προβλεφθεί στον αρχικό σχεδιασμό, χωρίς να χρειάζεται κάποια μετατροπή στη

[5] Σημαντική προσπάθεια είναι το SCA, πρότυπο που υποστηρίζεται, μεταξύ άλλων, από την IBM και την Oracle. Αναλυτική παρουσίαση των σχετικών τεχνολογίων γίνεται στο (10)

μονάδα που τις προσφέρει. Τα πρωτόκολλα μεταφοράς δεδομένων που χρησιμοποιούν τα εργαλεία SOA είναι μάλιστα έτσι σχεδιασμένα, ώστε να μην απαιτούν πελάτης και εξυπηρετητής να εκτελούνται σε όμοιο λειτουργικό σύστημα ή να έχουν υλοποιηθεί σε όμοια γλώσσα προγραμματισμού, επιτρέποντας μεγάλη ευελιξία στην σχεδίαση των πελατών.

- Οι διαθέσιμες τεχνολογίες SOA παρέχουν έτοιμες λύσεις σε δύσκολα προβλήματα που προκύπτουν κατά την επικοινωνία πελάτη-υπηρεσιών. Έτσι, ζητήματα όπως η κρυπτογράφηση της επικοινωνίας, η ταυτόχρονη υποστήριξη πολλών διαφορετικών πρωτοκόλλων μεταφοράς δεδομένων, η στοιχειοθέτηση αρχείων καταγραφής των μηνυμάτων που μεταφέρονται, η διατήρηση διαφορετικών εκδόσεων των υπηρεσιών και πολλά άλλα, μπορούν να διευθετηθούν απλώς με την παραμετροποίηση του εργαλείου SOA που χρησιμοποιείται, χωρίς την ανάγκη συγγραφής πραγματικού κώδικα.
- Η SOA αρχιτεκτονική πρόκειται εν πολλοίς για την συστηματική εφαρμογή πρακτικών σχεδίασης των οποίων η αξία έχει τεκμηριωθεί από παλιά. Μια SOA εφαρμογή αναγκάζεται να τηρήσει τις σχετικές πρακτικές, και άρα να επωφεληθεί από αυτές. Για παράδειγμα, η «χαλαρή» σύζευξη (loose-coupling) των μονάδων μεταξύ τους επιτρέπει τον εύκολο έλεγχο και αποσφαλμάτωσή τους, αφού η κάθε μονάδα μπορεί να ελεγχθεί ξεχωριστά από τις υπόλοιπες, και η λειτουργικότητά τους εκτίθεται σαφώς μέσω των υπηρεσιών.

Αν και η αρχιτεκτονική SOA μπορεί να εφαρμοστεί σε ευρύ φάσμα εφαρμογών, ακόμη σε desktop εφαρμογές, σχεδόν πάντα χρησιμοποιείται όταν υπάρχουν τμήματα της εφαρμογής που τρέχουν σε διαφορετικούς υπολογιστές στο διαδίκτυο. Σε τέτοιες περιπτώσεις, οι SOA υπηρεσίες αναφέρονται συνήθως ως Web Services.

Ένα σημαντικό στοιχείο για τα Web Services είναι πως υπάρχει για αυτό ένα κοινά αποδεκτό πρότυπο για την περιγραφή τους, το WSDL, το οποίο έχει εκδοθεί από το W3C. Το WCF, καθώς και οι άλλες SOA τεχνολογίες, είναι συμβατές με αυτό το πρότυπο.

## 4.3 HTTP Secure

Το πρωτόκολλο HTTP Secure (HTTPS) είναι η πλέον διαδεδομένη μέθοδος που χρησιμοποιείται στο διαδίκτυο για την δημιουργία

ασφαλούς σύνδεσης. Τεχνικά, δεν πρόκειται για ξεχωριστό πρωτόκολλο, αλλά αναφέρεται στην χρήση του συνηθισμένου HTTP πάνω από κάποιο κρυπτογραφημένο επίπεδο μεταφοράς (SSL ή TLS), μαζί με μια τυποποιημένη διαδικασία για την ανάκτηση και την πιστοποίηση ψηφιακών πιστοποιητικών.

Με το HTTPS είναι δυνατή η ταυτοποίηση και των δύο μελών που επιχειρούν να επικοινωνήσουν και η παρεμπόδιση επιθέσεων τύπου man-in-the-middle.

Όλοι οι διαδεδομένοι browsers (Internet Explorer, Mozilla Firefox, Safari, Chrome) υποστηρίζουν πρόσφατες εκδόσεις του HTTPS μέσω TLS και, σε περίπτωση που κάτι δεν φαίνεται να πηγαίνει σωστά με την διαδικασία ταυτοποίησης, την διακόπτουν και ενημερώνουν σχετικά τον χρήστη.

## 4.4 Λόγοι Επιλογής Αυτών των Τεχνολογιών

Η ανάγκη κρυπτογράφησης της σύνδεσης της εφαρμογής που εκτελείται στον browser του χρήστη με τον εξυπηρετητή κάνει την χρήση του HTTPS μονόδρομο, αφού πρόκειται για το de facto πρότυπο για αυτόν τον σκοπό.

Ένας πολύ σημαντικός παράγοντας για την επιλογή των υπόλοιπων τεχνολογιών είναι η λειτουργική απαίτηση η εφαρμογή να χρησιμοποιεί ακριβώς την ίδια βάση δεδομένων με εκείνη του προϋπάρχοντος συστήματος. Η βάση αυτή βρίσκεται σε έναν Microsoft SQL Server 2008, κάνοντας την απρόσκοπη επικοινωνία με αυτόν έναν πολύ βασικό περιοριστικό παράγοντα για την επιλογή των υπόλοιπων τεχνολογιών.

Πιο συγκεκριμένα, είναι γεγονός πως η Microsoft έχει καταφέρει, μέσω της πλατφόρμας Visual Studio, να ολοκληρώσει των προϊόντων της που απευθύνονται σε προγραμματιστές σε ένα ενιαίο περιβάλλον. Μέσω του Visual Studio, είναι πολύ απλό κάποιος να φτιάξει αναπαραστάσεις (Entity Models) μιας SQL Server βάσης δεδομένων, οι οποίες αναπαραστάσεις μπορούν άμεσα μετά να χρησιμοποιηθούν από WCF υπηρεσίες. Όμοια, το Visual Studio παρέχει έτοιμα templates για projects που σκοπεύουν να χρησιμοποιήσουν το Silverlight, και κάνει πολύ εύκολη την σύνδεση της Silverlight εφαρμογής με τις WCF. Μάλιστα, η τελευταία έκδοση του Visual Studio, το Visual Studio 2010, έχει ενσωματωμένο ένα εργαλείο που απεικονίζει άμεσα της XAML σελίδες,

κάνοντας περιττή, για απλές εργασίες, την χρήση επιπλέον προγράμματος για την σχεδίαση του interface.

Να υπενθυμίσουμε επίσης πως υποθέτουμε για τους προσωπικούς υπολογιστές των τελικών χρηστών πως τρέχουν λειτουργικό Microsoft Windows ή Intel Mac. Τα λειτουργικά αυτά υποστηρίζονται από το Silverlight, κάνοντας περιττή την χρήση άλλων τεχνολογιών (Adobe Flex) ειδικά για τον λόγο υποστήριξης και άλλων συστημάτων.

Μένοντας στους λόγους επιλογής του Silverlight, να παρατηρήσουμε πως πρόκειται για τεχνολογία πιο ώριμη από το ακόμη πολύ νέο JavaFX, και με πολύ περισσότερες δυνατότητες από λύσεις βασισμένες σε Ajax. Όσον αφορά το παρόμοιων δυνατοτήτων Adobe Flex το κριτήριο επιλογής για το Silverlight είναι αυτό που επισημάναμε πιο πριν: Η καλύτερη ολοκλήρωσή του με τις υπόλοιπες τεχνολογίες τις Microsoft, τις οποίες εκ των πραγμάτων καλούμαστε να χρησιμοποιήσουμε.

Τέλος, ο web server που χρειάζεται η εφαρμογή είναι ένας Internet Information Services 7.0, που είναι και η πιο συνηθισμένη επιλογή για την φιλοξενία WCF υπηρεσιών. Να σημειώσουμε πως η χρήση του web server που είναι ενσωματωμένος στο Visual Studio, του ASP.NET Development Server (ο οποίος ούτως ή άλλως γενικά ενδείκνυται μόνο για αρχικές φάσεις ανάπτυξης μιας εφαρμογής) δεν ήταν δυνατή στην περίπτωσή μας, επειδή ο εν λόγω server δεν υποστηρίζει το HTTPS πρωτόκολλο.

# 5. Υλοποίηση

## 5.1 Η Αρχιτεκτονική του Εξυπηρετητή

### 5.1.2 Η Βάση Δεδομένων

Η βάση δεδομένων παραμένει αυτή της προϋπάρχουσας εφαρμογής, όπως ήταν άλλωστε και μία από τις λειτουργικές απαιτήσεις.

Είναι αποθηκευμένη σε έναν Microsoft SQL Server 2008. Ο SQL Server δεν είναι απαραίτητο να βρίσκεται στο ίδιο μηχάνημα με αυτό που εκτελείται ο κώδικας των WCF Υπηρεσιών, αλλά οπουδήποτε στο τοπικό δίκτυο.

Έχει σχεδιαστεί έτσι ώστε να κάνει εύκολη την άντληση και αποθήκευση πληροφοριών σχετικά με την διαχείριση έργου, σύμφωνα με την μεθοδολογία που περιγράφηκε.

Ακολουθεί μια αναλυτική περιγραφή της, αντλημένη σε μεγάλο βαθμό από την αρχική τεκμηρίωσή της (11).

Η βάση αποτελείται από είκοσι πίνακες εκ των οποίων οι τέσσερις αποτελούν πίνακες αναζήτησης (lookup tables) που δεν μπορούν να τροποποιηθούν από τις WCF υπηρεσίες αλλά βοηθούν στο να είναι η βάση σε συνεπή κατάσταση (database consistency) .

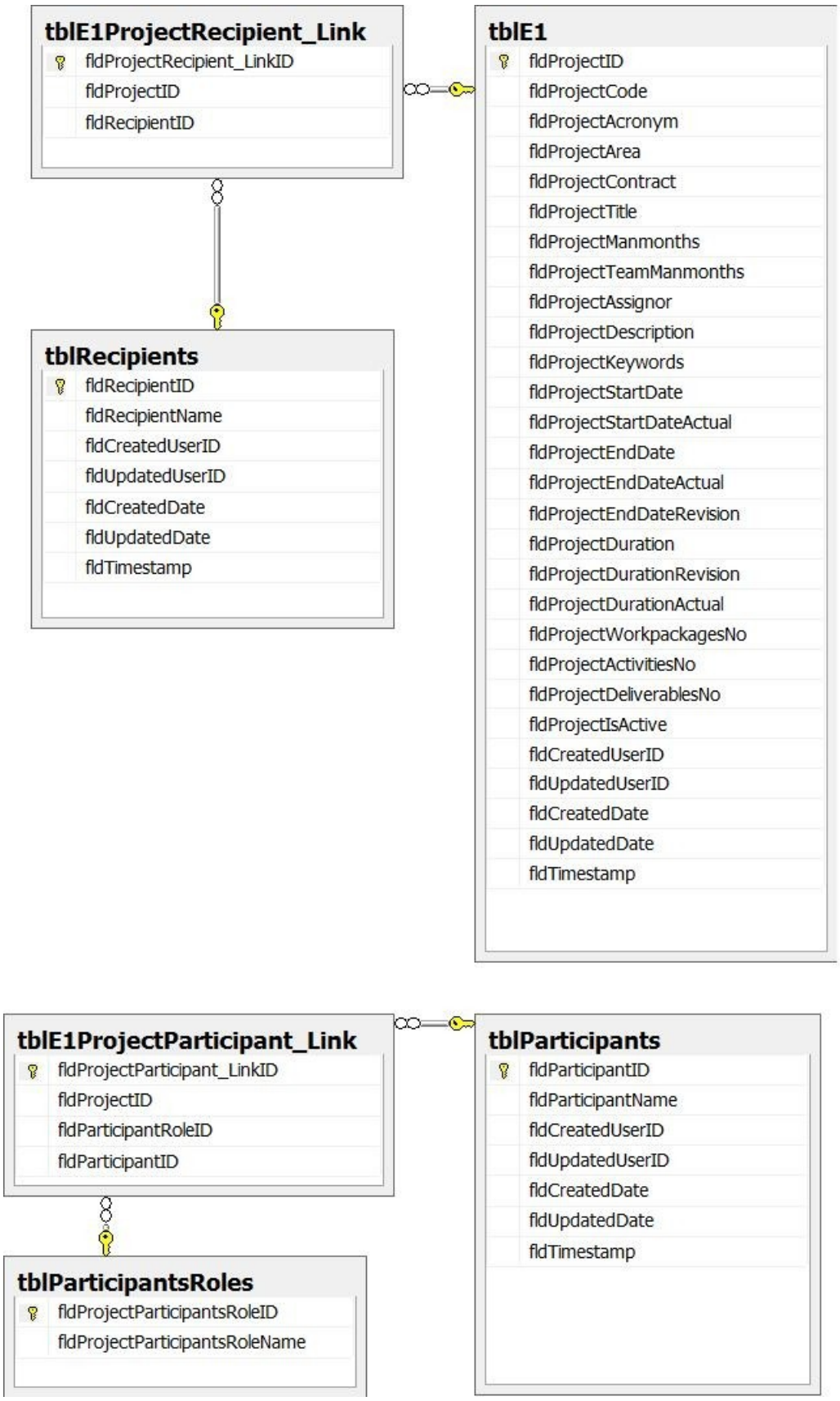
Η σχεδίασή της προσπαθεί να επιτύχει δύο στόχους: την αποφυγή διπλοεγγραφών και λοιπών περιττών στοιχείων και την καλή συσχέτιση των πινάκων για την αποφυγή μη έγκυρων εγγραφών.

Ο πρώτος στόχος εκπληρώνεται οριστικοποιώντας τις φόρμες που θα χρησιμοποιηθούν για την εισαγωγή των δεδομένων και στην συνέχεια ψάχνοντας τα πεδία τα οποία πηγάζουν από άλλες φόρμες. Αυτά τα πεδία

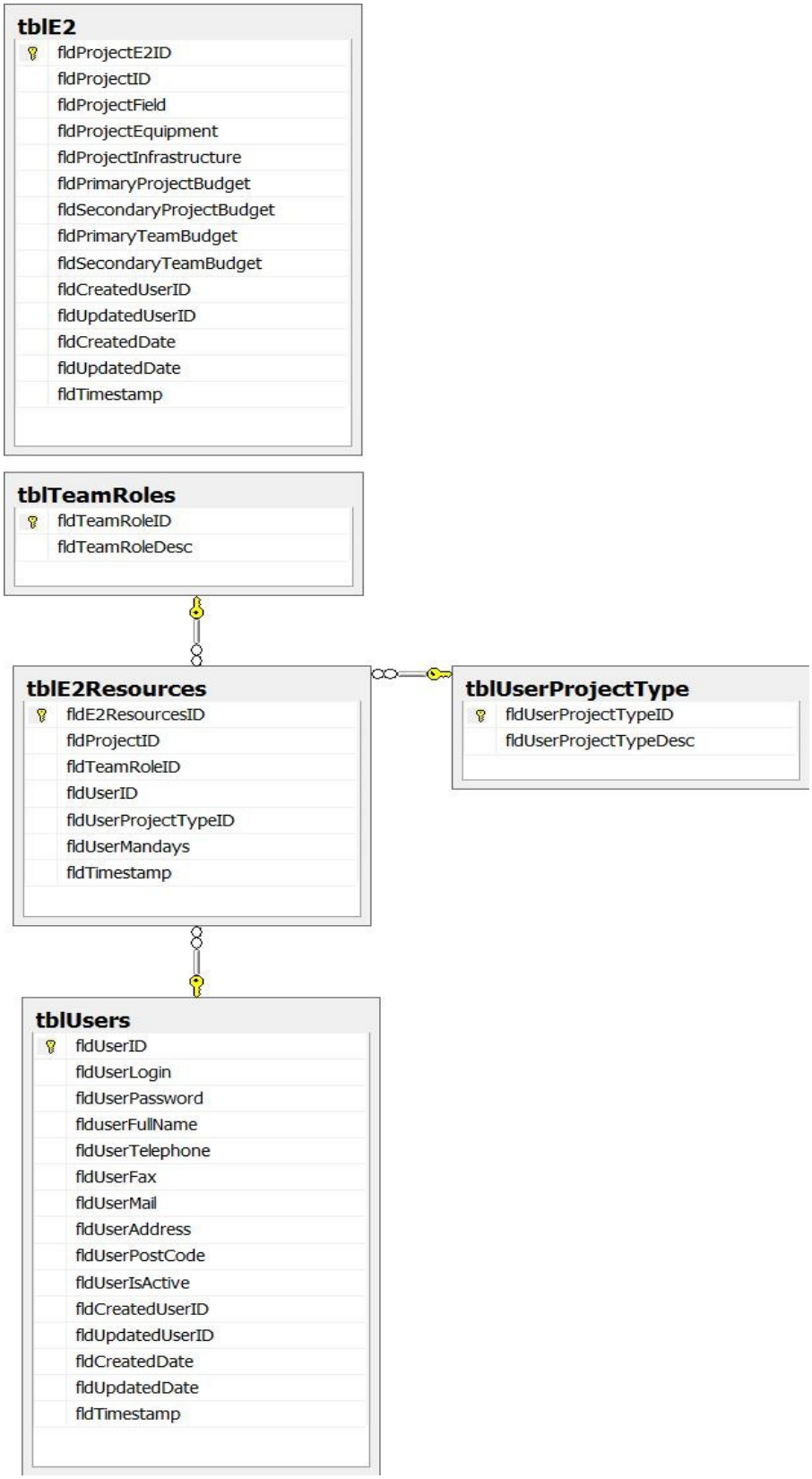
ενοποιήθηκαν σε ένα πεδίο της βάσης και από εκεί αντλείται η πληροφορία που ζητείται καλώντας συναρτήσεις από την εφαρμογή. Χαρακτηριστικό παράδειγμα είναι η ημερομηνία συμπλήρωσης του εντύπου Ε5. Γνωρίζοντας αυτή την πληροφορία, μπορούμε πολύ εύκολα να δούμε κατά την διάρκεια ποιας εβδομάδας συμπληρώθηκε με αποτέλεσμα να γνωρίζουμε ότι ο απολογισμός αναφέρεται στην εβδομάδα κατά την οποία συμπληρώθηκε ενώ η πρόβλεψη αναφέρεται στην επόμενη εβδομάδα.

Για την επίτευξη του δεύτερου στόχου, έγινε συστηματική μελέτη των πεδίων και αναζήτηση των συσχετίσεων αυτών. Μετά από αυτό ομαδοποιήθηκαν τα πεδία σε πίνακες και ορίστηκαν τα κλειδιά που χαρακτηρίζουν μοναδικά τον εκάστοτε πίνακα. Στην συνέχεια εισάγαμε τις κατάλληλες συσχετίσεις (relationships) και φροντίζουμε με αυτές να γίνεται έλεγχος της εγκυρότητας των δεδομένων προγραμματίζοντας τον sql server να ελέγχει τις συσχετίσεις πριν εισάγει νέα δεδομένα και να αφαιρεί τα δεδομένα που αναφέρονταν σε μια γραμμή που διαγράφηκε.

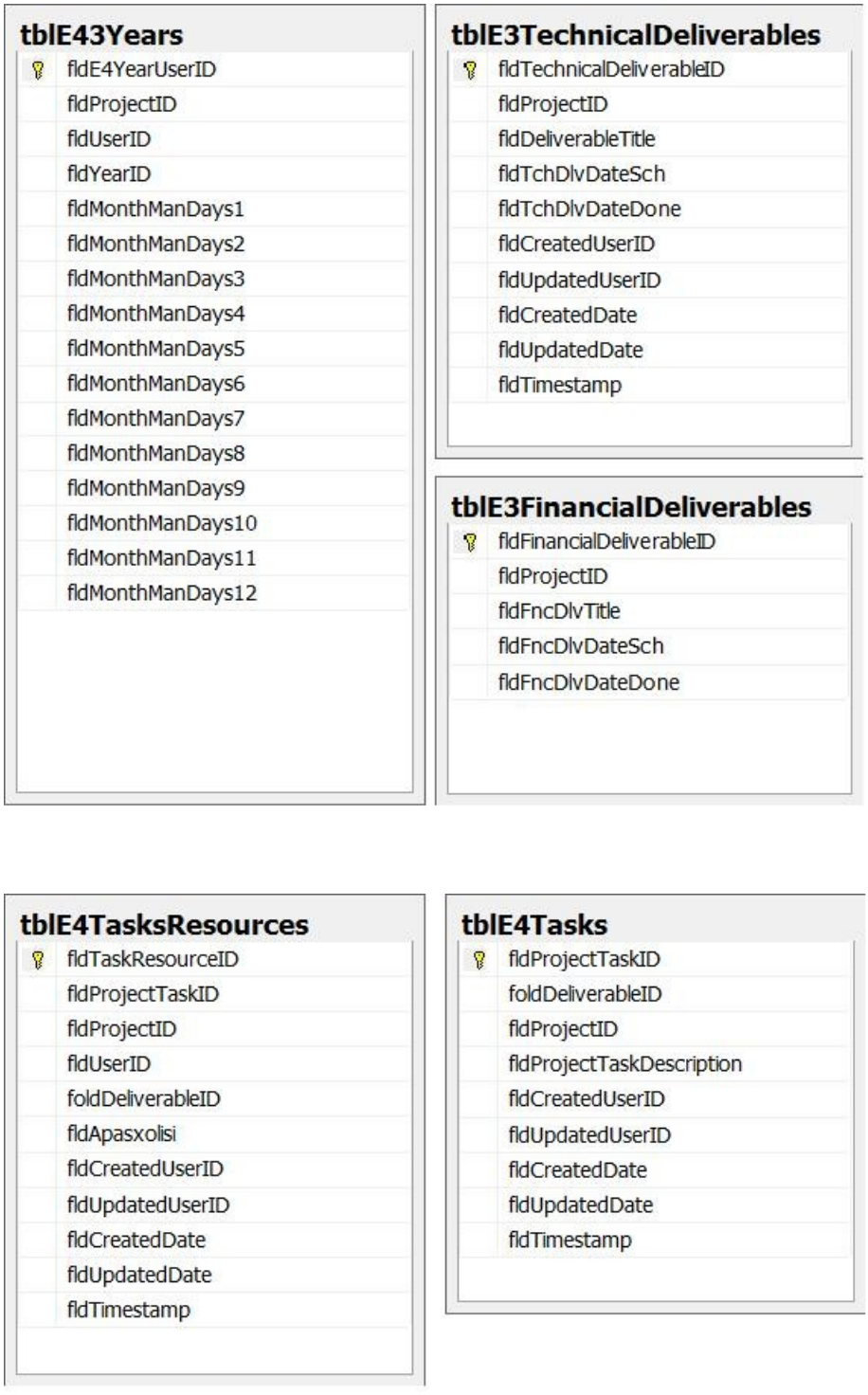
Στην συνέχεια παρατίθενται διαγράμματα με όλους τους πίνακες και τις συσχετίσεις τους.


tblE1ProjectRecipient_Link
fldProjectRecipient_LinkID
fldProjectID
fldRecipientID
tblE1
fldProjectID
fldProjectCode
fldProjectAcronym
fldProjectArea
fldProjectContract
fldProjectTitle
fldProjectManmonths
fldProjectTeamManmonths
fldProjectAssignor
fldProjectDescription
fldProjectKeywords
fldProjectStartDate
fldProjectStartDateActual
fldProjectEndDate
fldProjectEndDateActual
fldProjectEndDateRevision
fldProjectDuration
fldProjectDurationRevision
fldProjectDurationActual
fldProjectWorkpackagesNo
fldProjectActivitiesNo
fldProjectDeliverablesNo
fldProjectIsActive
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblRecipients
fldRecipientID
fldRecipientName
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblE1ProjectParticipant_Link
fldProjectParticipant_LinkID
fldProjectID
fldParticipantRoleID
fldParticipantID
tblParticipants
fldParticipantID
fldParticipantName
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblParticipantsRoles
fldProjectParticipantsRoleID
fldProjectParticipantsRoleName

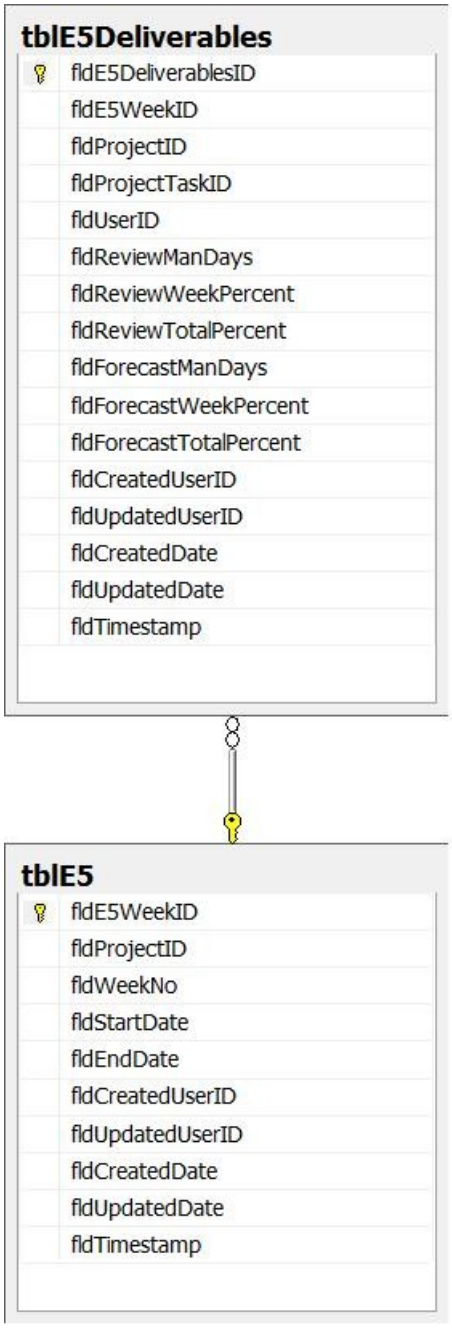
Εικόνα 5


tblE2
fldProjectE2ID
fldProjectID
fldProjectField
fldProjectEquipment
fldProjectInfrastructure
fldPrimaryProjectBudget
fldSecondaryProjectBudget
fldPrimaryTeamBudget
fldSecondaryTeamBudget
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblTeamRoles
fldTeamRoleID
fldTeamRoleDesc
tblE2Resources
fldE2ResourcesID
fldProjectID
fldTeamRoleID
fldUserID
fldUserProjectTypeID
fldUserMandays
fldTimestamp
tblUserProjectType
fldUserProjectTypeID
fldUserProjectTypeDesc
tblUsers
fldUserID
fldUserLogin
fldUserPassword
flduserFullName
fldUserTelephone
fldUserFax
fldUserMail
fldUserAddress
fldUserPostCode
fldUserIsActive
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp

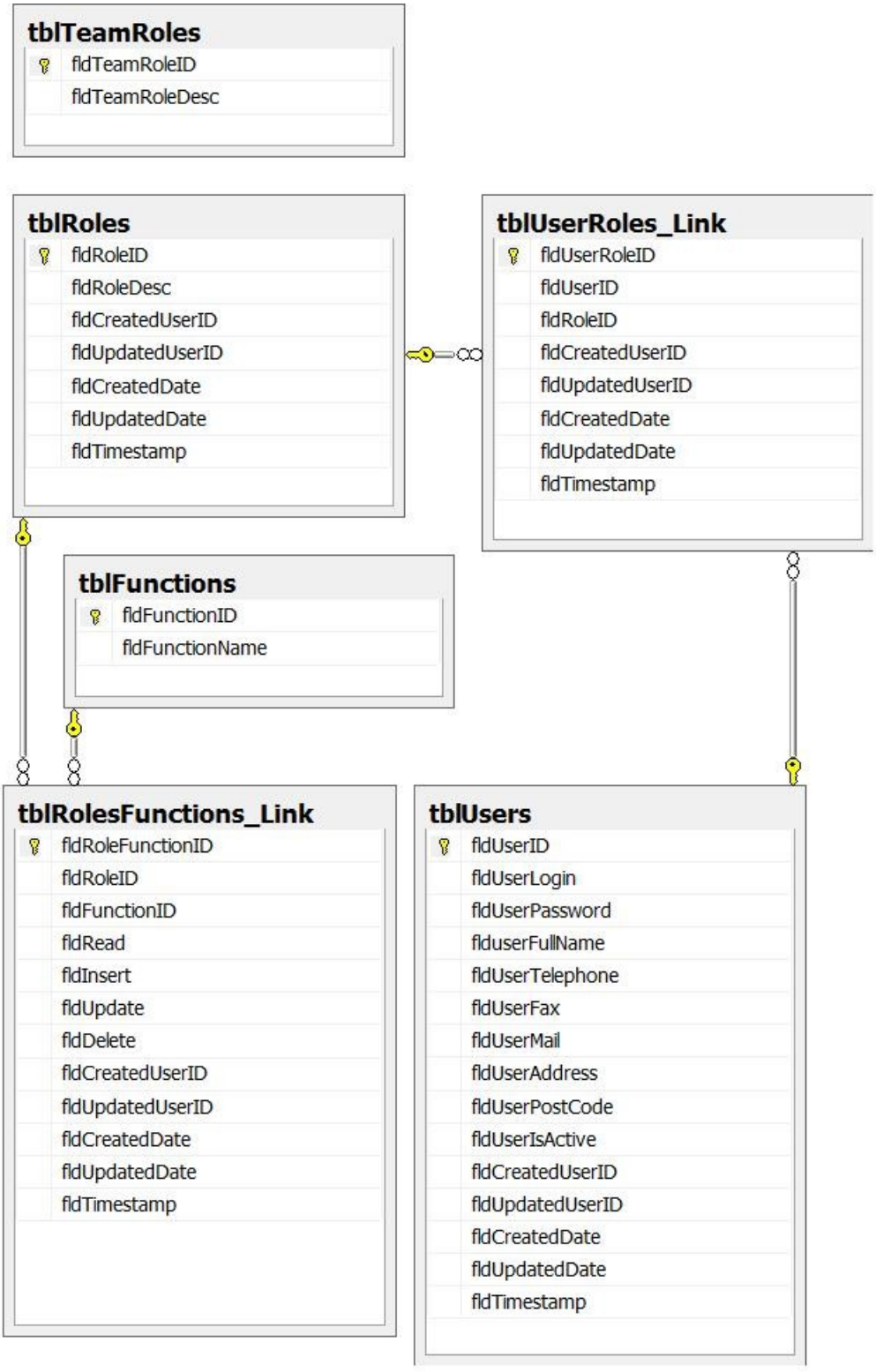
Εικόνα 6


tblE43Years
fldE4YearUserID
fldProjectID
fldUserID
fldYearID
fldMonthManDays1
fldMonthManDays2
fldMonthManDays3
fldMonthManDays4
fldMonthManDays5
fldMonthManDays6
fldMonthManDays7
fldMonthManDays8
fldMonthManDays9
fldMonthManDays10
fldMonthManDays11
fldMonthManDays12
tblE3TechnicalDeliverables
fldTechnicalDeliverableID
fldProjectID
fldDeliverableTitle
fldTchDlvDateSch
fldTchDlvDateDone
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblE3FinancialDeliverables
fldFinancialDeliverableID
fldProjectID
fldFncDlvTitle
fldFncDlvDateSch
fldFncDlvDateDone
tblE4TasksResources
fldTaskResourceID
fldProjectTaskID
fldProjectID
fldUserID
foldDeliverableID
fldApasxolisi
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblE4Tasks
fldProjectTaskID
foldDeliverableID
fldProjectID
fldProjectTaskDescription
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp

Εικόνα 7


tblE5Deliverables
fldE5DeliverablesID
fldE5WeekID
fldProjectID
fldProjectTaskID
fldUserID
fldReviewManDays
fldReviewWeekPercent
fldReviewTotalPercent
fldForecastManDays
fldForecastWeekPercent
fldForecastTotalPercent
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblE5
fldE5WeekID
fldProjectID
fldWeekNo
fldStartDate
fldEndDate
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp

**Εικόνα 8**


tblTeamRoles
fldTeamRoleID
fldTeamRoleDesc
tblRoles
fldRoleID
fldRoleDesc
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblUserRoles_Link
fldUserRoleID
fldUserID
fldRoleID
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblFunctions
fldFunctionID
fldFunctionName
tblRolesFunctions_Link
fldRoleFunctionID
fldRoleID
fldFunctionID
fldRead
fldInsert
fldUpdate
fldDelete
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
tblUsers
fldUserID
fldUserLogin
fldUserPassword
flduserFullName
fldUserTelephone
fldUserFax
fldUserMail
fldUserAddress
fldUserPostCode
fldUserIsActive
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp

Εικόνα 9

Η συγκέντρωση των πεδίων στους αντίστοιχους πίνακες έγινε με βάση το έντυπο στο οποίο συμπληρώνονται από τους υπεύθυνους του έργου. Ειδικότερα, οι παραπάνω πίνακες αναλύονται ως εξής:

*tblE1*

Το πεδίο fldProjectID είναι αυτό που διαχωρίζει τα έργα μεταξύ τους. Είναι ένας μοναδικός αριθμός που ανατίθεται σε κάθε έργο. Τα υπόλοιπα πεδία αποτελούν τα κύρια στοιχεία που χαρακτηρίζουν ένα έργο δηλαδή ο κωδικός του, το ακρωνύμιο, η επιστημονική περιοχή στην οποία υπάγεται, ο αριθμός συμβολαίου και ο τίτλος του. Άμα διαγραφεί μία γραμμή από τον πίνακα, αμέσως διαγράφονται οι αντίστοιχες εγγραφές που αναφέρονται στο συγκεκριμένο project. Σημαντική λεπτομέρεια είναι το γεγονός ότι κατά την δημιουργία ενός έργου πρέπει να δημιουργηθούν όλοι οι κύριοι πίνακες που προσδιορίζουν το έργο, δηλαδή οι πίνακες tblE1, tblE2, tblE3, tblE4, tblE5 καθώς και τα αντίστοιχα version 1 σε όσα έντυπα υποστηρίζεται.

Επίσης, σε αυτόν τον πίνακα αποθηκεύονται τα βασικά στοιχεία του εντύπου E1 είναι μονοσήμαντα ορισμένα. Πέρα όμως αυτών των στοιχείων, υπάρχουν και ορισμένα άλλα στοιχεία τα οποία δεν μπορούν να αποθηκευτούν μέσα στον ίδιο πίνακα. Αυτό συμβαίνει λόγω της πολλαπλής συσχέτισης που υπάρχει (one to many) με το κλειδί fldProjectID. Αυτά τα στοιχεία είναι οι παραλήπτες και οι συμμετέχοντες των οποίων οι πίνακες αναλύονται στην συνέχεια.

*tblRecipients*

Εδώ αποθηκεύονται οι αποδέκτες του έργου. Το γεγονός ότι δεν αναφέρεται το όνομα του παραλήπτη παρά ένα νούμερο κλειδί που χαρακτηρίζει τον παραλήπτη, οφείλεται στο γεγονός ότι οι παραλήπτες είθισται να επαναλαμβάνονται σε διάφορα έργα οπότε το ενδεχόμενο να γινόταν η εγγραφή του ονόματος (τύπου nvarchar) για κάθε παραλήπτη και για κάθε έργο, θα επιβάρυνε σημαντικά τον όγκο της βάσης. Συνεπώς έχουμε τον πίνακα tblRecipients σαν πίνακα αναζήτησης των ονομάτων (lookup table).

*tblParticipantsRoles*

Κατά τα πρότυπα του πίνακα tblRecipients, ο πίνακας αυτός έχει σχεδιαστεί σαν συνδετικός κρίκος μεταξύ του κλειδιού που χαρακτηρίζει το έργο, το όνομα του συμμετέχοντα και του ρόλου του στο έργο αυτό.

*tblParticipants*

Εδώ αποθηκεύονται όλοι όσοι συμμετέχουν στα διάφορα έργα. Από αυτόν τον πίνακα δημιουργείται η λίστα από την οποία μπορούν να επιλέξουν οι υπεύθυνοι σύμβασης τους συμμετέχοντες στο έργο τους.

*tblE2*

Σε αυτόν τον πίνακα γίνονται οι εγγραφές των δεδομένων από το έντυπο E2.

*tblUserRoles*

Οι ρόλοι που μπορεί να έχουν τα μέλη της ομάδας εργασίας είναι υπεύθυνος έργου, υπεύθυνος σύμβασης και μέλος ομάδας εργασίας. Κάθε άτομο μπορεί να έχει περισσότερους από έναν ρόλους. Βέβαια, το τι ρόλο έχει στο εκάστοτε έργο, καθορίζεται από το έντυπο E2 και μόνο από αυτό.

*tblUserType*

Όντας ένας ακόμα lookup table, περιέχει τρεις μόνο εγγραφές που συνιστούν στην συνέπεια της βάσης. Ένα μέλος της ομάδας μπορεί να είναι είτε μέλος της ομάδας, ή εξωτερικός συνεργάτης ή μέλος διοικητικής.

*tblE3TechnicalDeliverables*

Επειδή υπάρχει σχέση one to many όσον αφορά τόσο τα τεχνικά παραδοτέα όσο και τα οικονομικά παραδοτέα του έργου, αυτά τα στοιχεία καταχωρούνται σε ξέχωρους πίνακες στους οποίους αναγράφονται και το version στο οποίο αναφέρονται.

*tblE3FinancialDeliverables*

Εδώ αναγράφονται οι εγγραφές για τα οικονομικά παραδοτέα του έργου.

*tblE4*

Το έντυπο Ε4 αποτελείται κυρίως από one to many συσχετίσεις οπότε ο tblE4 υπάρχει μόνο για να υπάρχει database consistency και να αποθηκεύει στοιχεία για την συμπλήρωση του εντύπου. Εάν σε μετέπειτα έκδοση θεωρηθεί χρήσιμο να υπάρξει υποστήριξη εκδόσεων και σε αυτό το έντυπο, τότε ο αριθμός της έκδοσης και το εάν είναι ενεργή θα μπει σε αυτόν τον πίνακα.

*tblE43Years*

Ο μακροχρόνιος προγραμματισμός των ανθρωποημερών που θα ξοδέψει το κάθε μέλος ομάδας εργασίας (Μ.Ο.Ε.) για το συγκεκριμένο έργο αποθηκεύονται σε αυτόν τον πίνακα. Για να μπορέσει το σύστημα να υπολογίσει τα αθροίσματα των ανθρωποημερών ανά μήνα για κάθε Μ.Ο.Ε. έπρεπε να γίνεται η συμπλήρωση του εντύπου με κάποια τυποποίηση. Αυτή αποφασίστηκε να είναι ανά έτος με αποτέλεσμα όταν ο υπεύθυνος έργου θελήσει να συμπληρώσει ένα έτος θα συμπληρώσει μόνο τους μήνες στους οποίους έχει προγραμματίσει ένα Μ.Ο.Ε. να δουλέψει και όλους τους υπόλοιπους θα τους συμπληρώσει αυτόματα το σύστημα με μηδέν.

*tblE5*

Όπως έχει αναφερθεί, το έντυπο Ε5 αποτελεί την βασική οντότητα για τον εβδομαδιαίο προγραμματισμό της ομάδας. Από αυτόν τον πίνακα γίνεται το listing των εβδομάδων μαζί με τις εκδόσεις των εντύπων μέσα στην εφαρμογή.

*tblE5Deliverables*

Ο tblE5Deliverables είναι η δομική μονάδα του εντύπου Ε5. Το κάθε μέλος της ομάδας έχει μία εγγραφή σε αυτόν τον πίνακα για κάθε αντικείμενο εργασίας με το οποίο θα ασχοληθεί.

*tblUsers*

Μέσω αυτού του πίνακα προσδιορίζεται πλήρως η οντότητα του κάθε χρήστη στο σύστημα. Να σημειωθεί ότι στο fldUserPwd ο κωδικός αποθηκεύεται κρυπτογραφημένος σε MD5 έτσι ώστε να προστατευτεί το

απόρρητο του συνθηματικού, ακόμα και από τους διαχειριστές της εφαρμογής.

*tblRoles*

Εδώ αποθηκεύονται οι ρόλοι που έχουν οι χρήστες στο σύστημα. Κάθε χρήστης μπορεί να έχει περισσότερους από ένα ρόλους και γι' αυτό αναγκαστικά αυτή η πληροφορία έπρεπε να συμπληρώνεται σε ξεχωριστό πίνακα.

*tblUserRoles*

Ο παρόν πίνακας αποτελεί έναν πίνακα αναζήτησης για τους ρόλους του συστήματος που μπορούν να έχουν οι χρήστες.

*tblFunctions*

Οι λειτουργίες που μπορεί να έχει ένας ρόλος συστήματος.

*tblRolesFunction*

Συνδέει τους ρόλους συστήματος με τις λειτουργίες στις οποίες αντιστοιχούν.

### 5.1.3 Η Entity-Relationship-Model Αναπαράσταση της Βάσης Δεδομένων

Σε κάθε enterprice εφαρμογή, ενυπάρχει μια εγγενής ασυμφωνία: Από τη μία υπάρχει ο κώδικας της εφαρμογής, ο οποίος χρησιμοποιεί κλάσεις αντικειμένων για να περιγράψει την επιχειρησιακή λογική της εφαρμογής και τον τρόπο που αυτή μεταχειρίζεται και μεταβάλει τα δεδομένα. Από την άλλη υπάρχουν τα πραγματικά δεδομένα, τα οποία είναι αποθηκευμένα σε μορφή ασύμβατη με αυτή που τα μπορεί να μεταχειριστεί ο κώδικας του επιχειρησιακής λογικής, στην περίπτωσή μας ως εγγραφές μιας relational βάσης δεδομένων.

Η ασυμφωνία αυτή και οι συνέπειες της μπορούν να γίνουν κατανοητές

με ένα παράδειγμα.

Ας υποθέσουμε πως θέλουμε να εκτελέσουμε στον κώδικα την εξής εργασία: μας δίνεται το Login που χρησιμοποιεί ένας χρήστης για να συνδεθεί στο σύστημα και μας ζητείται να βρούμε όλα τα έργα στα οποία συμμετέχει αυτός ο χρήστης και τους ρόλους του σε καθένα από αυτά.

Όπως είδαμε στην προηγούμενη ενότητα, οι χρήστες του συστήματος αποθηκεύονται στη βάση δεδομένων ως εγγραφές του πίνακα tblE1 και οι πληροφορίες συμμετοχής τους στα έργα ως εγγραφές του πίνακα tblE2Resources.

Αν δεν υπάρχει ενδιάμεσο επίπεδο ανάμεσα στη βάση δεδομένων και στον κώδικα που εκτελεί την εργασία, τότε είμαστε υποχρεωμένοι να αλληλεπιδράσουμε απευθείας με την βάση. Κάτι τέτοιο απαιτεί τα εξής βήματα:

1) αρχικοποίηση μιας σύνδεσης με την βάση δεδομένων.

2) αποστολή ενός ερωτήματος SQL στη βάση για την εύρεση του ID του χρήστη που μας ενδιαφέρει. Το ερώτημα αυτό είναι το:

```
select fldUserID from Isotrack.dbo.tblUsers where
tblUsers.fldUserLogin="GivenString"
```

3) χρήση του ID που επέστρεψε το προηγούμενο ερώτημα για την άντληση των πληροφοριών που μας ενδιαφέρουν. Αυτό γίνεται με έναν νέο ερώτημα SQL:

```
select * from Isotrack.dbo.tblE2Resources where fldUserID=FoundID
```

Το πρώτο μειονέκτημα της προσέγγισης αυτής είναι πως θα πρέπει κάθε φορά να ασχολιόμαστε με τις λεπτομέρειες έναρξης σύνδεσης με την βάση δεδομένων. Το δεύτερο είναι πως καλούμαστε να ενσωματώσουμε συμβολοσειρές ερωτημάτων SQL στον κώδικα, των οποίων η ορθότητα είναι δύσκολο να ελεγθεί.
Τα δύο αυτά ζητήματα μπορούν να αντιμετωπιστούν χωρίς να καταφύγουμε σε ένα Entity μοντέλο. Πράγματι, θα μπορούσαμε να φτιάξουμε κλάσεις που να κάνουν εύκολη την δημιουργία συνδέσεων με τη βάση. Όσον αφορά τα SQL ερωτήματα, αυτά μπορούν να αντικατασταθούν με LINQ ερωτήματα (στοιχεία για το LINQ παρουσιάζονται στην συνέχεια της ενότητας), τα οποία είναι πολύ ευκολότερα στη χρήση και μπορούν να ελεγχθούν ως προς την ορθότητα σε χρόνο μεταγλώττισης.
Το πρόβλημα όμως που παραμένει είναι πως οι πληροφορίες που αντλούμε από την βάση επιστρέφονται σε μορφή πολύ δύσχρηστη για την εκτέλεση σύνθετων εργασιών. Το δεύτερο ερώτημα στην βάση,

ακόμη και αν γίνει με χρήση του LINQ, θα μας επιστρέψει τις πληροφορίες συμμετοχής του χρήστη στα έργα σαν μια λίστα από λίστες δεδομένων. Η μόνη πληροφορία που θα συνοδεύει αυτά τα δεδομένα ώστε να καταλάβουμε σε τι αναφέρονται θα είναι το όνομα του πεδίου του πίνακα που αντιστοιχούν. Κατά τα άλλα, δεν θα υπάρχει τίποτα που να διαχωρίζει την συγκεκριμένη λίστα από μια άλλη εντελώς διαφορετικού νοηματικού περιεχομένου (για παράδειγμα, η λίστα με το σύνολο των χρηστών του συστήματος). Και αν αυτό δεν αποτελεί πρόβλημα για εργασίες που εμπλέκουν μόνο έναν πίνακα της βάσης και πραγματοποιούν μόνο απλούς χειρισμούς στα δεδομένα που επιστρέφουν τα ερωτήματα της βάσης, δημιουργεί σίγουρα πολυπλοκότητα όταν οι εργασίες γίνονται σύνθετες και αυξάνουν σε αριθμό.

Όταν λοιπόν η επιχειρησιακή λογική αρχίζει να γίνεται σύνθετη, τότε γίνεται όλο και μεγαλύτερη η ανάγκη μιας εννοιολογική αναπαράσταση των δεδομένων σε μορφή που μπορεί να αξιοποιήσει άμεσα ο κώδικας της εφαρμογής.

Στο δικό μας σενάριο, αυτή η μορφή είναι ένα σύνολο κλάσεων που αντιστοιχούν στην Entity-Relationship αναπαράσταση της βάσης δεδομένων. Το σύνολο αυτό καλείται Entity-Relationship Model.

Σε γενικές γραμμές, η επιχειρισιακή λογική επιθυμεί το Entity-Relationship Model να είναι σε θέση:

- Να αναπαριστά όλες τις πληροφορίες που υπάρχουν στους πίνακες της βάσης δεδομένων.
- Εκτός από τις πληροφορίες, να αναπαριστά επίσης τις συσχετίσεις μεταξύ των πινάκων, οι οποίες υπονοούνται από τα foreign κλειδιά των πινάκων.
- Να παρέχει μια ολοκληρωμένη απεικόνιση της βάσης δεδομένων, ούτως ώστε όχι μόνο οι πληροφορίες που είναι αποθηκευμένες στη βάση να μπορούν εύκολα να προσπελαστούν μέσω του μοντέλου, αλλά και όσες αλλαγές γίνονται στα αντικείμενα του μοντέλου να μπορούν να μεταφραστούν άμεσα σε αλλαγές των δεδομένων που είναι αποθηκευμένα στη πράξη.

Η σωστή σχεδίαση και η υλοποίηση ενός τέτοιου μοντέλου αποτελεί πολύπλοκο προγραμματιστικό πρόβλημα. Για τον λόγο αυτό, η βιβλιοθήκη .NET περιλαμβάνει ένα εργαλείο, το ADO.NET Entity Framework, το οποίο πραγματοποιεί αυτόματα για λογαριασμό τού προγραμματιστή αυτήν την εργασία και που αξιοποιήσαμε στην εφαρμογή μας.

Γυρνώντας στο παράδειγμά μας, μπορούμε να χρησιμοποιήσουμε τις

κλάσεις του ADO.NET Entity Framework και να εκτελέσουμε την εργασία ως εξής:

```
var db = new IsotrackEntities();
tblUsers User = db.tblUsers.Where(w => w.fldUserLogin ==
Username).FirstOrDefault();

List<tblE2Resources> UserResources = User.tblE2Resources.ToList();
```

Η μεταβλητή db είναι η Entity Model αναπαράσταση της βάσης.
Η πρώτη αλληλεπίδραση με τη βάση γίνεται με ένα LINQ ερώτημα (χρησιμοποιούμε μια λάμδα συνάρτηση του LINQ για την έκφραση του ερωτήματος. Θα μπορούσαμε να είχαμε χρησιμοποιήσει ισοδύναμα τους αντίστοιχους LINQ τελεστές). Το ερώτημα, σε αντίθεση με πιο πριν που δεν είχαμε το Entity Model, επιστρέφει ένα αντικείμενο της κλάσης tblUsers και όχι απλώς μια λίστα από δεδομένα.
Η δεύτερη αλληλεπίδραση με τη βάση δεν χρησιμοποιεί καθόλου το LINQ, αλλά βασίζεται αποκλειστικά στις ευκολίες που δίνει το Entity Model: το γεγονός πως το πεδίο fldUserID του πίνακα tblE2Resources είναι ένα foreign key στον πίνακα tblUsers επιτρέπει στο EntityModel να μας οδηγεί κατευθείαν από το την μεταβλητή User στις συσχετισμένες εγγραφές του πίνακα tblE2Resources. Ο κώδικας που χρειάζεται να γράψουμε είναι έτσι συντομότερος και πολύ σαφέστερος.

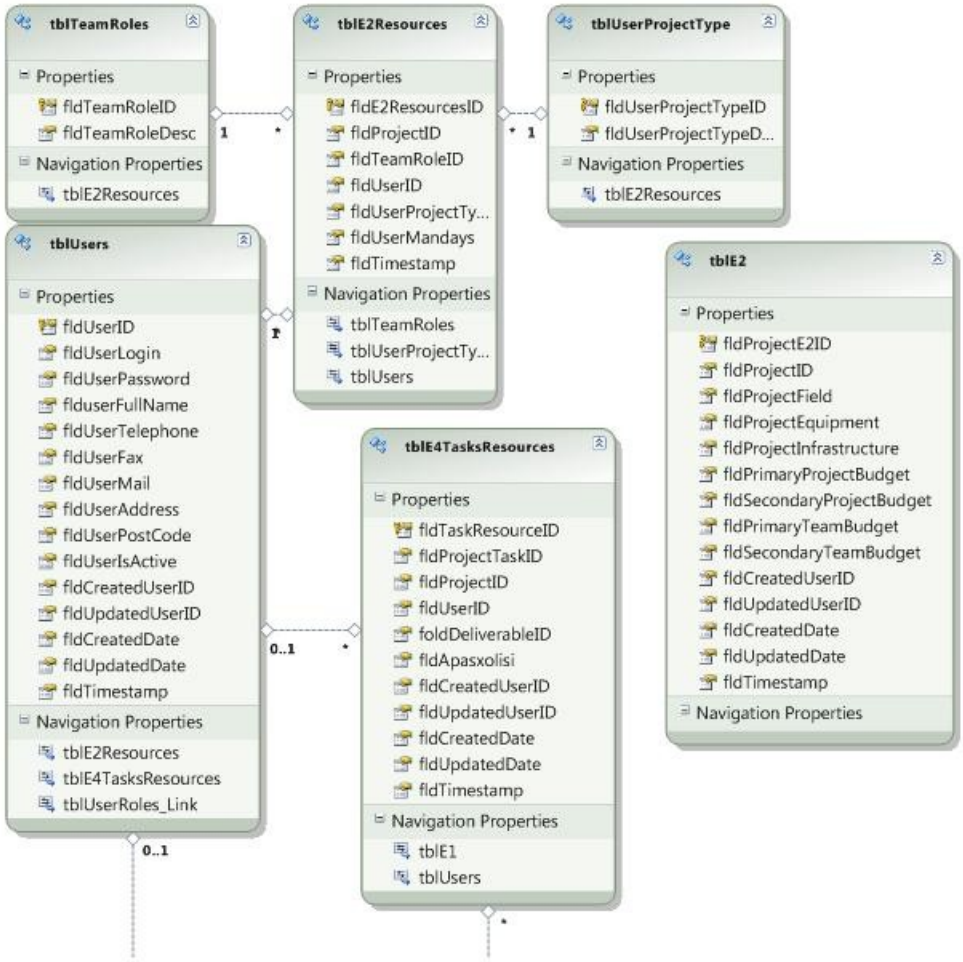
Όπως φάνηκε από το παράδειγμα, για τις περιπτώσεις που οι μέθοδοι που παρέχει το Entity-Relationship Model δεν είναι αρκετά εκφραστικές για τον επιθυμητό χειρισμό των δεδομένων, το .NET προβλέπει την χρήση του LINQ.

Το LINQ δεν είναι απλώς μια βιβλιοθήκη, αλλά είναι ενσωματωμένο στις .NET γλώσσες. Χρησιμοποιείται για την δημιουργία ερωτημάτων προς μια δομή δεδομένων. Τα ερωτήματα αυτά δομούνται σύμφωνα με μια ξεχωριστή γλώσσα που ορίζει το LINQ, η οποία μοιάζει αρκετά στην SQL. Αυτό δεν σημαίνει πως το LINQ περιορίζεται μόνο στην χρήση με συστήματα βάσης δεδομένων· η δομή δεδομένων μπορεί να είναι επίσης ένα XML αρχείο ή να βρίσκεται εξ ολοκλήρου στην κύρια μνήμη. Μερικές πιο εξωτικές υποστηριζόμενες μορφές δεδομένων, όχι πάντως από την Microsoft, είναι άρθρα της Wikipedia (στην περίπτωση αυτή τα LINQ ερωτήματα μεταφράζονται σε HTTP αιτήματα), δεδομένα στο Twitter (τα LINQ ερωτήματα μεταφράζονται σε κλήσεις στο Twitter API) και αποτελέσματα των αναζητήσεων της Google.

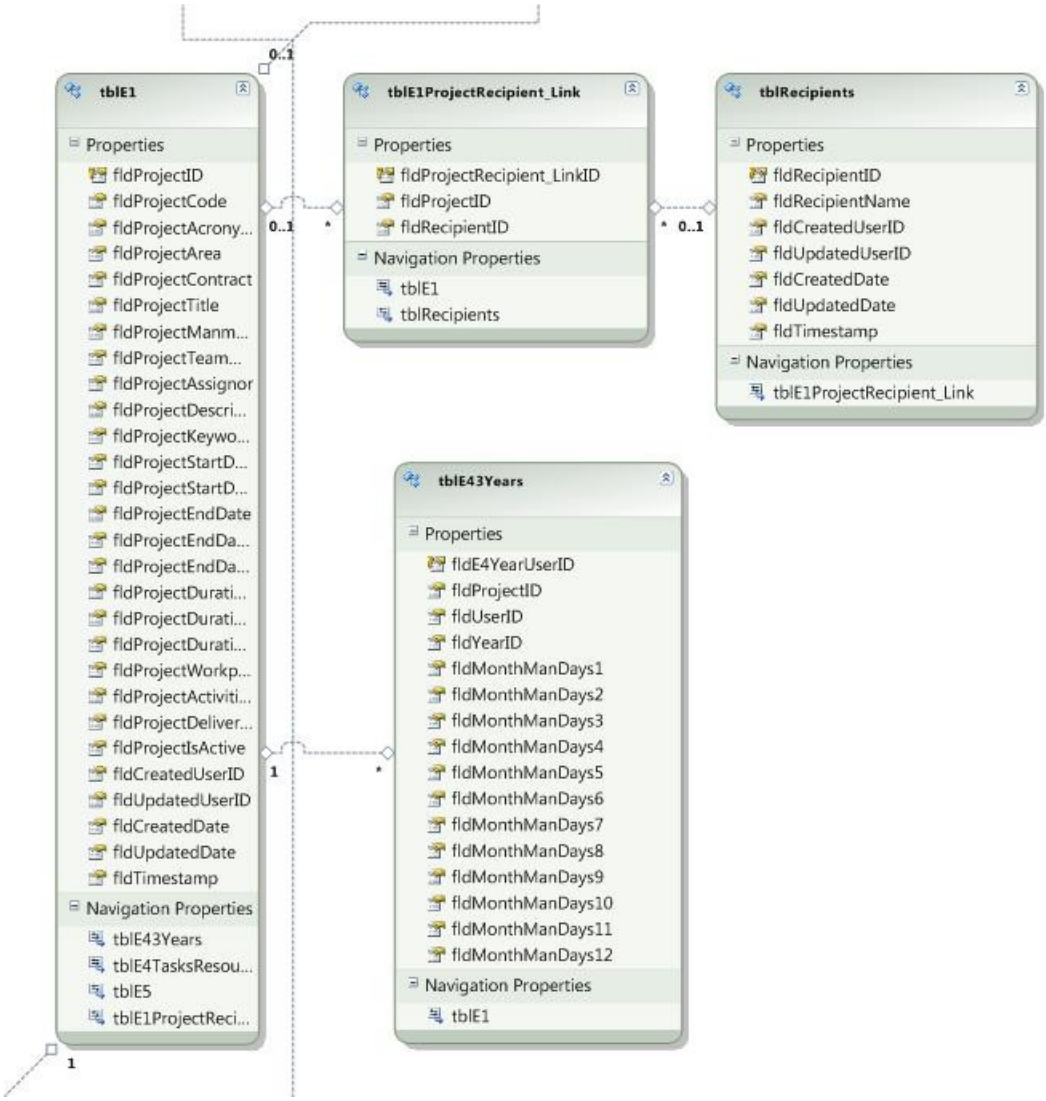
Για εμάς ενδιαφέρον έχει το LINQ To Entities, το οποίο δίνει την δυνατότητα δημιουργίας LINQ ερωτημάτων πάνω από το Entity-Relationship Model.

Η online τεκμηρίωση που δίνει η Microsoft στο msdn.microsoft.com για αυτές τις τεχνολογίες είναι αναλυτική. Για παράδειγμα, μια καλή αρχή για το ADO.NET Entity Framework είναι το άρθρο (12).

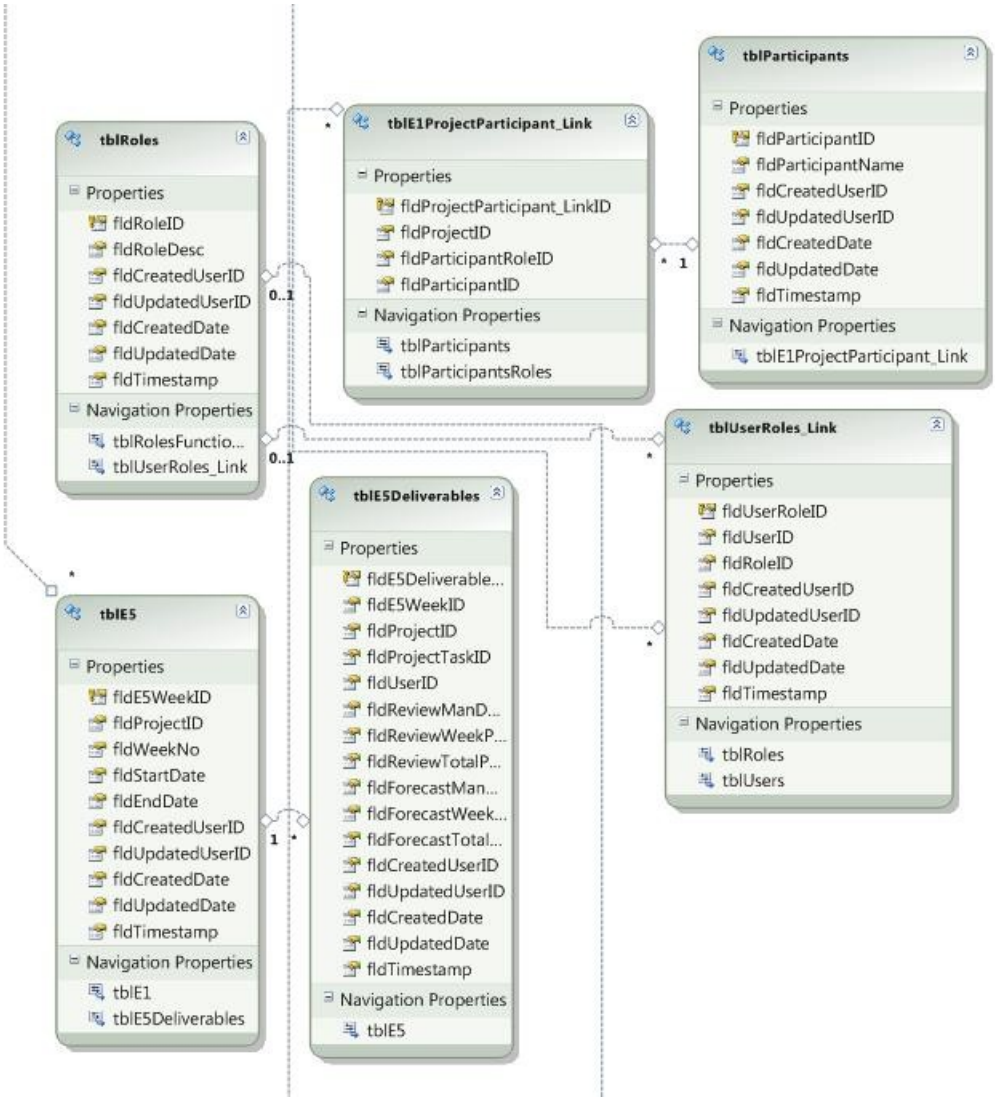
Τα διαγράμματα που ακολουθούν παρουσιάζουν σχηματικά το μοντέλο που δημιούργησε το ADO.NET Entity Framework για την βάση δεδομένων μας. Μπορούμε να παρατηρήσουμε πως το μοντέλο πρόκειται για μια πιστή αναπαράσταση της βάσης που είδαμε στην προηγούμενη ενότητα:



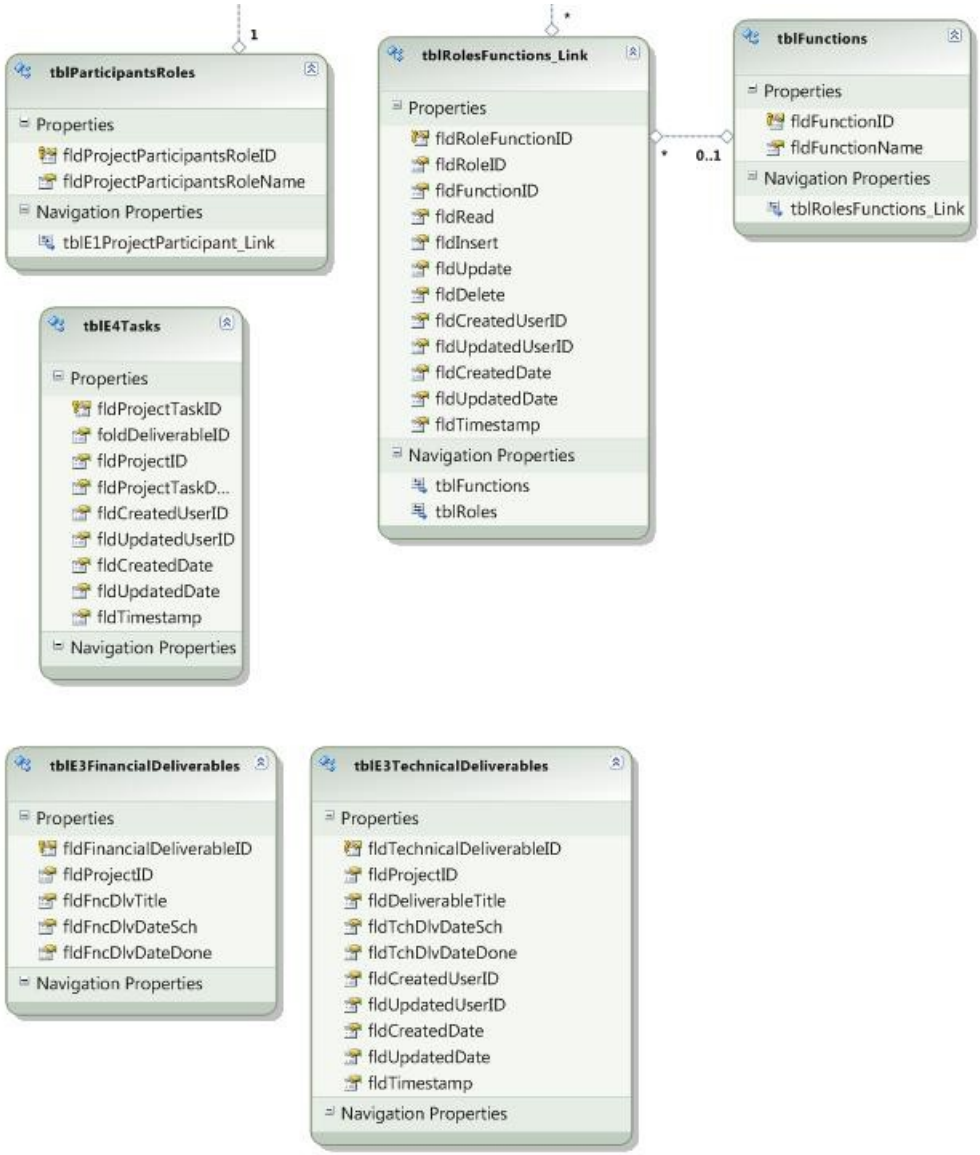

Εικόνα 10


tblE1
Properties
fldProjectID
fldProjectCode
fldProjectAcrony...
fldProjectArea
fldProjectContract
fldProjectTitle
fldProjectManm...
fldProjectTeam...
fldProjectAssignor
fldProjectDescri...
fldProjectKeywo...
fldProjectStartD...
fldProjectStartD...
fldProjectEndDate
fldProjectEndDa...
fldProjectEndDa...
fldProjectDurati...
fldProjectDurati...
fldProjectDurati...
fldProjectWorkp...
fldProjectActiviti...
fldProjectDeliver...
fldProjectIsActive
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblE43Years
tblE4TasksResou...
tblE5
tblE1ProjectReci...
tblE1ProjectRecipient_Link
Properties
fldProjectRecipient_LinkID
fldProjectID
fldRecipientID
Navigation Properties
tblE1
tblRecipients
tblRecipients
Properties
fldRecipientID
fldRecipientName
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblE1ProjectRecipient_Link
tblE43Years
Properties
fldE4YearUserID
fldProjectID
fldUserID
fldYearID
fldMonthManDays1
fldMonthManDays2
fldMonthManDays3
fldMonthManDays4
fldMonthManDays5
fldMonthManDays6
fldMonthManDays7
fldMonthManDays8
fldMonthManDays9
fldMonthManDays10
fldMonthManDays11
fldMonthManDays12
Navigation Properties
tblE1
0..1
*
1

Εικόνα 11


tblRoles
Properties
fldRoleID
fldRoleDesc
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblRolesFunctio...
tblUserRoles_Link
tblE1ProjectParticipant_Link
Properties
fldProjectParticipant_LinkID
fldProjectID
fldParticipantRoleID
fldParticipantID
Navigation Properties
tblParticipants
tblParticipantsRoles
tblParticipants
Properties
fldParticipantID
fldParticipantName
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblE1ProjectParticipant_Link
tblUserRoles_Link
Properties
fldUserRoleID
fldUserID
fldRoleID
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblRoles
tblUsers
tblE5Deliverables
Properties
fldE5Deliverable...
fldE5WeekID
fldProjectID
fldProjectTaskID
fldUserID
fldReviewManD...
fldReviewWeekP...
fldReviewTotalP...
fldForecastMan...
fldForecastWeek...
fldForecastTotal...
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblE5
tblE5
Properties
fldE5WeekID
fldProjectID
fldWeekNo
fldStartDate
fldEndDate
fldCreatedUserID
fldUpdatedUserID
fldCreatedDate
fldUpdatedDate
fldTimestamp
Navigation Properties
tblE1
tblE5Deliverables
0..1
0..1
1
1

Εικόνα 12




Εικόνα 13

### 5.1.4 WCF Υπηρεσίες

Εφαρμογές έξω από τον server μπορούν να ανακτήσουν πληροφορίες από την βάση δεδομένων μόνο μέσω των WCF υπηρεσιών.

Οι υπηρεσίες που παρέχονται είναι αυστηρά προσανατολισμένες στις ανάγκες ενός πελάτη που χρησιμοποιείται για την διαχείριση έργου σύμφωνα με την μέθοδο που περιγράφηκε στο δεύτερο κεφάλαιο (όπως άλλωστε είναι και η βάση δεδομένων). Σε αυτό το πλαίσιο, υπάρχει μια κλάση υπηρεσιών για κάθε κάρτα διαχείρισης. Υπάρχει επιπλέον μια κλάση υπηρεσιών για την ταυτοποίηση των χρηστών.

Η πρόσβαση στις υπηρεσίες ελέγχεται από μηχανισμούς προστασίας του WCF. Λεπτομερειακή περιγραφή του σχετικού μηχανισμού γίνεται στην επόμενη υποενότητα.

Έχουμε ήδη παρατηρήσει πως η βάση δεδομένων είναι οργανωμένη με τέτοιον τρόπο ώστε τα στοιχεία που ενδιαφέρουν την κάθε κάρτα διαχείρισης να βρίσκονται σε συγκεκριμένους πίνακες. Οι WCF υπηρεσίες απλώς θα πρέπει να επιτρέπουν τον χειρισμό των αντίστοιχων πινάκων.

Για παράδειγμα, υπάρχει στην βάση ο πίνακας tblE2, ο οποίος έχει μία εγγραφή για κάθε E2 κάρτα διαχείρισης. Σε αυτόν τον πίνακα αντιστοιχούν δύο WCF υπηρεσίες: Μία που επιστρέφει την εγγραφή που αντιστοιχεί σε ένα συγκεκριμένο έργο, και μία που δέχεται μια ανανεωμένη έκδοση αυτής της εγγραφής, και ενημερώνει με αυτήν την βάση δεδομένων (υπενθυμίζουμε πως η χρήση αυτών των υπηρεσιών δεν επιτρέπεται σε όλους τους χρήστες του συστήματος, μάλιστα η δεύτερη υπηρεσία απαιτεί περισσότερα δικαιώματα από την πρώτη. Περισσότερα στοιχεία για τον μηχανισμό εξουσιοδότησης υπάρχουν στην επόμενη ενότητα).

Όπως παρατηρήσαμε και στην προηγούμενη ενότητα, οι υπηρεσίες αυτές, καθώς και όλες οι υπόλοιπες, επιστρέφουν τις σχετικές πληροφορίες κατευθείαν ως αντικείμενα του Entity Model. Αυτό είναι δυνατό και επιθυμητό· το Entity Model που παράγει το Entity Model Framework είναι serializable, τα σχετικά αντικείμενα μπορούν επομένως να μεταφερθούν μέσω του δικτύου, και αποτελεί μια πολύ εύχρηστη αναπαράσταση της δομής της βάσης δεδομένων.

Στο σημείο αυτό φαίνεται μια συνέπεια της παραδοχής που κάναμε για τις ανάγκες του πελάτη: δεχόμενοι πως ο πελάτης αξιοποιείται για την διαχείριση κάποιου έργου, περιμένουμε πως σε κάθε κλήση θα ζητά πληροφορίες για ένα έργο την φορά. Έτσι, δεν υπάρχει λόγος να σχεδιάσουμε υπηρεσίες που θα επιστρέφουν πάνω από μία εγγραφές του tblE2 την φορά ή θα υποστηρίζουν την ταυτόχρονη (δηλαδή, με μία μόνο κλήση) αλλαγή πολλών εγγραφών.

Από την άλλη, ένα σημαντικό σχετικό ζήτημα ανακύπτει με πίνακες όπως ο tblE2Resources, τον πίνακα που κρατά στοιχεία για τους πόρους του κάθε έργου. Στον tblE2Resources, καθώς και σε πολλούς άλλους πίνακες, είναι δυνατό να αντιστοιχούν πάνω από μία εγγραφές για κάθε έργο. Στην περίπτωση αυτή, είναι πιθανόν ο πελάτης της υπηρεσίας να

επιθυμεί να εργαστεί με πάνω από ένα αντικείμενα την φορά. Γενικά, σε σενάρια όπως αυτό υπάρχουν δύο σχεδιαστικές επιλογές:

- Η πρώτη είναι να φτιάξουμε μια υπηρεσία για κάθε βασική λειτουργία που μπορεί να απαιτηθεί για μια εγγραφή. Αυτές οι βασικές λειτουργίες είναι οι: δημιουργία νέας εγγραφής, ανάκτηση όλων των εγγραφών που μας ενδιαφέρουν, ενημέρωση εγγραφής και διαγραφή εγγραφής. Η δεύτερη από αυτές τις λειτουργίες καλείται να επιστρέψει μια λίστα από εγγραφές, οι άλλες τρεις εργάζονται με μία εγγραφή την φορά. Η μέθοδος αυτή χειρισμού του πίνακα αναφέρεται ως CRUD (Create, Read (Retrieve), Update, Delete (Destroy) ). Για παράδειγμα, για τον πίνακα tblE2Resources, οι σχετικές υπηρεσίες θα ήταν τέσσερις και θα επέτρεπαν την δημιουργία μιας νέας εγγραφής με δοθέντες τιμές στα πεδία, την ανάκτηση όλων των εγγραφών που σχετίζονται με ένα συγκεκριμένο έργο, την ενημέρωση μιας συγκεκριμένης εγγραφής σε μια νέα τιμή, και την διαγραφή μιας συγκεκριμένης εγγραφής.
- Η δεύτερη επιλογή απαιτεί μόνο δύο υπηρεσίες: μία για την ανάκτηση της λίστας των εγγραφών που μας ενδιαφέρουν και άλλη μία που θα δέχεται μια ενημερωμένη έκδοση αυτής της λίστας. Για τον tblE2Resources, η πρώτη υπηρεσία θα είναι ίδια με την υπηρεσία Retrieve της μεθόδου CRUD της πρώτης επιλογής. Η δεύτερη υπηρεσία, θα δέχεται την ανανεωμένη λίστα με τις εγγραφές που αντιστοιχούν σε ένα συγκεκριμένο έργο, και θα ενημερώνει ανάλογα τον tblE2Resources: αν στην ανανεωμένη λίστα βρει μια καινούργια εγγραφή, θα την προσθέτει στην βάση, αν βρει μια ενημερωμένη έκδοση μιας προϋπάρχουσας εγγραφής, θα ενημερώνει την εγγραφή που ήδη υπάρχει, και για όσες προϋπάρχουσες εγγραφές (του σχετικού έργου) δε βρει αντίστοιχη στην ενημερωμένη λίστα, θα τις διαγράφει από την βάση.

Η πρώτη επιλογή είναι κατάλληλη για πελάτες που επιθυμούν να αλληλεπιδρούν με την βάση ανά μία εγγραφή την φορά, για όλες τις εγγραφές που τους ενδιαφέρουν, η δεύτερη για αυτούς που προτιμούν να δίνουν την σχετική ενημερωμένη λίστα μια μόνο φορά.

Από πλευράς αριθμού δεδομένων που διακινούνται στο δίκτυο, φαίνεται η πρώτη προσέγγιση να οδηγεί, στις περισσότερες περιπτώσεις, σε οικονομικότερη υλοποίηση. Πράγματι, αναμένουμε ο τελικός χρήστης που χρησιμοποιεί τον πελάτη να πραγματοποιεί μόνο λίγες τροποποιήσεις στην λίστα που του δίνουμε, κάθε φορά. Είναι λοιπόν σχετικά ασύμφορο να επιστρέφει ολόκληρη την λίστα εγγραφών την

στιγμή που τα περισσότερα στοιχεία της έχουν παραμείνει αμετάβλητα, ιδίως όταν η λίστα αυτή είναι μεγάλη.

Από την άλλη, η δεύτερη προσέγγιση οδηγεί σε απλούστερο σχεδιασμό του πελάτη, μιας και τότε αυτός το μόνο που έχει να κάνει, όταν ο χρήστης κάνει τις αλλαγές που θέλει, είναι να επιστρέψει την λίστα στην σχετική υπηρεσία.

Όπως περιγράφεται στην ενότητα της αρχιτεκτονικής του πελάτη, ο πελάτης σχεδιάστηκε τελικά έτσι ώστε να χρησιμοποιεί τις υπηρεσίες της δεύτερης προσέγγισης. Για τον λόγο αυτό, επιλέξαμε να μην δημιουργήσουμε τις CRUD υπηρεσίες. Κάτι τέτοιο πάντως, θα ήταν αρκετά εύκολο να γίνει σε πιθανή μελλοντική επέκταση της εφαρμογής.

Οι υπηρεσίες δηλώνονται στο WCF ως κλάσεις που είναι συσχετισμένες με την παράμετρο (attribute) ServiceContract. Οι λειτουργίες που μπορεί να επιτελέσει η υπηρεσία δηλώνονται ως μέθοδοι σε μια τέτοια κλάση, και θα πρέπει να είναι συσχετισμένες με την παράμετρο OperationContract. Ακολουθεί μια λίστα με τις δηλώσεις των υπηρεσιών του συστήματος (να σημειώσουμε πως από μόνες τους αυτές οι κλάσεις δεν επαρκούν για την φιλοξενία WCF υπηρεσιών στον IIS. Χρειάζεται επίσης να γίνουν οι κατάλληλες ρυθμίσεις τους WCF μέσα από το Web.config αρχείο της εφαρμογής. Οι βασικότερες από αυτές τις ρυθμίσεις παρουσιάζονται στην επόμενη υποενότητα, την σχετική με την Ασφάλεια):

```
[ServiceContract(Namespace = "")]
public interface IsvE2
{
   [OperationContract]
   public tblE2 GetE2(long ProjectID);

   [OperationContract]
   public tblE2 GetE2ByAcronym(string ProjectAcronym);

   [OperationContract]
   public List<tblUsers> GetAllUsers();

   [OperationContract]
   public List<tblE2Resources> GetE2ResourcesByAcronym(string
ProjectAcronym);

   [OperationContract]
   public List<tblE2Resources> GetE2ResourcesByProjectID(long
ProjectID);
```

```
    [OperationContract]
    public List<tblTeamRoles> GetTeamRoles();

    [OperationContract]
    public List<tblUserProjectType> GetProjectTypes();

    [OperationContract]
    public bool SaveE2(tblE2 ObjE2);

    [OperationContract]
    public bool DeleteAllE2Resources(long ProjectID);

    [OperationContract]
    public bool SaveE2ResourcesList(List<tblE2Resources>
E2ResourcesList);

    [OperationContract]
    public List<tblUsers> GetUsers(long ProjectID);
}

[ServiceContract(Namespace = "")]
public interface IsvE3
{
    [OperationContract]
    public List<tblE3TechnicalDeliverables>
GetTechnicalDeliverables(long ProjectID);

    [OperationContract]
    public List<tblE3FinancialDeliverables>
GetFinancialDeliverables(long ProjectID);

    [OperationContract]
    public bool DeleteAllTechnicalDeliverables(long ProjectID);

    [OperationContract]
    public bool DeleteAllFinancialDeliverables(long ProjectID);

    [OperationContract]
    public bool
SaveTechnicalDeliverablesList(List<tblE3TechnicalDeliverables);

    [OperationContract]
    public bool
SaveFinancialDeliverablesList(List<tblE3FinancialDeliverables);
}

[ServiceContract(Namespace = "")]
public interface IsvE4
```

```
{
   [OperationContract]
   public List<tblE4Tasks> GetE4TasksByProjectID(long ProjectID);

   [OperationContract]
   public List<tblE4Tasks> GetE4TasksResourcesByTaskID(long
ProjectTaskID);

   [OperationContract]
   public bool DeleteAllE4Tasks(long ProjectID);

   [OperationContract]
   public bool DeleteAllE4TasksResources(long ProjectTaskID);

   [OperationContract]
   public bool SaveE4TasksList(List<tblE4Tasks> Tasks);

   [OperationContract]
   public bool SaveE4TasksResourcesList(List<tblE4Tasks>
Resources);
}

[ServiceContract(Namespace = "")]
public interface IsvE43
{
   [OperationContract]
   public List<tblE43Years> GetE43Years(long ProjectID);

   [OperationContract]
   public bool DeleteAllE43Years(long ProjectID);

   [OperationContract]
   public bool SaveE43YearsList(List<tblE43Years> Years);
}

[ServiceContract(Namespace = "")]
public interface IsvE5
{
   [OperationContract]
   public List<tblE5> GetE5ListByProjectID(long ProjectID);

   [OperationContract]
   public List<tblE5Deliverables>
GetE5DeliverablesListByWeekID(long WeekID);

   [OperationContract]
   public tblE5 GetSuggestedWeekRecord(long ProjectID);

   [OperationContract]
```

```
    public bool DeleteAllWeeks(long ProjectID);

    [OperationContract]
    public bool DeleteAllDeliverables(long WeekID);

    [OperationContract]
    public bool SaveE5List(List<tblE5> Weeks);

    [OperationContract]
    public bool SaveDeliverablesList(List<tblE5Deliverables
Deliverables);
}


[ServiceContract(Namespace = "")]
public interface IsvProj
{
    [OperationContract]
    public List<tblE1> GetProjectList();

    [OperationContract]
    public List<string> GetProjectListAcronyms();

    [OperationContract]
    public GetProjectByAcronym(string ProjectAcronym);

    [OperationContract]
    public AddProject(string Code, string Acronym, string Title);

    [OperationContract]
    public bool AddParticipantsLink(long ProjectID, string
ParticipantName, string ParticipantRoleName);

    [OperationContract]
    public bool AddRecipientLink(long ProjectID, string
RecipientName)

    [OperationContract]
    public List<string> GetParticipantRoleNames();

    [OperationContract]
    public List<string> GetParticipantNames();

    [OperationContract]
    public List<tblE1ProjectParticipant_Link>
GetParticipantsLinks(long ProjectID);

    [OperationContract]
    public List<string> GetRecipientNames();
```

```
    [OperationContract]
    public List<tblE1ProjectRecipient_Link> GetRecipientsLinks(long
ProjectID);

    [OperationContract]
    public bool DeleteParticipantsLink(long ParticipationLinkID);

    [OperationContract]
    public bool DeleteRecipientLink(long RecipientLinkID);
}


[ServiceContract(Namespace = "")]
public interface IsvUsers
{
    [OperationContract]
    public List<tblUsers> GetAllUsers();

    [OperationContract]
    public List<tblUsers> GetAllUsersWithRoles();

    [OperationContract]
    public tblUsers UserInfo();

    [OperationContract]
    public bool UpdateUser(tblUsers UpdatedUser);

    [OperationContract]
    public bool SaveUsersList(List<tblUsers> Users);
}
```

Μπορούμε να παρατηρήσουμε, σε σχέση με την ανάλυση που κάναμε πιο πριν για τις υπηρεσίες που παρέχονται για τις λίστες στοιχείων, πως εκτός από την υπηρεσία λήψης λίστας και την υπηρεσία αποθήκευσης λίστας υπάρχει και η υπηρεσία διαγραφής όλων των στοιχείων της λίστας.

Διατηρώντας το ίδιο παράδειγμα, πέρα από τις GetE2ResourcesByProjectID(long ProjectID) και SaveE2ResourcesList(List<tblE2Resources> E2ResourcesList) υπάρχει και η υπηρεσία DeleteAllE2Resources(long ProjectID).

Αυτή η τρίτη υπηρεσία είναι απαραίτητη για το ενδεχόμενο η ανανεωμένη λίστα να είναι κενή, ο χρήστης δηλαδή έχει διαγράψει όλες τις εγγραφές σε αυτήν. Τότε, δεν υπάρχει τρόπος η SaveE2ResourcesList να εξαγάγει το ProjectID στο οποίο αναφέρεται ο πελάτης της (αφού το

όρισμά της θα είναι μια κενή λίστα). Σε αυτήν την περίπτωση, ο πελάτης θα πρέπει να καλέσει την DeleteAllE2Resources.

Οι διάφορες υπηρεσίες αποθήκευσης λίστας, σαν την SaveE2ResourcesList, καλούνται να επιτελέσουν σύνθετες εργασίες: πρέπει να βρουν ποια από τα στοιχεία της νέας λίστας αντιστοιχούν σε ανανεωμένες εγγραφές και ποιες από τις εγγραφές που υπάρχουν στην βάση δεν υπάρχουν στη νέα λίστα, και πρέπει να διαγραφούν.

Από την άλλη, όλες αυτές οι υπηρεσίες επιτελούν την ίδια λειτουργία, άρα έχει νόημα να υλοποιήσουμε μια μόνο φορά την σχετική λειτουργικότητα και να χρησιμοποιήσουμε την υλοποίηση σε όλες τις υπηρεσίες που την χρειάζονται.

Προτού παρουσιάσουμε την υλοποίηση σε C#, θα κάνουμε μια λεκτική περιγραφή του αλγόριθμου που χρησιμοποιήσαμε. Ο συγκεκριμένος αλγόριθμος είναι αρκετά σύνθετος, επειδή προσπαθεί να ελαχιστοποιήσει τις λειτουργίες που εκτελεί η βάση δεδομένων, οι οποίες είναι και οι πλέον χρονοβόρες (αν δε μας ενδιέφερε η ελαχιστοποίηση των λειτουργιών της βάσης, μια πολύ απλούστερη προσέγγιση θα ήταν η εξής: διαγράφουμε πρώτα όλες τις εγγραφές του πίνακα που αντιστοιχούν στο συγκεκριμένο έργο, και μετά δημιουργούμε στον πίνακα μια νέα εγγραφή για κάθε στοιχείο της ανανεωμένης λίστας).

Ο αλγόριθμος παίρνει τα εξής ορίσματα:

1) Ένα αντικείμενο db του Entity Model της βάσης δεδομένων.
2) Έναν iterator OldListEnum στην Entity Model λίστα των παλιών εγγραφών (ως λίστα παλιών εγγραφές εννοούμε τις εγγραφές του σχετικού πίνακα που αντιστοιχούν στο συγκεκριμένο έργο).
3) Έναν iterator NewListEnum στην Entity Model λίστα των ενημερωμένων στοιχείων (ως λίστα ενημερωμένων στοιχείων εννοούμε τη λίστα που δίνει ο πελάτης στην υπηρεσία).
4) Ένα αντικείμενο Compare το οποίο δείχνει σε μια συνάρτηση που μπορεί να συγκρίνει δύο Entity Model στοιχεία του σχετικού πίνακα με βάση το κλειδί τους και να επιστρέψει τιμή αρνητική, μηδέν ή θετική, ανάλογα με το αν το πρώτο στοιχείο προηγείται, είναι το ίδιο, ή έπεται του δεύτερου.
5) Ένα αντικείμενο Save το οποίο δείχνει σε μια συνάρτηση που μπορεί να αποθηκεύσει σε μια Entity Model αναπαράσταση της βάσης δεδομένων ένα στοιχείο του σχετικού πίνακα.

6) Ένα αντικείμενο Update το οποίο δέχεται δύο Entity Model αναπαραστάσεις στοιχείων του πίνακα και ενημερώνει το πρώτο στοιχείο με βάση το δεύτερο
7) Ένα αντικείμενο Delete το οποίο μπορεί να διαγράψει από μια Entity Model αναπαράσταση της βάσης δεδομένων ένα δοθέν στοιχείο.

Οι συναρτήσεις που δείχνουν τα αντικείμενα Save και Update θα πρέπει να είναι σε θέση να αναγνωρίζουν πότε ένα στοιχείο της νέας λίστας έχει όντως διαφορετικές τιμές από την αντίστοιχη παλιά εγγραφή, και να μην ζητούν καμιά λειτουργία από την βάση στην περίπτωση που δεν έχει γίνει καμιά αλλαγή. Για να μπορούν να το κάνουν αυτό γρήγορα, χωρίς δηλαδή να χρειάζεται να ελέγχουν ένα-ένα τα πεδία του ενημερωμένου στοιχείου και να τα συγκρίνουν με αυτά του αντίστοιχου παλιού, έχουμε θέσει τον εξής κανόνα για τις λίστες των ενημερωμένων στοιχείων:

*Ένα στοιχείο της λίστας που δίνει ο πελάτης σε μια υπηρεσία ενημέρωσης εγγραφών πίνακα, αντιστοιχεί σε ανανεωμένη έκδοση εγγραφής της βάσης μόνο αν το πεδίο fldTimestamp του στοιχείου είναι ίσο με μηδέν.*

Με άλλα λόγια, αν ο χρήστης αλλάξει κάποιο πεδίο ενός στοιχείου μιας λίστας, ο πελάτης θα πρέπει να θέσει το πεδίο fldTimestamp του στοιχείου ίσο με μηδέν, αν θέλει η υπηρεσία στην οποία μετά θα δώσει την λίστα να ενημερώσει την παλιά εγγραφή που αντιστοιχεί στο στοιχείο.

Ο αλγόριθμος εκτελεί τα παρακάτω βήματα:

1) Μετακινεί τον OldListEnum στο πρώτο στοιχείο της παλιάς λίστας, που είναι μεγαλύτερο ή ίσο με το πρώτο στοιχείο της ενημερωμένης λίστας. Όσα παλιά στοιχεία διατρέχει μέχρι να φτάσει σε αυτό, τα διαγράφει από τη βάση.
2) Μετακινεί τον NewListEnum στο στοιχείο της νέας λίστας που είναι ίσο με το πρώτο στοιχείο της παλιάς λίστας, ή στο τελευταίο εάν αυτό δεν υπάρχει. Όσο ενημερωμένα στοιχεία διατρέχει μέχρι να φτάσει σε αυτό, έχουν κλειδί που είναι μικρότερο από το κλειδί κάθε εγγραφής του πίνακα στη βάση. Αυτό, στην Entity Model αναπαράσταση, σημαίνει πως πρόκειται για στοιχεία με κλειδί μηδέν, δηλαδή για νέα στοιχεία, και επομένως τα αποθηκεύει στην Entity Model αναπαράσταση της βάσης.
3) Στο σημείο αυτό ο NewListEnum και ο OldListEnum δείχνουν σε ίσα στοιχεία. Ενημερώνει το δεύτερο με βάση το πρώτο.
4) Προχωράει στο επόμενο στοιχείο της νέας λίστας, εάν υπάρχει. Προχωράει τον OldListEnum μέχρι να βρει το ίσο του. Όσα παλιά

στοιχεία διατρέχει μέχρι να φτάσει σε αυτό, τα διαγράφει από τη βάση. Ξαναπηγαίνει στο βήμα 3.

Σε C# κώδικα:

```
        public delegate long CompareDlgt(EntityObject X, EntityObject
Y);
       public delegate bool SaveItemDlgt(IsotrackEntities db, tblUsers
User, EntityObject Item);
            public delegate bool UpdateItemDlgt(IsotrackEntities db,
tblUsers User, EntityObject OutdatedItem, EntityObject
UpdatedItem);
             public delegate void DeleteItemDlgt(IsotrackEntities db,
EntityObject Item);

       public static bool SaveList(
          IsotrackEntities db,
          tblUsers User,
          IEnumerator<EntityObject> OldListEnum,
          IEnumerator<EntityObject> NewListEnum,
          CompareDlgt Compare,
          SaveItemDlgt Save,
          UpdateItemDlgt Update,
          DeleteItemDlgt Delete
          )
       {
             // Initialize objects OldItem, NewItem, to point to the first
items of the old list
          // and new list respectively
          EntityObject OldItem, NewItem;
          if ((OldListEnum != null) && OldListEnum.MoveNext())
             OldItem = OldListEnum.Current;
          else
             OldItem = null;
          NewListEnum.MoveNext();
          NewItem = NewListEnum.Current;

            // Delete those items from the old list that are before the
first item of the new list
           while ((OldItem != null) && (Compare(OldItem, NewItem) <
0))
          {
             Delete(db, OldItem);
             if (OldListEnum.MoveNext())
                OldItem = OldListEnum.Current;
             else
                OldItem = null;
          }
```

```
            // Saves the items from the new list that are before the first
item of the old list
            // (these items are the ones with id zero, thus newly created
items, not updated ones)
                  while ((NewItem != null) && ((OldItem == null) ||
(Compare(OldItem, NewItem) > 0)))
            {
               if (!Save(db, User, NewItem))
                  return false;
               if (NewListEnum.MoveNext())
                  NewItem = NewListEnum.Current;
               else
                  NewItem = null;
            }

            while (NewItem != null)
            {
               // Compare(OldItem, NewItem) has to be zero here
               Update(db, User, OldItem, NewItem);
               if (NewListEnum.MoveNext())
               {
                  NewItem = NewListEnum.Current;
                  OldListEnum.MoveNext();
                  OldItem = OldListEnum.Current;
                  while (Compare(OldItem, NewItem) != 0)
                  {
                     Delete(db, OldItem);
                     OldListEnum.MoveNext();
                     OldItem = OldListEnum.Current;
                  }
               }
               else
                  NewItem = null;
            }
            return true;
         }
```

### 5.1.5 Ασφάλεια

Υπενθυμίζουμε τις απαιτήσεις ασφαλείας από το σύστημα:

- Οι χρήστες θα πρέπει να συνδέονται δίνοντας username και password,
- Η επικοινωνία μεταξύ πελάτη εξυπηρετητή θα πρέπει να πρέπει να αντιμετωπίζει αποτελεσματικά απόπειρες υποκλοπής,

- Οι συνδεδεμένοι χρήστες θα πρέπει να περιορίζονται στις λειτουργίες που είναι εξουσιοδοτημένοι να κάνουν
- Η βάση δεδομένων δεν θα πρέπει να είναι ορατή από τους πελάτες, ο μόνος τρόπος με τον οποίον θα μπορούν να αντλούν δεδομένα από αυτήν θα πρέπει να είναι μέσω των υπηρεσιών.

Η τέταρτη απαίτηση είναι η ευκολότερη να ικανοποιηθεί, αρκεί να ρυθμίσουμε έτσι την βάση στον Sql Server έτσι ώστε να κάνει δεκτές συνδέσεις μόνο από τις WCF υπηρεσίες.

Για την πρώτη απαίτηση, θα πρέπει να βρούμε έναν τρόπο να δίνει ο πελάτης στον εξυπηρετητή το username και το password του. Τα στοιχεία αυτά καλούνται credentials (διαπιστευτήρια). Η ικανοποίηση της δεύτερης απαίτησης μας οδηγεί στην χρήση κρυπτογραφίας στο πρωτόκολλο επικοινωνίας.

Και τα δύο αυτά χαρακτηριστικά της επικοινωνίας, κρυπτογραφία και μεταφορά των username-password credentials, μπορούν να διαμορφωθούν κάνοντας τις κατάλληλες ρυθμίσεις στις παραμέτρους του WCF. Ο τρόπος που γίνεται αυτό παρουσιάζεται στην επόμενη υποενότητα.

#### 5.1.5.1 Ταυτοποίηση Χρήστη

Θα παραθέσουμε αποσπάσματα από το Web.config αρχείο του εξυπηρετητή, που παραμετροποιούν το WCF ώστε να ταυτοποιεί τους πελάτες που προσπαθούν να συνδεθούν, και κατ' επέκταση τους τελικούς χρήστες. Τα αποσπάσματα αυτά θα συμπληρώσουν την εικόνα για τον τρόπο που δηλώνονται οι υπηρεσίες στο WCF.

Όλα τα αποσπάσματα που ακολουθούν βρίσκονται μέσα στο στο στοιχείο <configuration><system.ServiceModel> του Web.config.

Η πρώτη σειρά ρυθμίσεων δημιουργούν μια διαμόρφωση του BasicHttpBinding του WCF (το συγκεκριμένο binding χρησιμοποιεί το HTTP για την μεταφορά SOAP 1.1 μηνυμάτων. Δημιουργεί endpoints για μια υπηρεσία που υπακούν στις οδηγίες του WS-I BP 1.1). Την διαμόρφωση αυτή την ονομάζει secEnabledBinding. Η secEnabledBinding παραμετροποιεί το BasicHttpBinding ώστε

1) να χρησιμοποιεί HTTPS,

2) να χρησιμοποιεί τόσο ασφάλεια επιπέδου μηνύματος όσο και ασφάλειας επιπέδου πρωτοκόλλου μεταφοράς,
3) να απαιτεί από τον πελάτη να δίνει σε κάθε μήνυμα το username και το password του, ώστε να μπορεί να αυθεντικοποιηθεί,
4) να επιτρέπει μεγαλύτερα όρια για το μέγεθος των μηνυμάτων που μεταφέρονται μεταξύ πελάτη και υπηρεσίας (αυτή είναι ρύθμιση άσχετη με την ασφάλεια).

Το σχετικό απόσπασμα είναι το:

```
<bindings>

  <basicHttpBinding>
    <binding name="secEnabledBinding"
          maxReceivedMessageSize="50000000"
          maxBufferSize="50000000">
      <security mode="TransportWithMessageCredential">
        <message  clientCredentialType="UserName" />
      </security>
    </binding>
  </basicHttpBinding>

</bindings>
```

Η διαμόρφωση αυτή χρησιμοποιείται στην δήλωση των υπηρεσιών ως εξής:

```
<services>
  <service name="Isotrack3.Web.svProj">

    <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svProj" />
    <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
  </service>
  <service name="Isotrack3.Web.svE2">
    <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svE2" />
    <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
  </service>
  <service name="Isotrack3.Web.svE3">
```

```
      <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svE3" />
      <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
    </service>
    <service name="Isotrack3.Web.svE4">
      <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svE4" />
      <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
    </service>
    <service name="Isotrack3.Web.svE5">
      <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svE5" />
      <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
    </service>
    <service name="Isotrack3.Web.svSecurity">
      <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svSecurity" />
      <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
    </service>
    <service name="Isotrack3.Web.svUsers">
      <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svUsers" />
      <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
    </service>
    <service name="Isotrack3.Web.svE43">
      <endpoint address="" binding="basicHttpBinding"
bindingConfiguration="secEnabledBinding"
contract="Isotrack3.Web.svE43" />
      <endpoint address="mex" binding="mexHttpsBinding"
contract="IMetadataExchange" />
    </service>
  </services>
```

Για κάθε υπηρεσία, ορίζουμε λοιπόν το binding που θα χρησιμοποιεί για την μεταφορά δεδομένων να είναι το basicHttpBinding, διαμορφωμένο κατά την secEnabledHttpBinding.

Το contract που ορίζεται στις υπηρεσίες είναι απλώς τα public interface που είδαμε στην προηγούμενη ενότητα. Η διαφορά είναι πως εκεί τα είχαμε παρουσιάσει σε πιο καθαρή μορφή, ως public interfaces με ονόματα όπως IsvE2. Στον πραγματικό κώδικα, ακολουθώντας μια όχι τόσο καθαρή προσέγγιση, φτιάξαμε κατευθείαν κλάσεις, με ονόματα όπως svE2, και εμπιστευτήκαμε το WCF για να φτιάξει αυτόματα αντίστοιχα public interfaces. Γι' αυτό και στο παραπάνω απόσπασμα το contract και το όνομα της υπηρεσίας είναι ακριβώς το ίδιο.

Πέρα από το binding για την μεταφορά των δεδομένων, χρειάζεται και ένα binding για την μεταφορά των μεταδεδομένων της υπηρεσίας, πληροφορίες σχετικές με τις λειτουργίες που παρέχει, τις απαιτήσεις της, κτλ. Για αυτές χρησιμοποιούμε το mexHttpsBinding, το οποίο προσφέρει κρυπτογραφημένη επικοινωνία, χωρίς όμως να απαιτεί κάποιο username ή password από τον πελάτη.

Τέλος, πρέπει να ορίσουμε τον τρόπο με τον οποίον θα γίνεται ο έλεγχος του username και του password που δίνει ο πελάτης. Το WCF παρέχει αρκετές μεθόδους για να γίνει αυτό· εμείς θέλουμε να χρησιμοποιήσουμε δική μας ξεχωριστή μέθοδο, μιας και τα user credentials είναι αποθηκευμένα ήδη στην βάση δεδομένων της εφαρμογής:

```
   <behaviors>
     <serviceBehaviors>

       <behavior name="">
         <serviceCredentials>
           <userNameAuthentication
userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="Isotrack3.Web.CustomUs
erNamePasswordValidator, Isotrack3.Web" />
         </serviceCredentials>
         <dataContractSerializer maxItemsInObjectGraph="50000"/>
         <serviceMetadata httpGetEnabled="true" />
         <serviceDebug includeExceptionDetailInFaults="true" />
       </behavior>
     </serviceBehaviors>
   </behaviors>
```

Έτσι, οδηγούμε την λογική για την επικύρωση των credential του χρήστη στη δική μας υλοποίηση της κλάσης UserNameValidator:

```
   public class CustomUserNamePasswordValidator :
UserNamePasswordValidator
   {
```

```
        public override void Validate(string userName, string password)
        {
            IsotrackEntities db = new IsotrackEntities();
            byte[] PasswordHashBytes =
System.Security.Cryptography.MD5.Create().ComputeHash((new
System.Text.UTF8Encoding()).GetBytes(password));
            string PasswordHash = (new
System.Text.UTF8Encoding()).GetString(PasswordHashBytes);
            return;
            if (db.tblUsers.Where(w => (w.fldUserLogin == userName)
&& (w.fldUserPassword == PasswordHash)).FirstOrDefault() ==
null)
                throw new
System.IdentityModel.Tokens.SecurityTokenException("wrong
username/password combination");
        }
    }
```

Όταν ένας πελάτης θα προσπαθήσει να καλέσει μια υπηρεσία, το username και το password που θα δώσει θα δοθούν σαν ορίσματα στην παραπάνω συνάρτηση Validate. Αν αυτή δεν εγείρει κάποια εξαίρεση, το WCF θεωρεί πως ο πελάτης ταυτοποιήθηκε με επιτυχία.

Η Validate απλώς ελέγχει αν στον πίνακα tblUsers της βάσης δεδομένων υπάρχει εγγραφή με πεδίο fldUsername ίσο με το δοθέν Username, και πεδίο fldPassword ίσο με το MD5 Hash του δοθέντος Password.

Η απόφαση να αποθηκεύονται στην βάση τα hashes των κωδικών και όχι οι ίδιοι οι κωδικοί θεωρείται μια σωστή πρακτική ασφαλείας: ακόμη και αν κάποιος μπορέσει να διαβάσει τις εγγραφές στον tblUsers πίνακα, θα του είναι πολύ δύσκολο να εξαγάγει από τα hashes τους κωδικούς των χρηστών.

### *5.1.5.2 Εξουσιοδότηση Χρήστη*

Το σύστημα απαιτεί από την φύση του την υλοποίηση πολιτικών εξουσιοδότησης, αφού πρέπει να υποστηρίζει τύπους χρηστών με διαφορετικά δικαιώματα.

Αν και το WCF, μέσω άλλων βιβλιοθηκών του .NET, προσφέρει λύσεις για την σχεδίαση συστήματος εξουσιοδότησης, προτιμήσαμε να υλοποίησουμε ένα δικό μας σχήμα, επειδή κρίναμε πως κάτι τέτοιο είναι απλούστερο από το να ασχοληθούμε με τις λεπτομέρειες των σχετικών βιβλιοθηκών του .NET.

Καταρχήν, υλοποιήσαμε ένα σύνολο συναρτήσεων που ενημερώνουν για τους ρόλους ενός χρήστη, με βάση το ID του:

```
   public class UserRoles
   {
      public enum Role { Administrator = 4, ProjectCreator = 3, User
= 2, Visitor = 1, NoRole = 0 };
      public enum TeamRole { Director = 3, Manager = 2, Member =
1, NoTeamRole = 0 };

      public static Role RoleInDb(IsotrackEntities db, long UserID)
      {
         bool PrevState = db.ContextOptions.LazyLoadingEnabled;
         db.ContextOptions.LazyLoadingEnabled = true;
         long? DbRole =
                  db.tblUserRoles_Link.Where(w => w.fldUserID ==
UserID).Select(url => url.tblRoles.fldRoleID).FirstOrDefault();
         db.ContextOptions.LazyLoadingEnabled = PrevState;

         if (DbRole == null)
            return Role.NoRole;

         switch ((long) DbRole)
         {
            case (1):
               return Role.Administrator;
            case (2):
               return Role.ProjectCreator;
            case (3):
               return Role.User;
            case (4):
               return Role.Visitor;
            default:
               return Role.NoRole;
         }
      }

       public static TeamRole TeamRoleInProject(IsotrackEntities db,
long UserID, long ProjectID)
      {
         bool PrevState = db.ContextOptions.LazyLoadingEnabled;
         db.ContextOptions.LazyLoadingEnabled = true;
                  var E2Resource = db.tblE2Resources.Where(w =>
(w.fldProjectID == ProjectID) && (w.fldUserID ==
UserID)).FirstOrDefault();
         db.ContextOptions.LazyLoadingEnabled = PrevState;

         if (E2Resource == null)
            return TeamRole.NoTeamRole;
         int DbTeamRole = E2Resource.fldTeamRoleID;
         switch (DbTeamRole)
```

```
            {
                case (1):
                    return TeamRole.Director;
                case (2):
                    return TeamRole.Manager;
                case (3):
                    return TeamRole.Member;
                default:
                    return TeamRole.NoTeamRole;
            }
        }

         public static bool CanSeeAllUsers(IsotrackEntities db, long
UserID)
        {
            bool PrevState = db.ContextOptions.LazyLoadingEnabled;
            db.ContextOptions.LazyLoadingEnabled = true;

            Role DbRole = RoleInDb(db, UserID);
            if ((DbRole == Role.Administrator) ||
                (DbRole == Role.ProjectCreator))
            {
                db.ContextOptions.LazyLoadingEnabled = PrevState;
                return true;
            }

            var Test = db.tblE2Resources.Where(w =>
                        (w.fldUserID == UserID) && (w.fldTeamRoleID !=
3)).FirstOrDefault();

            db.ContextOptions.LazyLoadingEnabled = PrevState;
            if (Test == null)
                return false;
            else
                return true;
        }

        public static bool DirectorAccessInProject(IsotrackEntities db,
long UserID, long ProjectID)
        {
            Role DbRole = RoleInDb(db, UserID);
            if (DbRole == Role.Administrator)
                return true;
            TeamRole ProjectTeamRole = TeamRoleInProject(db, UserID,
ProjectID);
            if (ProjectTeamRole == TeamRole.Director)
                return true;
            if (DbRole == Role.ProjectCreator)
            {
```

```
            tblE1 Project = db.tblE1.Where(w => w.fldProjectID ==
ProjectID).FirstOrDefault();
            if (Project == null)
                return false;
            if (Project.fldCreatedUserID == UserID)
                return true;
        }
        return false;
    }

     public static bool ManagerAccessInProject(IsotrackEntities db,
long UserID, long ProjectID)
    {
        Role DbRole = RoleInDb(db, UserID);
        if (DbRole == Role.Administrator)
            return true;
        TeamRole ProjectTeamRole = TeamRoleInProject(db, UserID,
ProjectID);
                    if ((ProjectTeamRole == TeamRole.Director) ||
(ProjectTeamRole == TeamRole.Manager))
            return true;
        if (DbRole == Role.ProjectCreator)
        {
            tblE1 Project = db.tblE1.Where(w => w.fldProjectID ==
ProjectID).FirstOrDefault();
            if (Project == null)
                return false;
            if (Project.fldCreatedUserID == UserID)
                return true;
        }
        return false;
    }

     public static bool TeamMemberAccessInProject(IsotrackEntities
db, long UserID, long ProjectID)
    {
        Role DbRole = RoleInDb(db, UserID);
        if (DbRole == Role.Administrator)
            return true;
        TeamRole ProjectTeamRole = TeamRoleInProject(db, UserID,
ProjectID);
        if ((ProjectTeamRole == TeamRole.Director) ||
            (ProjectTeamRole == TeamRole.Manager) ||
            (ProjectTeamRole == TeamRole.Member))
            return true;
        if (DbRole == Role.ProjectCreator)
        {
            tblE1 Project = db.tblE1.Where(w => w.fldProjectID ==
ProjectID).FirstOrDefault();
```

```
            if (Project == null)
                return false;
            if (Project.fldCreatedUserID == UserID)
                return true;
        }
        return false;
    }
}
```

Με αυτές τις συναρτήσεις, οι υπηρεσίες μπορούν να μάθουν τον ρόλο ενός χρήστη συνολικά στο σύστημα (Administrator, ProjectCreator, User, Visitor) και συγκεκριμένα για κάποιο δεδομένο έργο (Director, Manager, Member, No Role In Project).

Η εξουσιοδότηση των χρηστών γίνεται λοιπόν στην υλοποίηση των υπηρεσιών. Αν ο χρήστης δεν έχει αρκετά δικαιώματα για την εκτέλεση της λειτουργίας που ζήτησε, η υπηρεσία του επιστρέφει μια τιμή που υποδεικνύει την παρουσία σφάλματος (NULL ή false).

Σε κάποιες περιπτώσεις, το αποτέλεσμα που θα επιστρέψει η υπηρεσία είναι διαφορετικό, ανάλογα με τους ρόλους τους πελάτη. Αυτό συμβαίνει στην λειτουργία GetProjectListAcronym της υπηρεσία svProj. Η λειτουργία αυτή επιστρέφει τα ακρωνύμια των έργων στα οποία ο χρήστης είναι μέλος, εκτός και αν αυτός έχει δικαιώματα Administrator, οπότε του επιστρέφει τα ακρωνύμια όλων των έργων στη βάση:

```
/// <summary>
///
/// </summary>
/// <returns>
/// the acronyms of the projects  that the user is a member of
/// (if the user has administrator rights, the full list of projects)
/// </returns>
[OperationContract]
public List<string> GetProjectListAcronym()
{
    try
    {
        IsotrackEntities db = new IsotrackEntities();

        string Username =
ServiceSecurityContext.Current.PrimaryIdentity.Name;
        tblUsers User = db.tblUsers.Where(w => w.fldUserLogin
== Username).FirstOrDefault();

        db.ContextOptions.LazyLoadingEnabled = false;
```

```
            if (UserRoles.RoleInDb(db, User.fldUserID) ==
UserRoles.Role.Administrator)
                return (from pr in db.tblE1 select
pr.fldProjectAcronym).ToList();
            else
                return (
                    from pr in db.tblE1
                    join e2r in db.tblE2Resources
                    on pr.fldProjectID equals e2r.fldProjectID
                    where e2r.fldUserID == User.fldUserID
                    select pr.fldProjectAcronym
                    ).ToList();
        }
        catch (Exception)
        {
            return null;
        }
    }
```

Να παρατηρήσουμε πως ο πελάτης δεν χρειάζεται να δίνει σαν όρισμα το Username και το Password του· αυτά, όπως ήδη περιγράψαμε, μεταφέρονται με τα μεταδεδομένα της επικοινωνίας, και είναι προσπελάσιμα από την υπηρεσία από τις κλάσεις που παρέχει το WCF.

Τέλος, ο πελάτης μπορεί να μάθει τους ρόλους του στο σύστημα καλώντας την υπηρεσία svSecurity:

```
   [ServiceContract(Namespace = "")]
   [AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
   public class svSecurity
   {
      [OperationContract]
      public int RoleID()
      {
         try
         {
            IsotrackEntities db = new IsotrackEntities();

            string Username =
ServiceSecurityContext.Current.PrimaryIdentity.Name;
            tblUsers User = db.tblUsers.Where(w => w.fldUserLogin
== Username).FirstOrDefault();

            var Role = db.tblUserRoles_Link.Where(w => w.fldUserID
== User.fldUserID).FirstOrDefault();
            if ((Role == null) || (Role.fldRoleID == null))
               return 0;
```

```
            else
                return (int) Role.fldRoleID;
        }
        catch (Exception)
        {
            return 0;
        }
    }


    [OperationContract]
    public bool DirectorAccessInProject(long ProjectID)
    {
        try
        {
            IsotrackEntities db = new IsotrackEntities();

            string Username =
ServiceSecurityContext.Current.PrimaryIdentity.Name;
            tblUsers User = db.tblUsers.Where(w => w.fldUserLogin
== Username).FirstOrDefault();

            return (UserRoles.DirectorAccessInProject(db,
User.fldUserID, ProjectID));
        }
        catch (Exception)
        {
            return false;
        }
    }

    [OperationContract]
    public bool ManagerAccessInProject(long ProjectID)
    {
        try
        {
            IsotrackEntities db = new IsotrackEntities();

            string Username =
ServiceSecurityContext.Current.PrimaryIdentity.Name;
            tblUsers User = db.tblUsers.Where(w => w.fldUserLogin
== Username).FirstOrDefault();

            return (UserRoles.ManagerAccessInProject(db,
User.fldUserID, ProjectID));
        }
        catch (Exception)
        {
            return false;
```

```
        }
    }

    [OperationContract]
    public bool TeamMemberAccessInProject(long ProjectID)
    {
        try
        {
            IsotrackEntities db = new IsotrackEntities();

            string Username =
ServiceSecurityContext.Current.PrimaryIdentity.Name;
            tblUsers User = db.tblUsers.Where(w => w.fldUserLogin
== Username).FirstOrDefault();

            return (UserRoles.TeamMemberAccessInProject(db,
User.fldUserID, ProjectID));
        }
        catch (Exception)
        {
            return false;
        }
    }
}
```

## 5.2 Η Αρχιτεκτονική Του Πελάτη

### 5.2.1 Το Model-View-ViewModel Αρχιτεκτονικό Μοτίβο

Αυτό το αρχιτεκτονικό μοτίβο έχει εισηχθεί από την Microsoft, ως μια από τις τεχνικές που συνιστά για την ανάπτυξη Silverlight και WPF εφαρμογών. Πρόκειται για μια εξειδίκευση του Μοτίβου Παρουσίασης του Martin Fowler (13), ειδικά για την περίπτωση που χρησιμοποιείται η XAML για την παρουσίαση (View) της εφαρμογής. Αναφέρεται συντομογραφικά και ως MVVM.

Ένας από τους κύριους σκοπούς της χρήσης αυτού του μοτίβου είναι η όσον το δυνατόν πληρέστερη ανεξαρτητοποίηση της ανάπτυξης της παρουσίασης από τα υπόλοιπα τμήματα της εφαρμογής. Αυτό επιτρέπει στους προγραμματιστές που είναι εξειδικευμένοι στην κατασκευή του UI να επικεντρωθούν στην εργασία που τους ενδιαφέρει, δίχως να χρειάζεται να λάβουν υπόψιν τους τις λεπτομέρειες τις εταιρικής λογικής ή κώδικα άλλον από αυτόν της XAML. Άλλες ευεργετικές συνέπειες από

την υιοθέτηση του μοτίβου θα φανούν κατά την αναλυτική περιγραφή του που θα κάνουμε στην συνέχεια.

Το μοτίβο περιγράφει την συμπεριφορά των εξής τριών οντοτήτων:

- Του Μοντέλου Δεδομένων (Model). Πρόκειται για την κλασικό στοιχείο όλων των σχετικών μοτίβων, είναι δηλαδή το ομώνυμο στοιχείο του Model-View-Controller μοτίβου. Αναφέρεται στην runtime αναπαράσταση των πραγματικών δεδομένων που διαχειρίζεται η εφαρμογή. Στην περίπτωση μας, δεν πρόκειται βέβαια για την αρχική εκδοχή των δεδομένων, αφού αυτά βρίσκονται αποθηκευμένα κεντρικά στην βάση δεδομένων του εξυπηρετητή, αλλά σε ένα τοπικό αντίγραφό τους. Το μοντέλο δεν πρέπει να ασχολείται καθόλου ούτε με το πώς θα αναπαρασταθούν αυτά τα δεδομένα.
- Της Παρουσίασης (View). Πάλι σε αντιστοιχία με το Model-View-Controller μοτίβο, πρόκειται για το ομώνυμο στοιχείο του Model-View-Controller. Αναφέρεται στην τελική, οπτική αναπαράσταση των δεδομένων. Αναφέρεται δηλαδή στα στοιχεία που βλέπει ο τελικός χρήστης (lists, datagrids, κτλ).
- Του Μοντέλου Της Παρουσίασης (ModelView). Πρόκειται για έννοια δανεισμένη από το Μοτίβο Παρουσίασης (Presentation Pattern). Αναφέρεται σε μια αφηρημένη αναπαράσταση αυτού που εμφανίζει το στοιχείο View. Ενώ δηλαδή το Μοντέλο Δεδομένων κρατά μια «καθαρή» αναπαράσταση των δεδομένων, το Μοντέλο Της Παρουσίασης αναλαμβάνει να μετατρέψει την προηγούμενη σε κάτι που μπορεί να εμφανιστεί άμεσα στην διεπαφή από το στοιχείο Παρουσίασης.

Αυτές οι οντότητες από μόνες τους, δεν αρκούν για απλοποιήσουν αρκετά την σχεδίαση της διεπαφής της εφαρμογής, ούτε περιγράφουν πλήρως το MVVM. Το επιπλέον βασικό στοιχείο που χαρακτηρίζει το μοτίβο είναι η χρήση μιας εξειδικευμένης markup γλώσσας για την κωδικοποίηση της Παρουσίασης και της περιγραφής της σύζευξής της με το Μοντέλο Της Παρουσίασης. Για την δική μας εφαρμογή, η γλώσσα αυτή είναι η XAML.

Μερικά από τα πλεονεκτήματα που προσφέρει η χρήση της XAML αναφέρθηκαν στο προηγούμενο κεφάλαιο. Εδώ θα δούμε πως αξιοποιείται στο MVVM για την απλοποίησης του κώδικα της διεπαφής,

χρησιμοποιώντας ένα κατάλληλο παράδειγμα (το οποίο προέρχεται από την εξαιρετική ομιλία (14)). Δεν χρησιμοποιήσαμε παράδειγμα από την εφαρμογή μας γιατί τότε θα έπρεπε να παραθέσουμε πολλές γραμμές κώδικα και επειδή το παράδειγμα που θα παραθέσουμε εφαρμόζει το MVVM μοτίβο με πιο καθαρό τρόπο.

Ας δούμε πρώτα το View επίπεδο.

Στο Silverlight, αυτό είναι πάντα μια κλάση UserControl, ή μια κλάση που την κληρονομεί. Το ενδιαφέρον είναι πως δεν χρειάζεται να γράψουμε καθόλου C# για την υλοποίηση της κλάσης, όλη η λειτουργικότητα περιγράφεται από την XAML:

```
<UserControl.Resources>
   <DataTemplate x:Key="CustomerTemplate">
      <StackPanel>
         <TextBlock Text="{Binding FirstName}"
                    Foreground="White"
                    FontSize="18" FontFamily="Ravie" />
         <TextBlock Text="{Binding LastName}"
                    Foreground="White"
                    FontSize="36" FontFamily="Ravie" />
      </StackPanel>
   </DataTemplate>

   <vm:MainViewModel x:Key="mvm" />
</UserControl.Resources>

<Grid x:Name="LayoutRoot"
```

```
            DataContext="{Binding Source={StaticResource
mvm}}">

    <ListBox ItemsSource="{Binding Path=Customers}"
                                ItemTemplate="{StaticResource
CustomerTemplate}"
            Background="Blue" />
</Grid>
```

Με τον κώδικα αυτόν απεικονίζεται μια λίστα (ListBox), η οποία βρίσκεται μέσα σε ένα Grid. Τα στοιχεία της λίστας θα ληφθούν από την μεταβλητή (property) Customers του αντικειμένου mvm. Το mvm ορίζεται στα Resources της σελίδας ως εξής: <vm:MainViewModel x:Key="mvm" /> . Αυτό σημαίνει πως όταν θα εκτελεστεί ο κώδικας XAML από το Silverlight, θα έχει ήδη δημιουργηθεί ένα αντικείμενο της κλάσης MainViewModel, και θα είναι στη διάθεση του κώδικα της σελίδας με το όνομα mvm.

Τα στοιχεία της λίστας Customers θα απεικονιστούν με τον τρόπο που περιγράφει το DataTemplate με όνομα CustomerTemplate. Αυτό απλώς ορίζει πως πρέπει να απεικονιστεί πρώτο το πεδίο FirstName του κάθε στοιχείου της λίστας, με κάποιο συγκεκριμένο font, και μετά το πεδίο LastName, με κάποιο άλλο font.

Περιγράψαμε πρώτα το View Επίπεδο, γιατί έτσι φαίνεται πιο εύκολα τις απαιτήσεις που αυτό έχει από το ViewModel Επίπεδο: Η ViewModel κλάση θα πρέπει να έχει ένα πεδίο με όνομα Customers, το οποίο θα πρέπει να είναι μια λίστα αντικειμένων που έχουν στα πεδία τους δύο συμβολοσειρές, τις FirstName και LastName. Αυτός είναι γενικά ο βασικός ρόλος του ViewModel στο MVVM: να παρέχει την πιο κατάλληλη δυνατή μορφή των δεδομένων, για την απεικόνισή τους από το View.

Η ViewModel κλάση είναι η εξής:

```
public class MainViewModel : INotifyPropertyChanged
```

```
    {
        public const string CustomersPropertyName =
"Customers";

        private ObservableCollection<Customer> _customers;

        public ObservableCollection<Customer> Customers
        {
            get
            {
                return _customers;
            }

            set
            {
                if (_customers == value)
                {
                    return;
                }

                _customers = value;
                RaisePropertyChanged(CustomersPropertyName);
            }
        }
```

```
        public MainViewModel()
        {
            if (DesignerProperties.IsInDesignTool)
            {
                Customers = new
ObservableCollection<Customer>();

                for (var index = 0; index < 15; index++)
                {
                    var customer = new Customer
                    {
                        FirstName = "FirstName" + index,
                        LastName = "LastName" + index
                    };

                    Customers.Add(customer);
                }
            }
            else
            {
                var service = new DataService();
                service.GetCustomers((customers, error) =>
                {
                    if (error != null)
                    {
```

```
                    MessageBox.Show(error.Message);
                }
                else
                {
                    Customers = new
ObservableCollection<Customer>(customers);
                }
            });
        }
    }

    public event PropertyChangedEventHandler
PropertyChanged;

    public void RaisePropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new
PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Παρατηρούμε πως η MainViewModel υλοποιεί το interface INotifyPropertyChanged. Αυτό σημαίνει πως όποτε αλλάζει η τιμή κάποιου πεδίου της MainViewModel το οποίο πιθανώς ενδιαφέρει το

View, η MainViewModel δεσμεύεται πως θα εγείρει το γεγονός PropertyChanged, ώστε να ενημερώσει το View για την αλλαγή.

Τα πραγματικά δεδομένα, τα οποία θα χρησιμοποιηθούν για την δημιουργία των Customers, προέρχονται από την κλάση DataService. Η τελευταία αποτελεί μέρος του Model επιπέδου.

Επίσης, στον συγκεκριμένο κώδικα βλέπουμε πως είναι δυνατόν να μάθουμε σε runtime χρόνο αν ο κώδικας εκτελείται μέσα σε ένα περιβάλλον σχεδίασης, ελέγχοντας το Boolean πεδίο DesignerProperties.IsInDesignTool. Εάν συμβαίνει κάτι τέτοιο, τότε δημιουργήσουμε μια τυχαία λίστα από Customers χωρίς να εμπλέξουμε το Model επίπεδο (το Model επίπεδο επικοινωνεί, όπως θα δούμε σε λίγο, με κάποιον server για την λήψη των Customers. Τα τρέχοντα εργαλεία σχεδίασης δεν έχουν την δυνατότητα να καλούν web services. Επομένως, αν δεν φροντίσουμε να δημιουργήσουμε κατευθείαν κάποιους Customers το πρόγραμμα σχεδίασης δεν θα εμφανίσει καθόλου την λίστα, ή μπορεί και να εγείρει exception).

Φαίνεται τώρα και ένα άλλος βασικό πλεονέκτημα του MVVM: βοηθά πολύ την δημιουργία unit tests για το interface της εφαρμογής. Μιας και είναι δύσκολο να ελεγχθεί κατευθείαν η πραγματική απεικόνιση με unit tests, μπορούμε εναλλακτικά να φτιάξουμε unit tests για το ViewModel επίπεδο, το οποίο περιέχει ακριβώς ό,τι θα εμφανιστεί μέσω του View.

Το Model επίπεδο είναι η κλάση DataService:

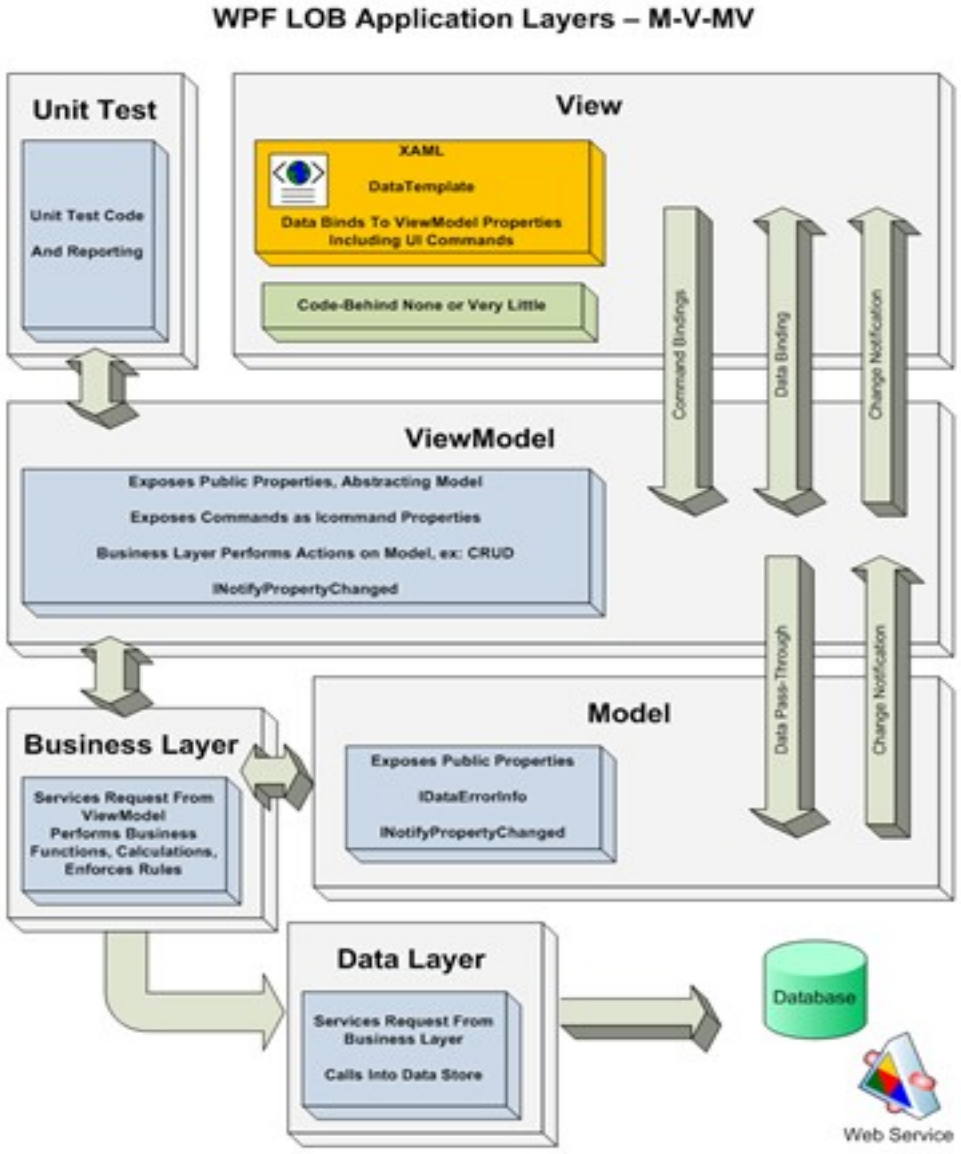
```
public class DataService

{

   public void GetCustomers(

      Action<IEnumerable<Customer>, Exception>
callback)

   {

      var client = new
CustomerServiceClient("BasicEndPoint");



      client.GetAllCustomersCompleted += (s, e) =>
```

```
            {
                var userCallback = e.UserState
                    as Action<IEnumerable<Customer>, Exception>;

                if (userCallback == null)
                {
                    return;
                }

                if (e.Error != null)
                {
                    userCallback(null, e.Error);
                    return;
                }

                userCallback(e.Result, null);
            };

            client.GetAllCustomersAsync(callback);
        }
    }
```

Όποτε το MainViewModel χρειάζεται δεδομένα, καλεί την GetCustomers. Αυτή καλεί τότε με την σειρά της την κατάλληλη web service. Όταν η web service επιστρέψει κάποιο αποτέλεσμα, ή όταν τελειώσει η προθεσμία για την λήψη αποτελέσματος, ενημερώνεται το MainViewModel, χρησιμοποιώντας την συνάρτηση callback. Η callback έχει δοθεί σαν όρισμα στην GetCustomers, ώστε να κληθεί με όρισμα

είτε τους Customers που έδωσε το web service, είτε μια περιγραφή σφάλματος, αν κάτι πήγε στραβά στην επικοινωνία.

Το σχήμα στην Εικόνα 14 περιγράφει σχηματικά όσα είπαμε για το Model-View-ViewModel.




Εικόνα 14

## 5.2.2 Navigation Σελίδων Και Ιστορικό Σελίδων

Παραδοσιακά, ήταν δύσκολη η πλοήγηση με ταυτόχρονη χρήση του ιστορικού ανάμεσα στις σελίδες μιας RIA εφαρμογής. Αυτό είναι

μειονέκτημα εγγενές στην αρχιτεκτονική των RIA εφαρμογών: το περιεχόμενο της σελίδας δεν είναι πια αποθηκευμένο εξ' ολοκλήρου στον server και επομένως η αποθήκευση του URL από τον browser δεν αρκεί για την πλοήγηση σε προηγούμενες σελίδες.

Στην εφαρμογή μας, οι κάρτες διαχείρισης απεικονίζονται στον browser του τελικού χρήστη ως Silverlight σελίδες. Το ενδιαφέρον είναι πως το Silverlight μπορεί να παρεμβαίνει στην σχετική με το Navigation λειτουργικότητα των browser που υποστηρίζει, έτσι ώστε το ιστορικό πλοήγησης για την Silverlight εφαρμογή να συμπεριφέρεται σαν να πρόκειται για κλασική web σελίδα (15).

Τις σχετικές δυνατότητες προσφέρει η κλάση Frame του Silverlight. Πρόκειται για ένα από τα Controls του Silverlight. Ένα αντικείμενο Frame μπορεί να βρίσκεται οπουδήποτε στην βασική σελίδα της εφαρμογής. Αν τώρα, για τις σελίδες που μεταβαίνει ο χρήστης, ορίζουμε σαν parent control αυτό το Frame και ορίσουμε ένα URI που θα αντιστοιχεί στην εν λόγω σελίδα, τότε ο χρήστης θα μπορεί να πλοηγηθεί στις προηγούμενες σελίδες που επισκέφτηκε χρησιμοποιώντας απλά τα κουμπιά πλοήγησης «Μπρος» και «Πίσω» του browser.

Μιας και επιθυμούμε το πρόγραμμα Silverlight να λειτουργεί και εξίσου ομαλά και εκτός browser, έχουμε προσθέσει στο βασικό μενού του προγράμματος δύο κουμπιά που λειτουργούν ακριβώς ίδια με τα κουμπιά πλοήγησης του browser, τα οποία μπορούν να χρησιμοποιηθούν όταν αυτός δεν είναι παρών.

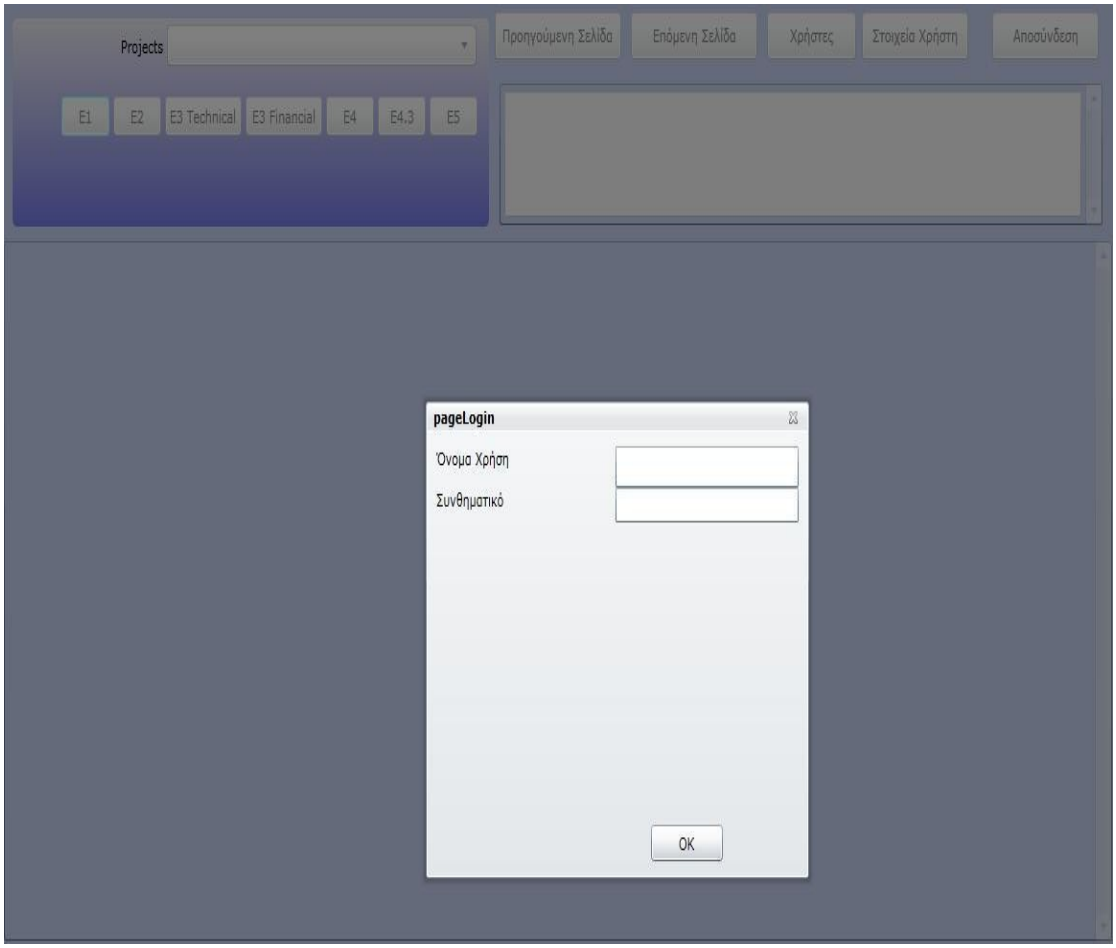
# 6. Παρουσίαση

Στο κεφάλαιο αυτό παρουσιάζουμε εικόνες (screenshots) από τις βασικές λειτουργίες του πελάτη.

## 6.1 Login

Στην παρακάτω εικόνα εμφανίζεται η πρώτη οθόνη που αντικρίζει ο χρήστης όταν φορτώνεται η Silverlight εφαρμογή, η οποία του ζητά να εισάγει το username και τον κωδικό του.

Αν και ο χρήστης μπορεί να δει το κεντρικό μενού, στην πραγματικότητα στο σημείο αυτό ο μπορεί να αλληλεπιδράσει μόνο με την φόρμα login. Μόλις πατήσει το κουμπί OK, η Silverlight εφαρμογή ελέγχει την εγκυρότητα των στοιχείων που δόθηκαν, επικοινωνώντας με το κατάλληλο web service. Αν τα στοιχεία δεν είναι σωστά, τότε το πρόγραμμα επανεμφανίζει την φόρμα login, αλλιώτικα ενεργοποιεί το κεντρικό μενού.




Εικόνα 15

## 6.2 Κεντρικό Μενού

Το κεντρικό μενού αποτελείται από μία αναδιπλούμενη λίστα (drop-down list) για την επιλογή του έργου, έναν χώρο μέσα στον οποίο

εμφανίζονται ενημερωτικά μηνύματα για την επιτυχή ή αποτυχή έκβαση των εργασιών που ζητά ο χρήστης από το πρόγραμμα και κουμπιά για την πλοήγηση στις διάφορες σελίδες διαχείρισης: εφτά κουμπιά για τις κάρτες διαχείρισης της μεθοδολογίας και άλλα δύο για την διαχείριση των χρηστών και την διαχείριση των προσωπικών στοιχείων. Επίσης, υπάρχει επιλογή αποσύνδεσης, η οποία οδηγεί στην φόρμα login, και κουμπιά για πλοήγηση στο ιστορικών των σελίδων που έχει επισκεφθεί ο χρήστης (τα κουμπιά αυτά είναι ακριβώς ίδιας λειτουργικότητας με τα αντίστοιχα του browser μέσα στον οποίον τρέχει το Silverlight. Βρίσκονται για τις περιπτώσεις που το πρόγραμμα εκτελείται εκτός browser).

Η λίστα με τα έργα εμφανίζει τα ακρωνύμια μόνο των έργων στα οποία ο συνδεδεμένος χρήστης είναι μέλος, εκτός αν αυτός έχει δικαιώματα διαχειριστή, οπότε εμφανίζει τα ακρωνύμια όλων των έργων.

Οι σελίδες διαχείρισης προσωπικών στοιχείων και αποσύνδεσης είναι διαθέσιμες σε όλους τους χρήστες. Τα κουμπιά για τις υπόλοιπες λειτουργίες, είναι ενεργοποιημένα κατά περίπτωση, σύμφωνα με τους ρόλους του χρήστη στο επιλεγμένο έργο:

- Οι κάρτες E4, E43, E5 είναι διαθέσιμες σε όλα τα μέλη του έργου.
- Οι κάρτες E1, E2, E3 Technical και E3 Financial μόνο στους Διευθυντές και Υπευθύνους έργου.

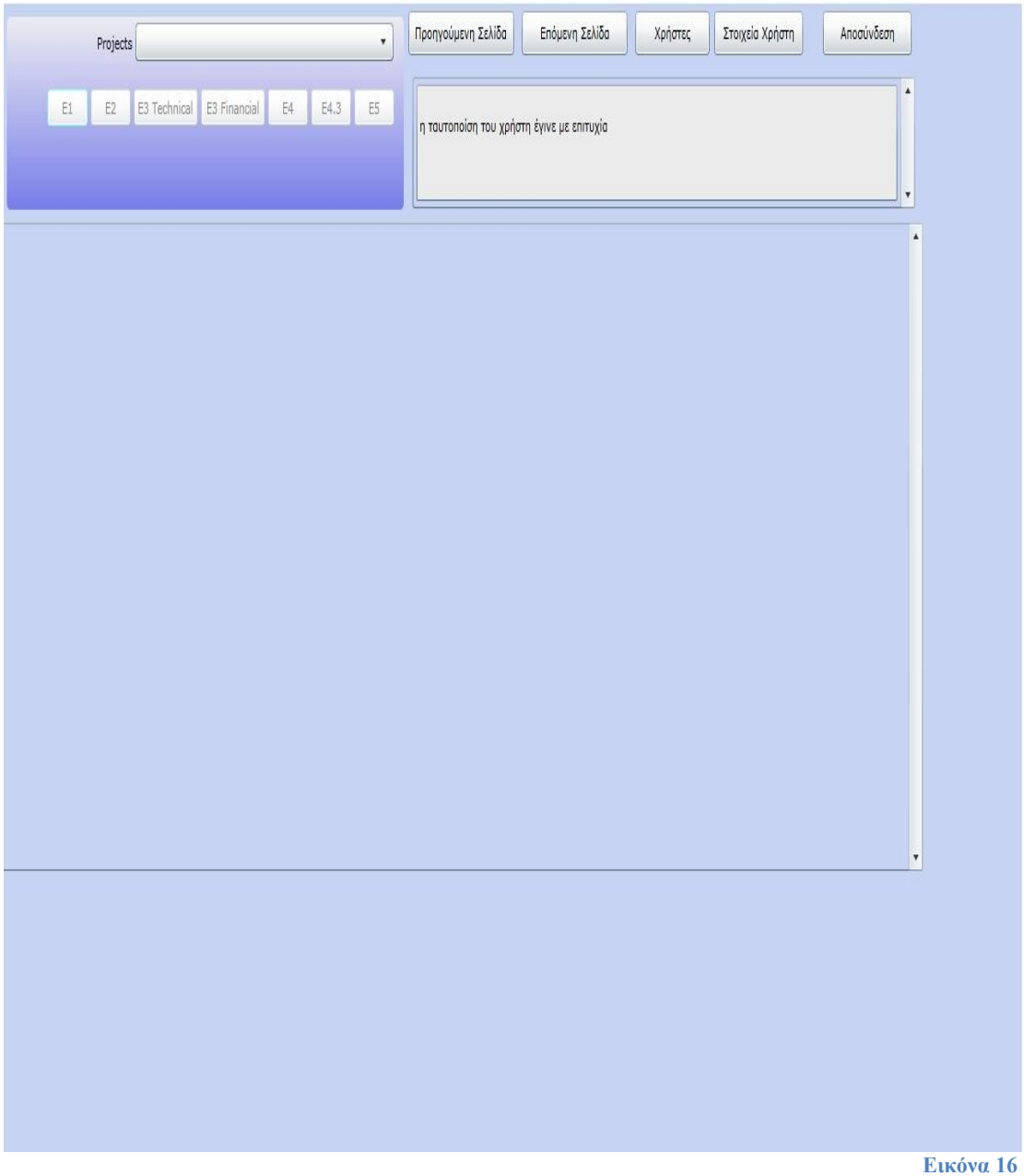
Το κεντρικό μενού είναι πάντα προσβάσιμο στον χρήστη, ανεξάρτητα της σελίδας στην οποία βρίσκεται. Αυτό σημαίνει πως αν για παράδειγμα ο χρήστης εργάζεται στην κάρτα διαχείρισης E1 κάποιου έργου και χρησιμοποιήσει την λίστα των ακρωνύμια των έργων για να αλλάξει έργο, τότε αυτόματα θα μεταβεί στην κάρτα διαχείρισης E1 του καινούργιου έργου (αν έχει βέβαια αρκετά δικαιώματα για να την προσπελάσει).

Για τον παραπάνω λόγο, στις σελίδες διαχείρισης έργου, δεν έχουμε επαναλάβει τα πεδία ορισμού του έργου στο οποίο αναφέρονται. Το σχετικό έργο φαίνεται στην λίστα του κεντρικού μενού, και δεν υπάρχει λόγος να εμφανίζονται οι ίδιες πληροφορίες στις σελίδες διαχείρισης.

Όλες οι σελίδες στις οποίες πλοηγείται ο χρήστης εμφανίζονται κάτω από το κεντρικό μενού, σε μια περιοχή προκαθορισμένου μεγέθους. Αν η σελίδα είναι πολύ μεγάλη για να χωρέσει εκεί, εμφανίζεται ένα μενού κύλισης ειδικά για αυτήν την περιοχή.

Στο ακόλουθο screenshot εμφανίζεται η οθόνη που βλέπει ένας χρήστης, όταν έχει μόλις συνδεθεί. Μιας και δεν έχει επιλέξει ακόμη έργο, τα

κουμπιά διαχείρισης είναι απενεργοποιημένα (εκτός από το κουμπί για τα Στοιχεία Χρήστη. Πρόκειται επομένως για διαχειριστή).




Εικόνα 16

## 6.3 Σελίδα E1

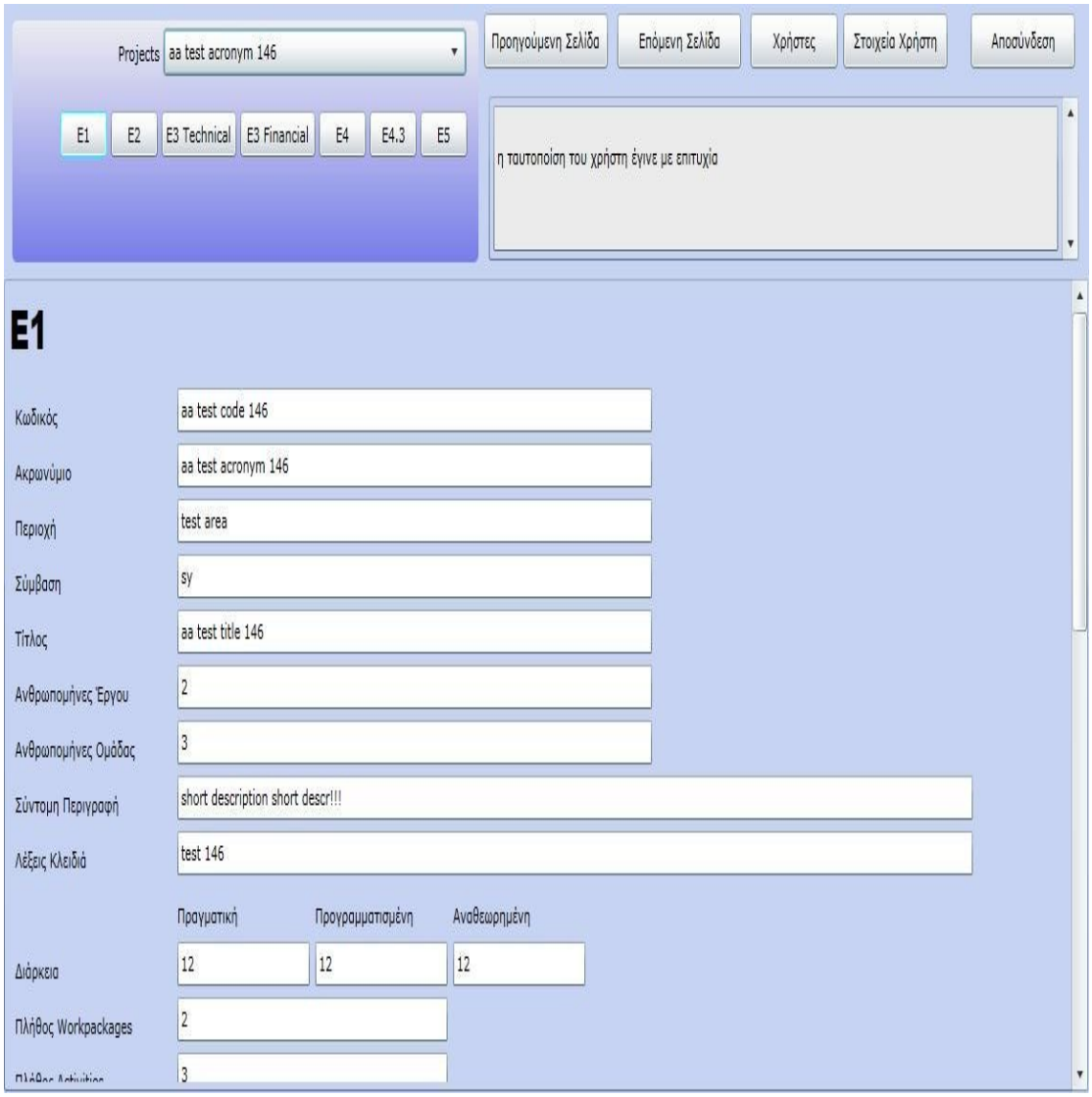
Στις επόμενες δύο εικόνες φαίνονται δύο στιγμιότυπα από την κάρτα διαχείρισης E1.

Η σελίδα αυτή, υπενθυμίζουμε, είναι προσπελάσιμη τόσο στους Project Managers του έργου όσο και στους Project Directors. Αλλαγές στα στοιχεία όμως, μπορούν να κάνουν μόνο οι Directors.

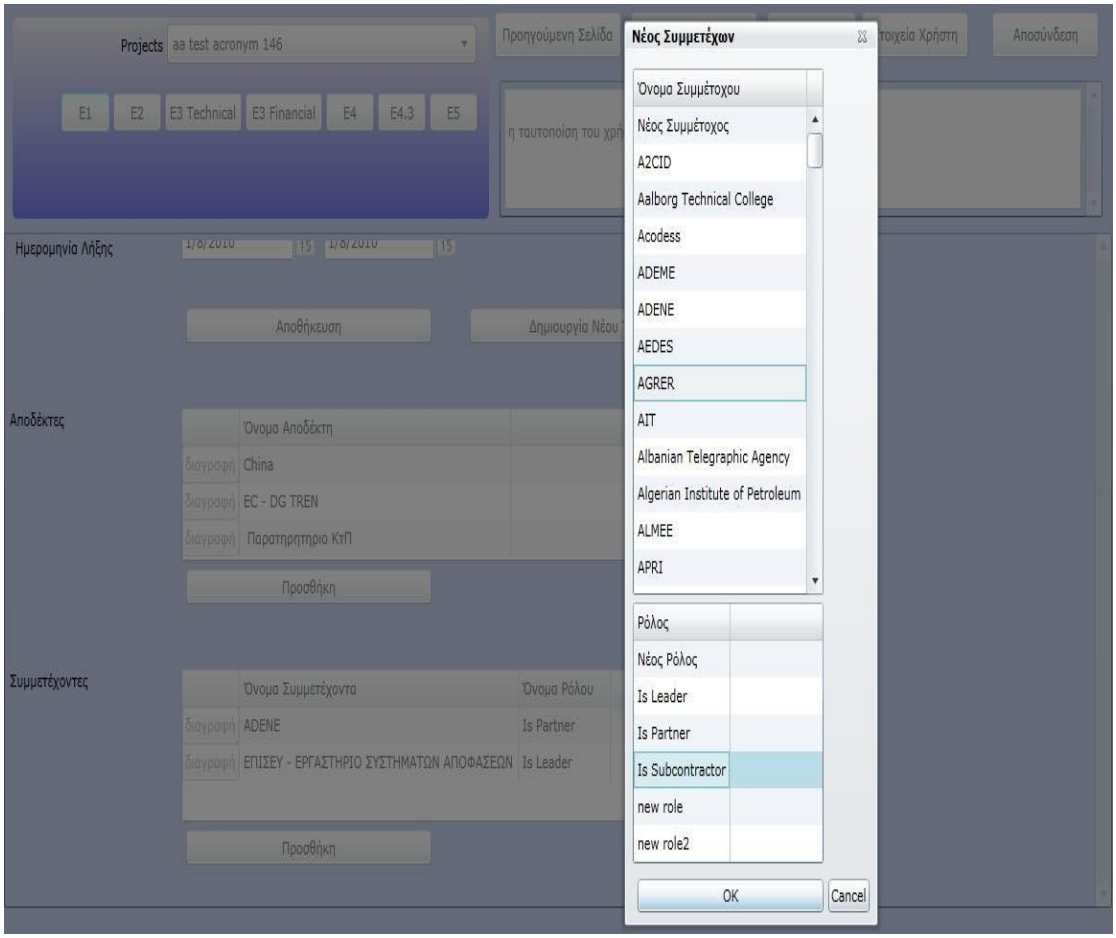
Η πρώτη εικόνα είναι τα πρώτα πεδία της κάρτας, εμφανιζόμενα ως TextBoxes, στα οποία ο χρήστης μπορεί να κάνει τις αλλαγές που θέλει (προφανώς πρόκειται για Project Director).

Για να αποθηκευθούν οι αλλαγές, ο χρήστης θα πρέπει να πατήσει το κουμπί Αποθήκευση. Να σημειώσουμε εδώ πως θα ήταν δυνατή μια υλοποίηση που προσφέρει μεγαλύτερη διαδραστικότητα: οι αλλαγές που κάνει ο χρήστης σε κάθε πεδίο να στέλνονται στα web services για αποθήκευση κατευθείαν, χωρίς ο χρήστης να χρειάζεται να πατήσει κάποιο κουμπί. Δεν το σχεδιάσαμε έτσι, γιατί κρίναμε πως, κατά κανόνα, ο χρήστης θα επιθυμεί να κάνει μια συνολική ανασκόπηση των αλλαγών που έχει κάνει στα πεδία της κάρτας, προτού θελήσει να προβεί σε αποθήκευση των αλλαγών. Επίσης, η επιλογή που κάναμε οδηγεί σε απλούστερη υλοποίηση (βλέπε σχετική συζήτηση στην ενότητα των WCF Υπηρεσιών του κεφαλαίου της Υλοποίησης).

Στην δεύτερη εικόνα βλέπουμε την φόρμα επιλογής νέου Συμμετέχοντα. Τα υπόλοιπα πεδία της E1 Κάρτας, που φαίνονται στο παρασκήνιο της φόρμας, θα γίνουν πάλι προσβάσιμα στον χρήστη μόλις επιλέξει τον Συμμετέχοντα και πατήσει OK (ή Cancel). Μπορούμε να παρατηρήσουμε πως, εκτός από τα πεδία της Κάρτας και το κουμπί για Αποθήκευση, υπάρχει και η επιλογή για δημιουργία νέου έργου. Αυτή δίνει την δυνατότητα στον χρήστη να ορίσει το όνομα, το ακρωνύμιο και τον κωδικό για ένα καινούργιο έργο.


Projects
aa test acronym 146
Προηγούμενη Σελίδα
Επόμενη Σελίδα
Χρήστες
Στοιχεία Χρήστη
Αποσύνδεση
E1
E2
E3 Technical
E3 Financial
E4
E4.3
E5
η ταυτοποίση του χρήστη έγινε με επιτυχία
E1
Κωδικός
aa test code 146
Ακρωνύμιο
aa test acronym 146
Περιοχή
test area
Σύμβαση
sy
Τίτλος
aa test title 146
Ανθρωπομήνες Έργου
2
Ανθρωπομήνες Ομάδας
3
Σύντομη Περιγραφή
short description short descr!!!
Λέξεις Κλειδιά
test 146
Πραγματική
Προγραμματισμένη
Αναθεωρημένη
Διάρκεια
12
12
12
Πλήθος Workpackages
2
3

**Εικόνα 17**




Εικόνα 18

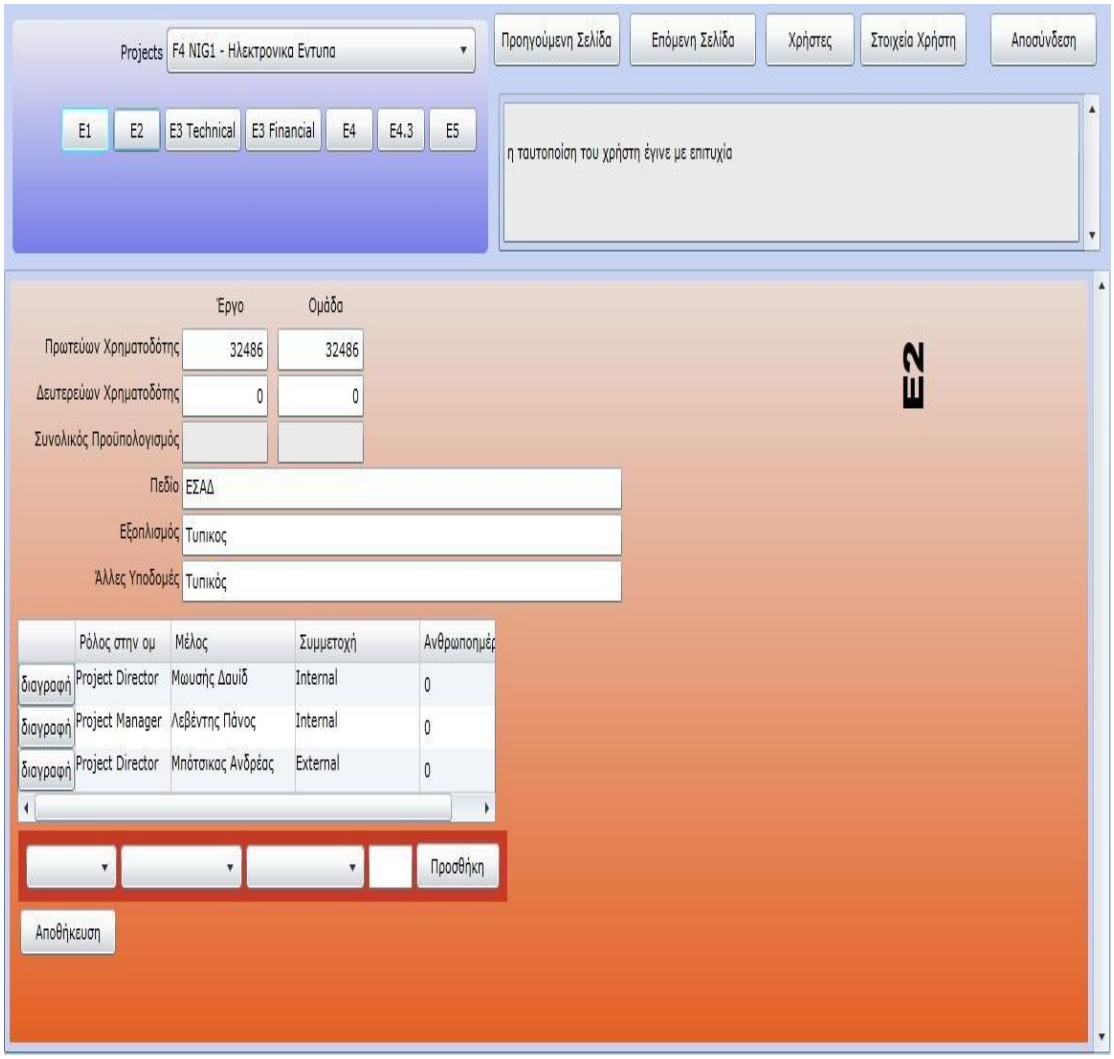
## 6.3 Σελίδα E2

Τα δικαιώματα στην ανάγνωση και γράψιμο της E2 Κάρτας είναι ανάλογα με αυτά της E1.

Η προσθήκη νέου μέλους στο έργο γίνεται στην γραμμή κάτω από την λίστα με τα μέλη. Σε αυτήν, ο χρήστης μπορεί να επιλέξει τα επιμέρους πεδία του νέου ανθρώπινου πόρου. Το συγκεκριμένο μέλος που θα αποτελέσει τον καινούργιο πόρο, καθορίζεται από μία αναδιπλούμενη λίστα, η οποία έχει τα ονοματεπώνυμα των χρηστών του συστήματος.

Όπως ήδη εξηγήσαμε στην προηγούμενη υποενότητα, δεν υπάρχει λόγος να εμφανίσουμε τα πεδία της E2 Κάρτας που ταυτοποιούν το έργο στο οποίο αναφέρεται.

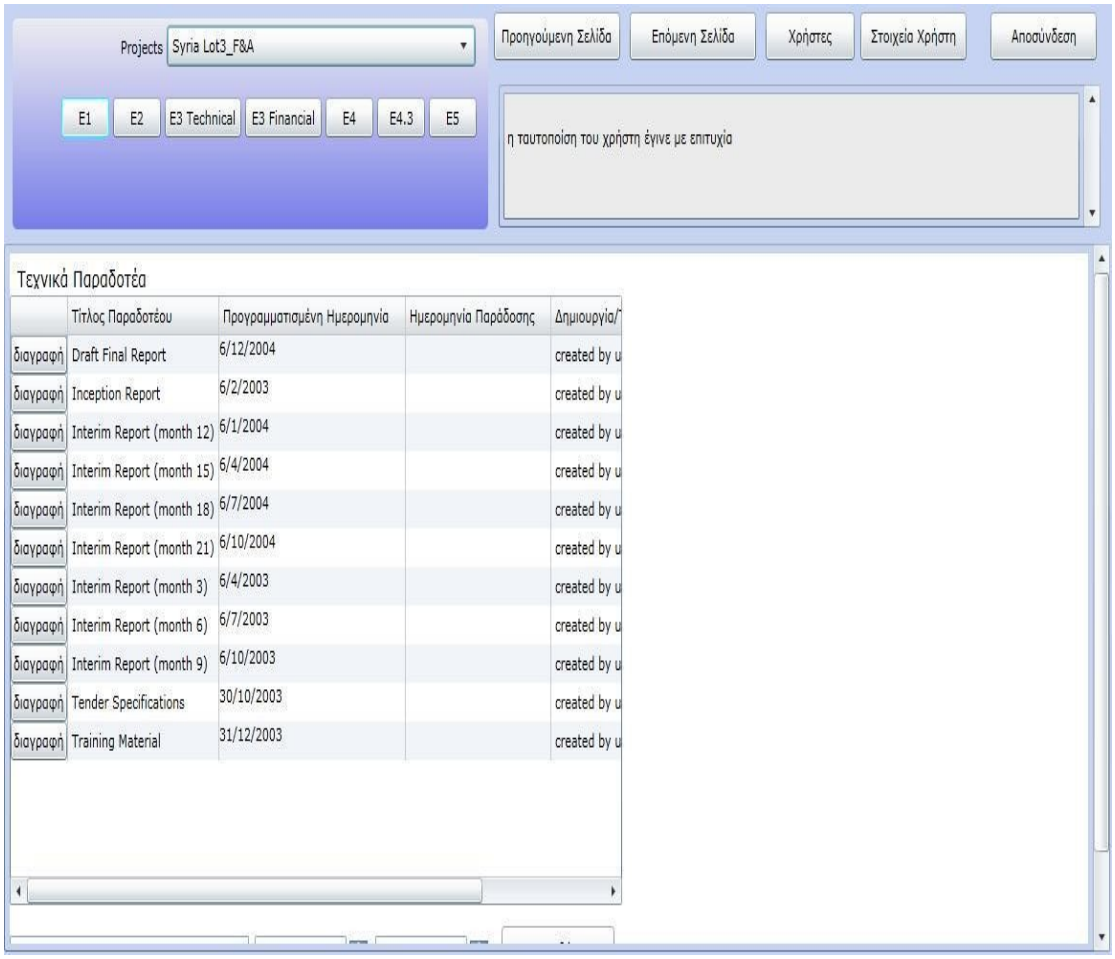
Η ενημέρωση της βάσης δεδομένων για τις αλλαγές που πραγματοποιήθηκαν γίνεται με το κουμπί Αποθήκευση (όπως σε όλες τις σελίδες διαχείρισης).



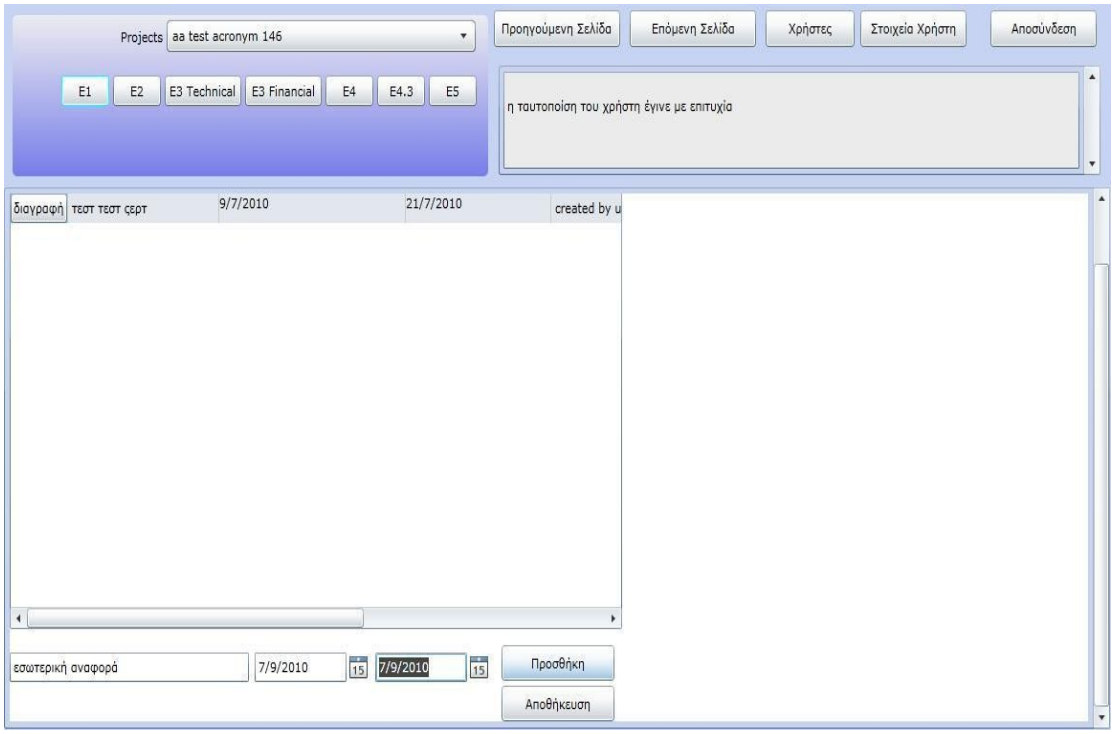

Εικόνα 19

## 6.4 Σελίδα E3 Τεχνικών Παραδοτέων

Τα επόμενα δύο screenshots είναι από την κάρτα E3 Τεχνικών Παραδοτέων. Το σχετικό κουμπί στο κεντρικό μενού είναι το E3 Technical. Η εισαγωγή νέου παραδοτέου γίνεται ανάλογα με την εισαγωγή νέου πόρου στην E2 κάρτα· με μια γραμμή κάτω από την λίστα των παραδοτέων, η οποία διακρίνεται στο δεύτερο screenshot.



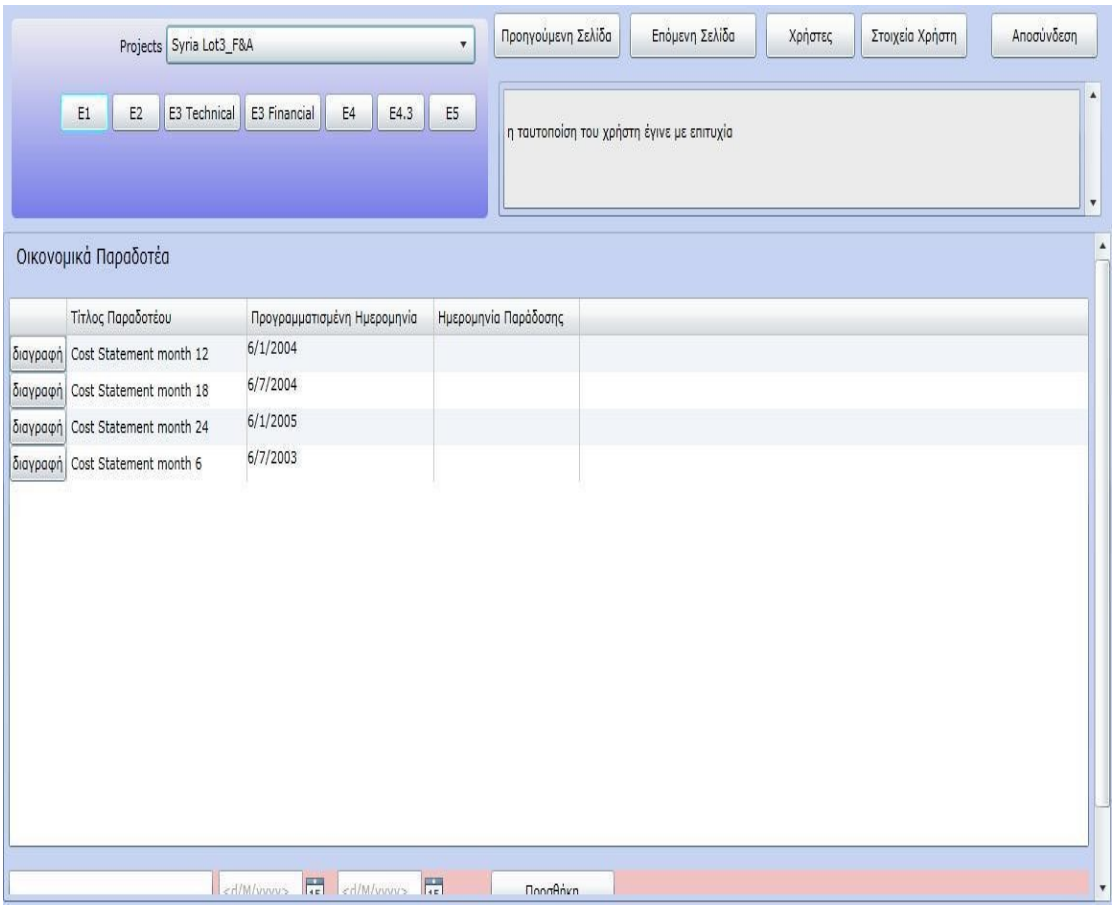

**Εικόνα 20**




**Εικόνα 21**

## 6.5 Σελίδα E3 Οικονομικών Παραδοτέων

Η σελίδα των οικονομικών παραδοτέων είναι όμοια με αυτή των τεχνικών παραδοτέων:



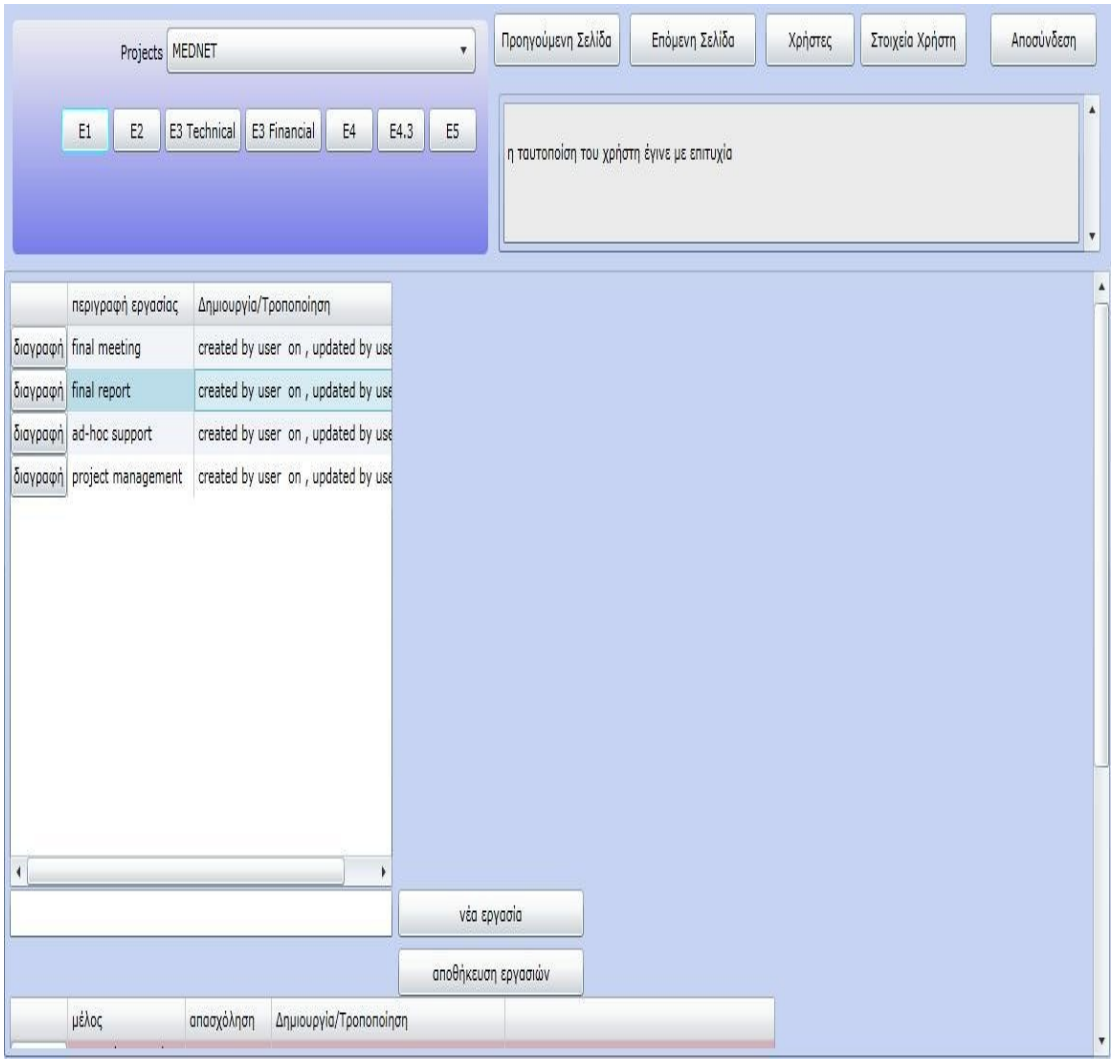

Εικόνα 22

## 6.6 Σελίδα E4

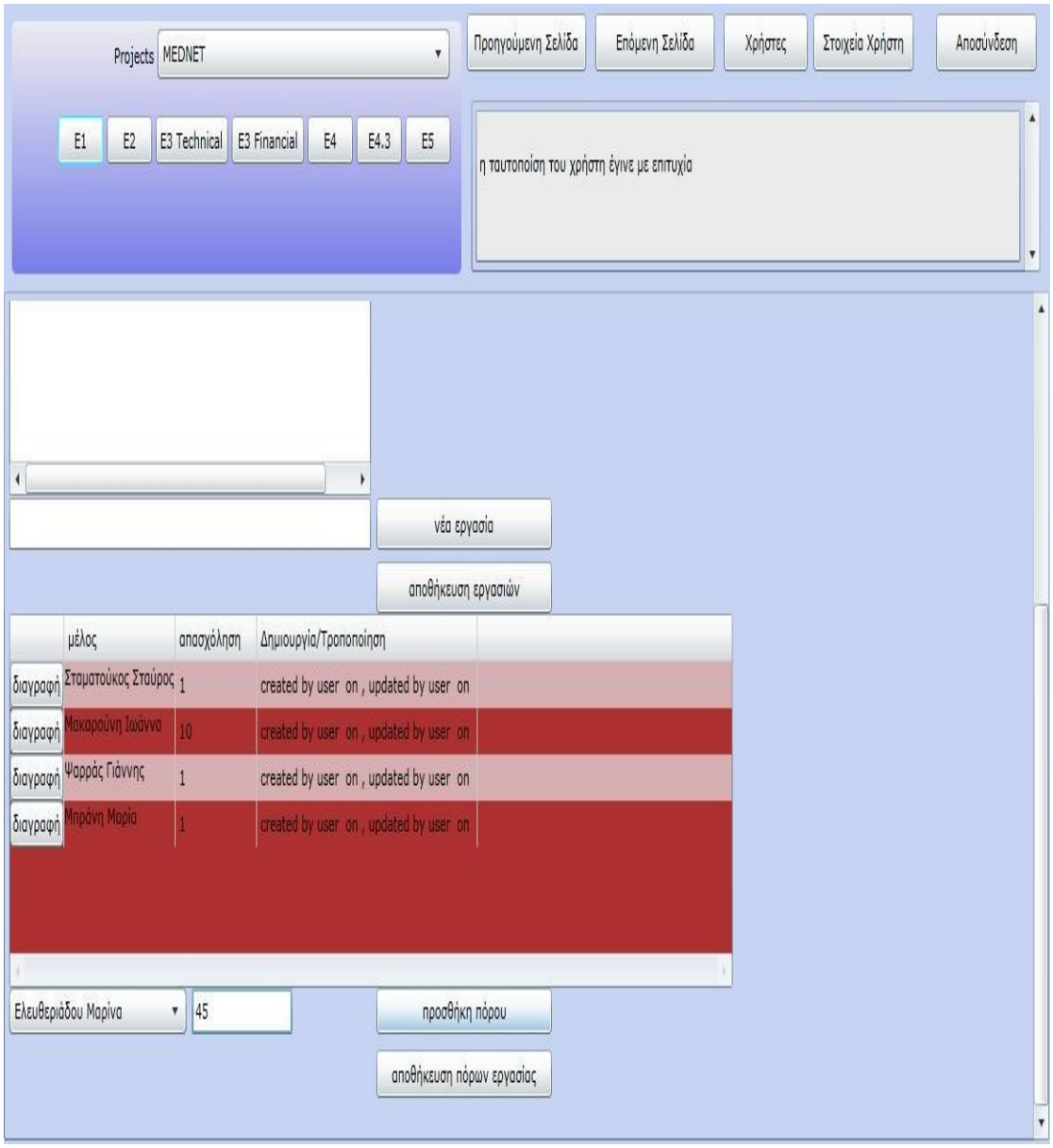
Η Κάρτα E4 αποτελείται από δύο πίνακες: έναν για την διαχείριση των E4 εργασιών και έναν για την διαχείριση των πόρων που δεσμεύει η κάθε εργασία.

Στον πίνακα διαχείρισης εργασιών ο χρήστης μπορεί να αλλάξει την περιγραφή των εργασιών, και να προσθαφαιρέσει εργασίες (23). Για κάθε εργασία που επιλέγει, εμφανίζονται στο δεύτερο πίνακα, από κάτω, οι ανθρώπινοι πόροι που δεσμεύει (Εικόνα 24).

Στην γραμμή προσθήκης νέου ανθρώπινου πόρου, η αναδιπλούμενη λίστα εμφανίζει τα ονόματα μόνο των χρηστών που είναι μέλη στο συγκεκριμένο έργο.


Projects
MEDNET
Προηγούμενη Σελίδα
Επόμενη Σελίδα
Χρήστες
Στοιχεία Χρήστη
Αποσύνδεση
E1
E2
E3 Technical
E3 Financial
E4
E4.3
E5
η ταυτοποίση του χρήστη έγινε με επιτυχία
περιγραφή εργασίας
Δημιουργία/Τροποποίηση
διαγραφή
final meeting
created by user on , updated by use
διαγραφή
final report
created by user on , updated by use
διαγραφή
ad-hoc support
created by user on , updated by use
διαγραφή
project management
created by user on , updated by use
νέα εργασία
αποθήκευση εργασιών
μέλος
απασχόληση
Δημιουργία/Τροποποίηση

**Εικόνα 23**




Εικόνα 24

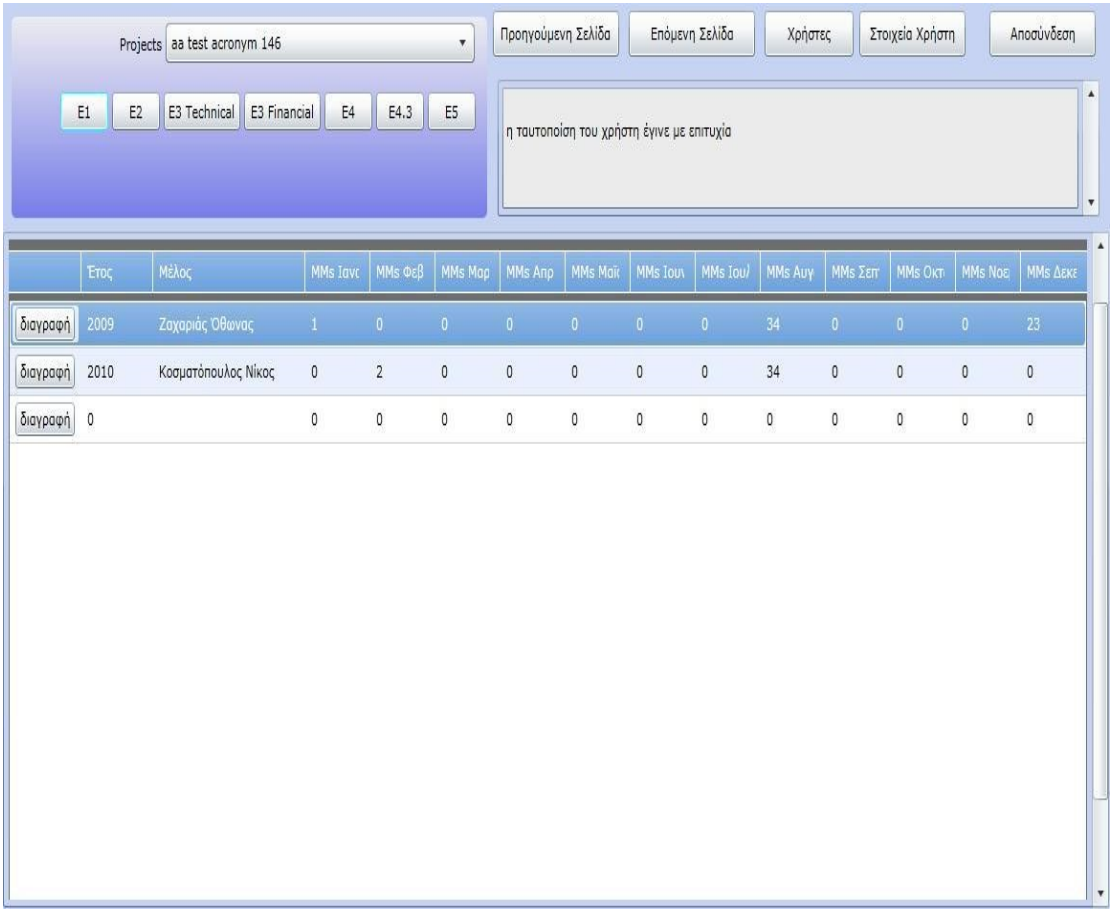
## 6.7 Σελίδα E43

Με την E43 Κάρτα οι διαχειριστές μπορούν να ορίσουν τον χρόνο εργασίας του κάθε μέλους για τους μήνες που διαρκεί το έργο.

Οι πληροφορίες εμφανίζονται σε έναν πίνακα, με μια γραμμή να αντιστοιχεί στους μήνες ενός συγκεκριμένου μέλους για ένα συγκεκριμένο έτος. Η τελευταία γραμμή είναι πάντα κενή και προορίζεται για την προσθήκη μιας νέας εγγραφής.

Οι εγγραφές σε αυτόν τον πίνακα μπορούν να ομαδοποιηθούν σύμφωνα με όποια στήλη επιθυμεί ο χρήστης, απλώς σύροντας την επικεφαλίδα

της στήλης στο πάνω μέρος του πίνακα. Είναι μάλιστα δυνατή η ομαδοποίηση σύμφωνα με δυο πίνακες ταυτόχρονα, απλώς σύροντας και δεύτερη στήλη. Αν και αυτή η λειτουργικότητα δεν παρουσιάζεται στο επόμενο screenshot, θα απεικονιστεί στα screenshots της επόμενης κάρτας διαχείρισης, η οποία χρησιμοποιεί παρόμοιους πίνακες.



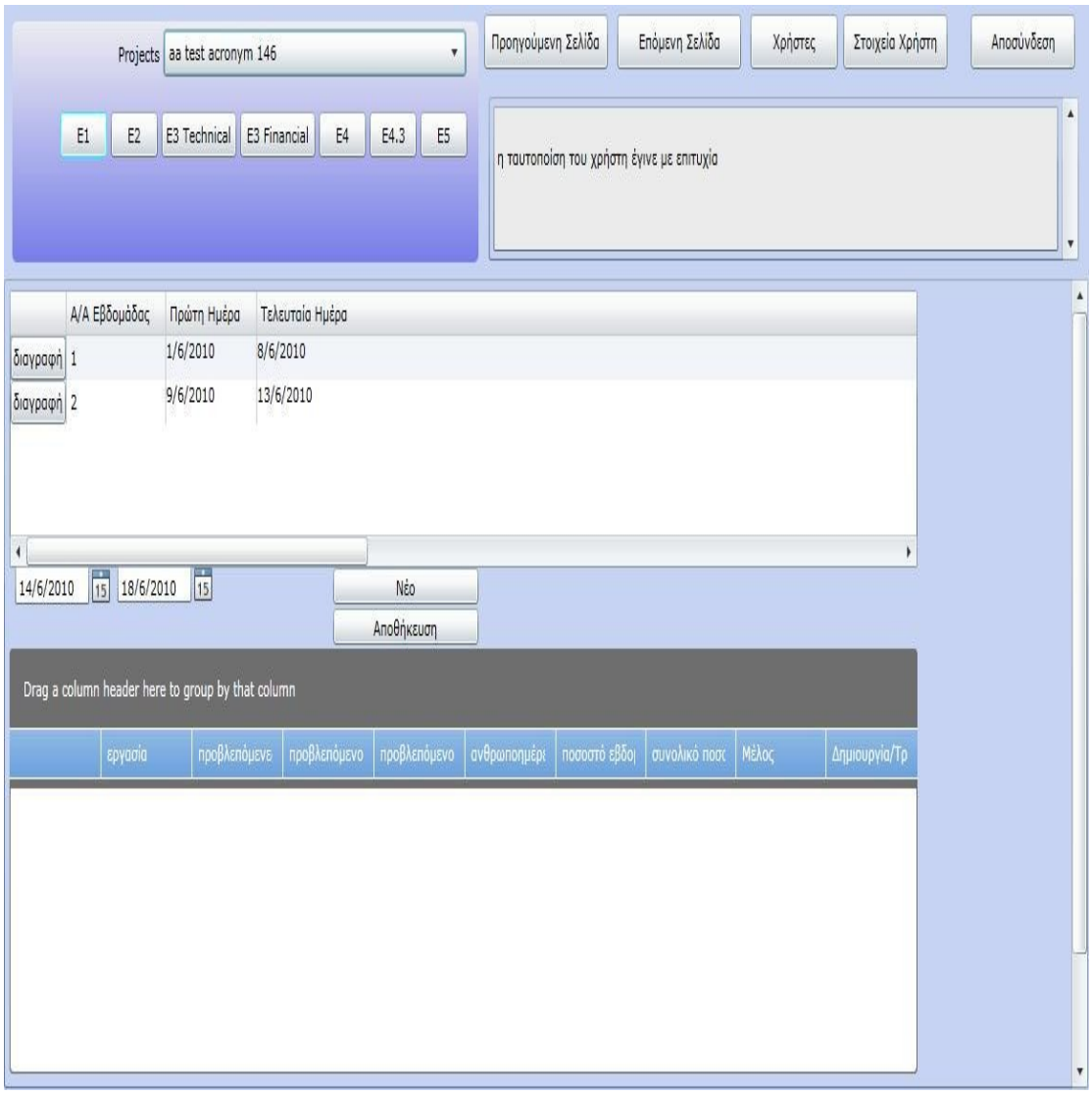

Εικόνα 25

## 6.8 Σελίδα E5

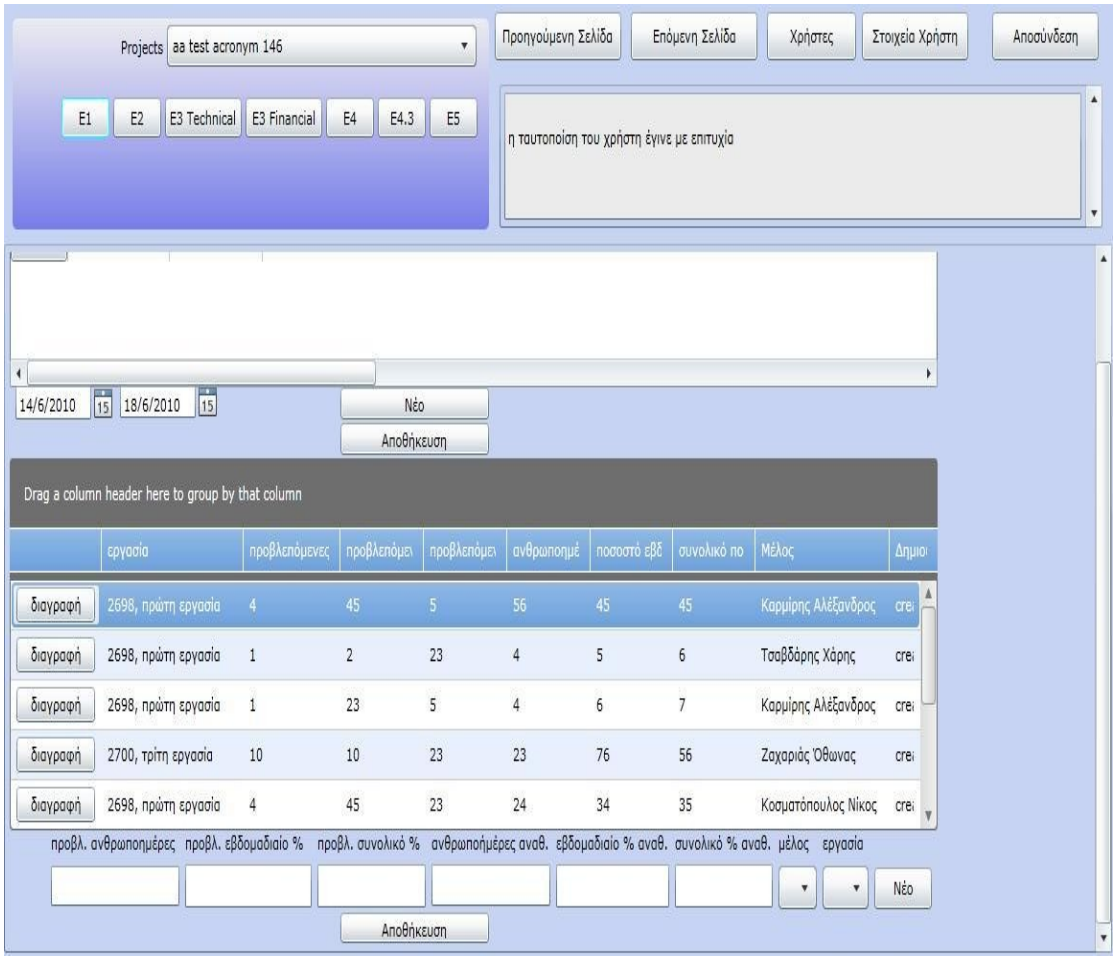
Η E5 Κάρτα αποτελείται και αυτή από δύο πίνακες, έναν για την καταγραφή των εργάσιμων εβδομάδων του έργου και ένα για τον χρονοπρογραμματισμό της κάθε εβδομάδας, ορίζοντας για καθεμία ποιοι θα εργαστούν, για ποιες E4 εργασίες και για πόσο.

Για τον δεύτερο πίνακα, είναι πολύ χρήσιμη η δυνατότητα της ομαδοποίησης των εγγραφών σύμφωνα με κάποια στήλη, συνηθέστερα αυτή του ονόματος του μέλους και αυτή που ορίζει την E4 εργασία.

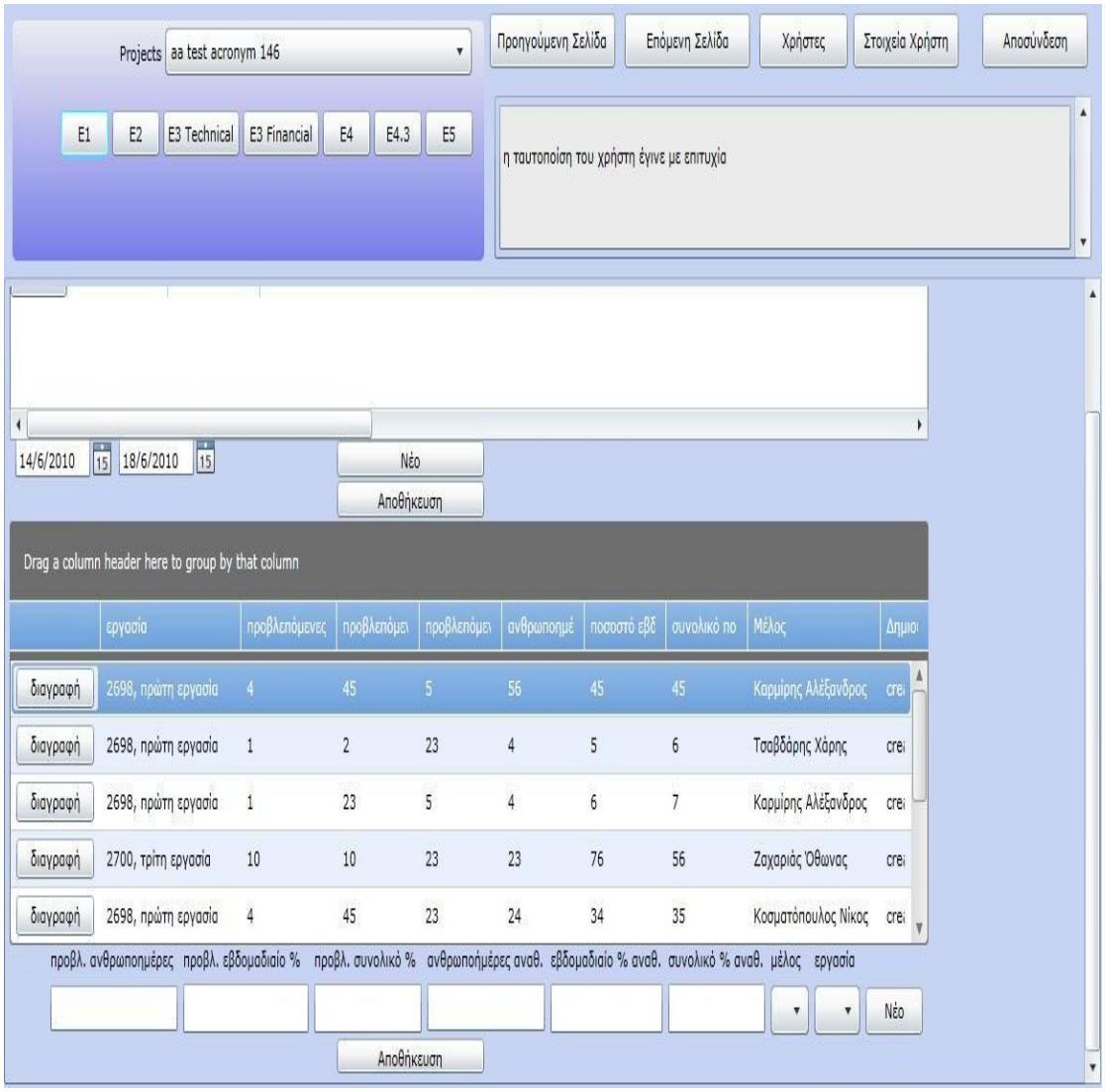
Στα παρακάτω screenshots, απεικονίζεται πρώτα ο πρώτος πίνακας, με τις εβδομάδες κάποιου έργου. Στα επόμενα δύο, απεικονίζεται ο δεύτερος πίνακας για την πρώτη εβδομάδα του συγκεκριμένου έργου, πρώτα χωρίς να έχει γίνει κάποια ομαδοποίηση και έπειτα ταξινομημένες και ομαδοποιημένες ανά Ε4 Εργασία.




Εικόνα 26


Projects
aa test acronym 146
Προηγούμενη Σελίδα
Επόμενη Σελίδα
Χρήστες
Στοιχεία Χρήστη
Αποσύνδεση
E1
E2
E3 Technical
E3 Financial
E4
E4.3
E5
η ταυτοποίση του χρήστη έγινε με επιτυχία
14/6/2010
18/6/2010
Νέο
Αποθήκευση
Drag a column header here to group by that column
εργασία
προβλεπόμενες
ανθρωποημέ
ποσοστό εβδ
συνολικό πο
Μέλος
διαγραφή
2698, πρώτη εργασία
Καρμίρης Αλέξανδρος
2698, πρώτη εργασία
Τσαβδάρης Χάρης
2698, πρώτη εργασία
Καρμίρης Αλέξανδρος
2700, τρίτη εργασία
Ζαχαριάς Όθωνας
2698, πρώτη εργασία
Κοσματόπουλος Νίκος
προβλ. ανθρωποημέρες
προβλ. εβδομαδιαίο %
προβλ. συνολικό %
ανθρωποήμέρες αναθ.
εβδομαδιαίο % αναθ.
συνολικό % αναθ.
μέλος
εργασία
Νέο
Αποθήκευση

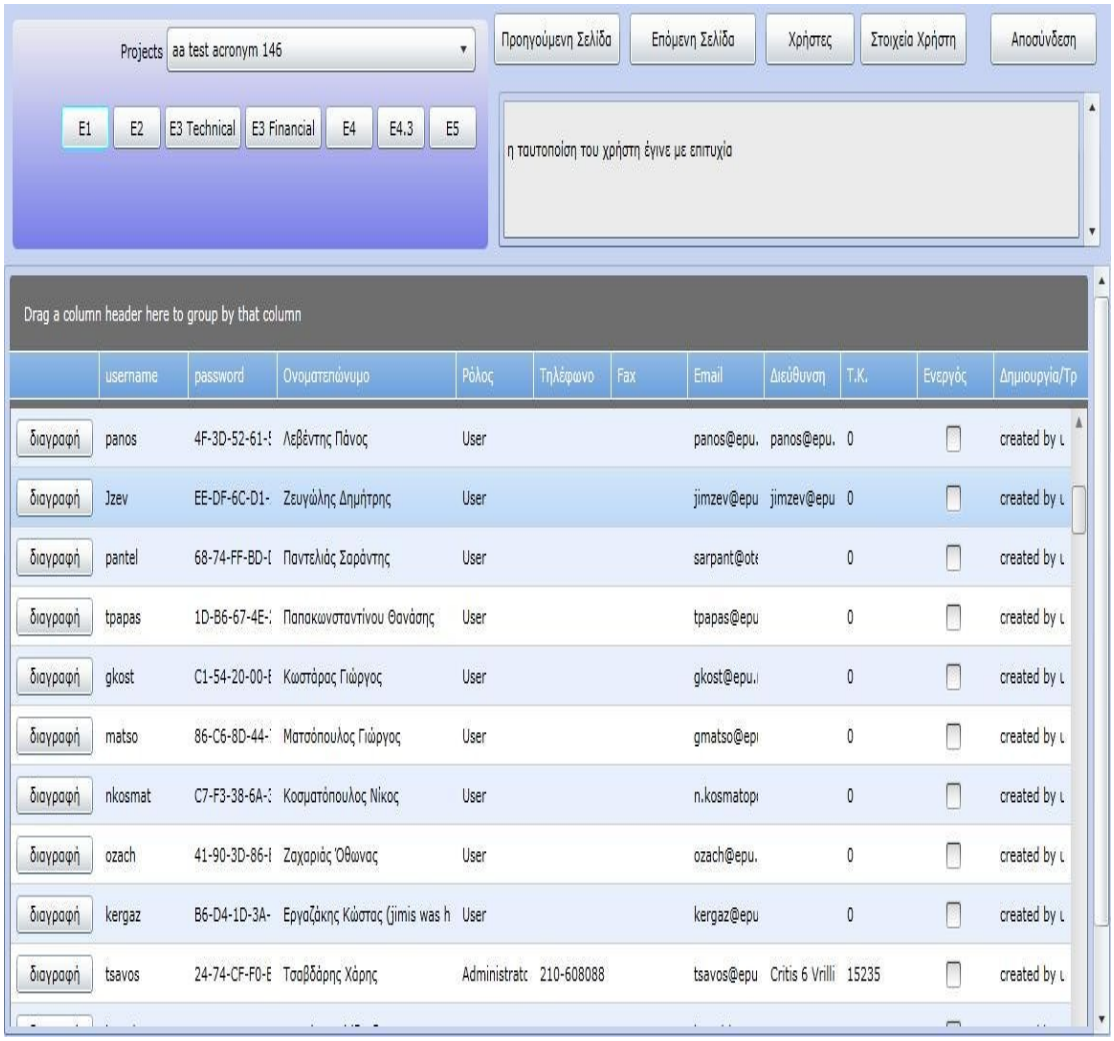
**Εικόνα 27**




Εικόνα 28

## 6.9 Διαχείριση Χρηστών

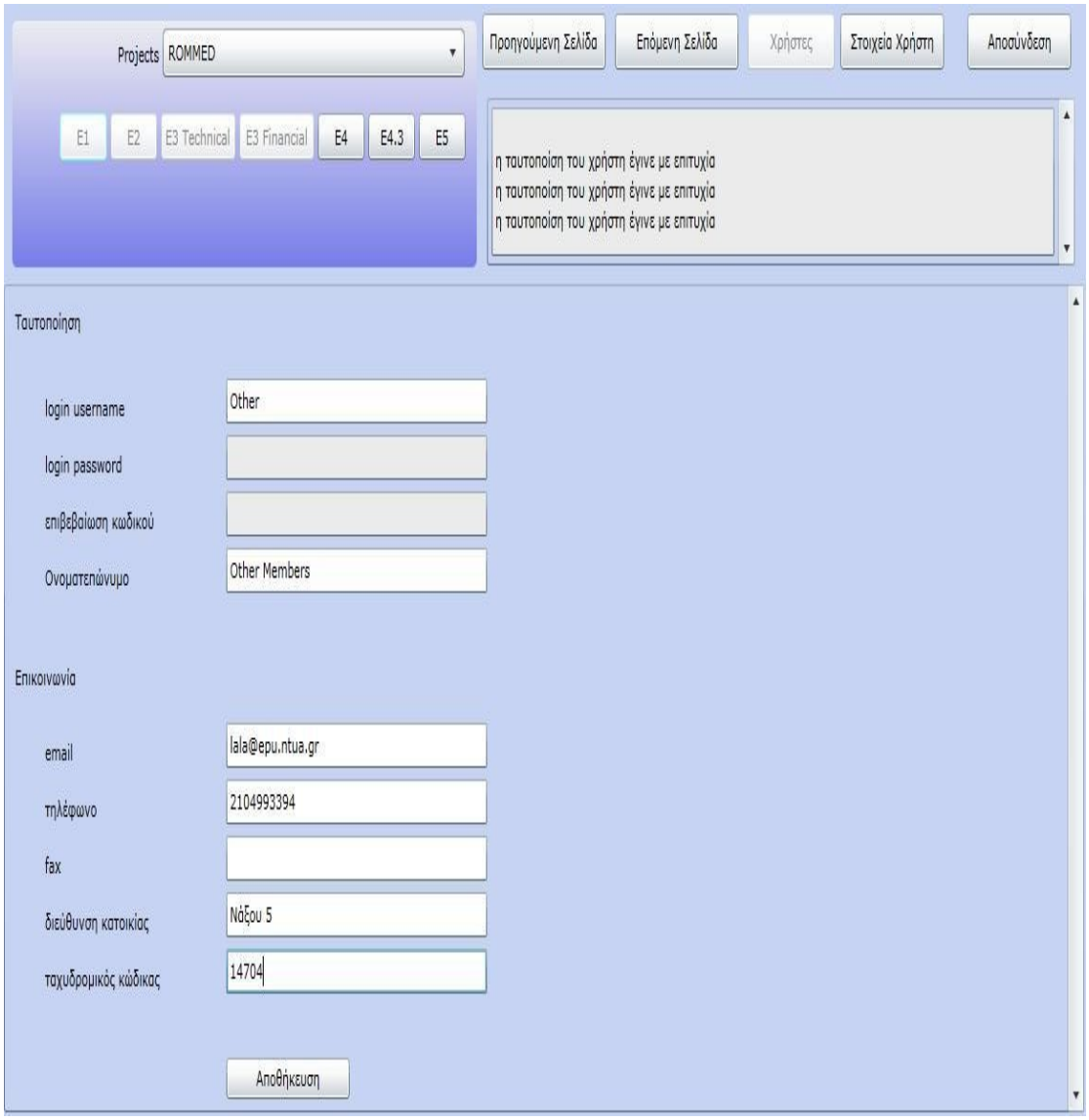
Η σελίδα αυτή είναι διαθέσιμη μόνο στους διαχειριστές του συστήματος. Εμφανίζει σε έναν πίνακα τα προσωπικά στοιχεία όλων των χρηστών του συστημάτων, και επιτρέπει την ομαδοποίηση της παρουσίασης σύμφωνα με μια ή περισσότερες στήλες, την μεταβολή των στοιχείων και την προσθαφαίρεση χρηστών.




Εικόνα 29

## 6.10 Προσωπικά Στοιχεία

Από εδώ οι χρήστες μπορούν να αλλάξουν τα προσωπικά τους στοιχεία, περιλαμβανομένου του username τους και του συνθηματικού εισόδου στο σύστημα


Projects
ROMMED
Προηγούμενη Σελίδα
Επόμενη Σελίδα
Χρήστες
Στοιχεία Χρήστη
Αποσύνδεση
E1
E2
E3 Technical
E3 Financial
E4
E4.3
E5
η ταυτοποίση του χρήστη έγινε με επιτυχία
η ταυτοποίση του χρήστη έγινε με επιτυχία
η ταυτοποίση του χρήστη έγινε με επιτυχία
Ταυτοποίηση
login username
Other
login password
επιβεβαίωση κωδικού
Ονοματεπώνυμο
Other Members
Επικοινωνία
email
lala@epu.ntua.gr
τηλέφωνο
2104993394
fax
διεύθυνση κατοικίας
Νάξου 5
ταχυδρομικός κώδικας
14704
Αποθήκευση

**Εικόνα 30**

# 7. Επίλογος

## 7.1 Αξιολόγηση

Το σύστημα ικανοποιεί τις βασικές λειτουργικές και μη απαιτήσεις που είχαν τεθεί από πριν.

Πιο συγκεκριμένα, όσον αφορά τον εξυπηρετητή, παρέχεται ένα αρκετά ισχυρό σχήμα προστασίας της επικοινωνίας με τους πελάτες και των δεδομένων που είναι αποθηκευμένα στη βάση (βλέπε την ενότητα την σχετική με την Ασφάλεια στο κεφάλαιο της Υλοποίησης). Οι υπηρεσίες μπορούν να ταυτοποιούν τον πελάτη και να τον περιορίζουν στις εργασίες διαχείρισης που είναι εξουσιοδοτημένος να κάνει. Δεν γίνονται υποθέσεις για το περιβάλλον εκτέλεσης του πελάτη και την φύση του (browser add-on, desktop εφαρμογή, κτλ) και επιτρέπεται η ταυτόχρονη σύνδεση πολλών πελατών. Η βάση δεδομένων είναι ακριβώς η ίδια με αυτή της προϋπάρχουσας εφαρμογής, κάνοντας έτσι εύκολη την μετάβαση από το ένα σύστημα στο άλλο.

Όσον αφορά τον πελάτη, η υλοποίηση ικανοποιεί σε μεγάλο βαθμό τις κύριες απαιτήσεις: διαδραστικότητα, δομή σελίδων διαχείρισης παρόμοια με αυτήν της προηγούμενης εφαρμογής, δυνατότητα εκτέλεσης της εφαρμογής σε πληθώρα web browser και στις ζητούμενες εκδόσεις των Windows και Mac.

## 7.2 Πιθανές Βελτιώσεις της Εφαρμογής και Επεκτάσεις Λειτουργικότητας

Η εφαρμογή του πελάτη έχει περιθώρια βελτίωσης στους παρακάτω τομείς:

- Μεγαλύτερη ομοιομορφία στο interface.
- Καλύτερη αξιοποίηση του διαθέσιμου χώρου στην οθόνη.
- Μείωση του όγκου των δεδομένων που μεταφέρονται κατά την αποθήκευση των αλλαγών στους πίνακες, μεταφέροντας μόνο τις εγγραφές που έχουν αλλάξει.

Η λειτουργικότητα του πελάτη θα μπορούσε να επεκταθεί ώστε να περιλαμβάνει:

- Σελίδες αναφορών, μέσω των οποίων ο διαχειριστής θα μπορεί να αποκτά εύκολα μια γενική εικόνα της πορείας του έργου.
- Δυνατότητα για άμεση εκτύπωση των περιεχομένων των σελίδων.

Όσον αφορά τον εξυπηρετητή, βασικές προσθήκες και επεκτάσεις θα μπορούσαν να είναι:

- Η προσθήκη CRUD υπηρεσιών (βλέπε την ενότητα των WCF υπηρεσιών, του κεφαλαίου της υλοποίησης).
- Η καλύτερη υποστήριξη ταυτόχρονης διαχείρισης του ίδιου έργου από διαφορετικούς πελάτες. Αν και κάτι τέτοιο είναι δυνατό με το υπάρχον σύστημα, αυτό δεν γίνεται με ελεγχόμενο τρόπο. Δεν υπάρχει, για παράδειγμα, τρόπος να μάθει ένας πελάτης αν κάποιος άλλος χρήστης κάνει μεταβολές σε κάποιο έργο την στιγμή που το προσπελαύνει αυτός. Για να μπορέσει να γίνει αυτό, θα πρέπει ο πελάτης να κάνει γνωστό πως πρόκειται να κάνει μεταβολές σε κάποιο έργο, να αποθηκεύεται αυτή η πληροφορία στην βάση δεδομένων και να ενημερώνονται σχετικά όσοι άλλοι πελάτες θελήσουν να πραγματοποιήσουν κάποια εργασία διαχείρισης.

# 8. Βιβλιογραφία Και Πηγές


1. **(PMI), Project Management Institute.** A Guide To Project Management Body Of Knowledge. s.l. : Project Management Institute (PMI), 2004. διεθνώς καθιερωμένο, περιγράφει μια παραδοσιακή, βασισμένη σε διαδικασίες μεθοδολογία διαχείρισης.
2. **Kerzner, Harold.** Project Management, A Systems Approach To Planning, Scheduling, and Controlling. s.l. : John Wiley and Sons, 2009. Από τις πλέον καθιερωμένες αναφορές, η οποία μπορεί μάλιστα να προσπελαστεί δωρεάν online (για παράδειγμα, από το books.google.gr).
3. **Turner, J. Rodney.** *The Handbook Of Project-Based Management.* 2. s.l. : McGRAW-HILL, 1999.
4. **Goldratt, Eliyahu.** *Critical Chain.* 1997.
5. **Virine, Lev and Trumper, Michael.** *Schedule Network Analysis Using Event Chain.* s.l. : ProjectDecisions.org.
6. **Staff, Office of Government Commerce.** *Managing Successful Projects with Prince2 2009.* 5. s.l. : Stationary Office Bo, 2009.
7. **Department for Business, Enterprise and Regulatory Reform.** *Guidelines For Managing Projects.* 2007. συμβουλές για την διαχείριση έργου, συμβατές με την PRINCE2 μεθοδολογία. Μπορεί να βρεθεί στο http://www.berr.gov.uk/files/file40647.pdf.
8. **Garrett, Jesse James.** Ajax: A New Approach to Web Applications. *www.adaptivepath.com.* [Online] February 18, 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php.
9. **Wikipedia.** Microsoft Silverlight. *wikipedia.org.* [Online] Wikipedia. [Cited: August 27, 2010.] www.wikipedia.org/wiki/Microsoft_Silverlight.
10. **Erl, Thomas.** *SOA Design Patterns.* s.l. : Prentice Hall, 2009.
11. **Μπότσικας, Ανδρέας.** Ανάπτυξη Διαδικτυακής Εφαρμογής Για Την Διαχείριση Έργων. s.l. : ΕΜΠ, Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, 2003. μπορεί να βρεθεί στην διαδικτυακή βάση δεδομένων artemis.cslab.ntua.gr.
12. **Microsoft Corporation.** The ADO.NET Entity Framework Overview. *msdn.microsoft.com.* [Ηλεκτρονικό] June 2006. http://msdn.microsoft.com/en-us/library/aa697427(VS.80).aspx#ado.netenfrmovw_topic4.
13. **Fowler, Martin.** Presentation Model. *martinfowler.com.* [Ηλεκτρονικό] 2004. http://martinfowler.com/eaaDev/PresentationModel.html.
14. **Bugnion, Laurent.** Understanding The Model-View-ViewModel Pattern. *live.visitmix.com.* [Ηλεκτρονικό] 2010. http://live.visitmix.com/MIX10/Sessions/EX14.
15. **Microsoft Corporation.** Navigation Overview. *msdn.microsoft.com.* [Ηλεκτρονικό] 2010.
16. *Managing the Development of Large Software Systems.* **Royce, Winston W.** 1970. Proceedings of IEEE WESCON 26 (August): 1–9.
17. **Parnas, David.** *A rational design process and how to fake it.* δημοσίευση που έχει ασκήσει ιδιαίτερη επιρροή και αναφέρεται συχνά. Μπορεί να βρεθεί στην διεύθυνση: http://users.ece.utexas.edu/~perry/education/SE-Intro/fakeit.pdf.
18. **McConnell, Steve.** *Code Complete: A Practical Handbook of Software Construction.* 2. s.l. : Microsoft Press, 2004.

19. **Cockburn, Alistair.** *Agile Software Development.* s.l. : Addison Wesley Longman, 2001.
20. **Fowler, Martin.** The New Methodology. *martinfowler.com.* [Online] December 5, 2005. http://martinfowler.com/articles/newMethodology.html.
21. **IEEE.** Draft IEEE Standard for software and system test documentation (Revision of IEEE 829-1998). s.l. : IEEE, 2008.
22. Event Chain Methodology. *www.intaver.com.* [Ηλεκτρονικό] http://www.intaver.com/Articles/RP_Art_EventChainMethodology.html.
23. **Wikipedia.** Windows Presentation Foundation. *wikipedia.org.* [Ηλεκτρονικό] Wikipedia. [Παραπομπή: 27 August 2010.] en.wikipedia.org/wiki/Windows_Presentation_Foundation#XAML.
24. **Microsoft Corporation.** XAML Overview. *msdn.microsoft.com.* [Online] msdn.microsoft.com/en-us/library/ms752059.aspx.
25. **Wikipedia.** Windows Workflow Foundation. *wikipedia.org.* [Ηλεκτρονικό] Wikipedia. [Παραπομπή: 27 August 2010.] en.wikipedia.org/wiki/Windows_Workflow_Foundation#Authoring_Workflows.
26. [Ηλεκτρονικό] [Παραπομπή: 27 August 2010.] msdn.microsoft.com/en-us/library/ms752059.aspx.

# 9. Επιλεγμένα Τμήματα του Κώδικα της Εφαρμογής

Έχουμε ήδη παραθέσει, σε διάφορες ενότητες, τα βασικά κομμάτια του κώδικα του εξυπηρετητή. Εδώ θα κάνουμε το ίδιο και για τον κώδικα του πελάτη.

Οι κλάσεις των σελίδων διαχείρισης, κληρονομούν όλες την κλάση abstract κλάση EPage:

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace Isotrack3.View
{
   public abstract class EPage : Page
   {
      public Object VModelSync = new object();
      public ViewModel.AbstractViewModel VModel;

      protected bool IsLoaded = false;

      protected abstract void ResetViewData();
      protected abstract void ResetModel();
      protected abstract void InitializeModelHandlers();

      protected void InitializeHandlers()
      {
         Loaded += new RoutedEventHandler(Page_Loaded);
         Unloaded += new
RoutedEventHandler(Page_Unloaded);

         ViewModel.StaticInfo.ProjectChanged += new
ViewModel.StaticInfo.ProjectChangedEventHandler(StaticInfo_
ProjectChanged);
         ViewModel.StaticInfo.User.UserCredentialsChanged
+= new
```

```
ViewModel.LoggedUser.UserCredentialsChangedEventHandler
(User_UserCredentialsChanged);
        }

        protected void User_UserCredentialsChanged(object
sender)
        {
            lock (VModelSync)
            {
                if (!IsLoaded)
                    return;

                VModel.SetUserCredentials();
            }
        }

        protected void StaticInfo_ProjectChanged()
        {
            lock (VModelSync)
            {
                if (!IsLoaded)
                    return;

                ResetViewData();
                ResetModel();
                InitializeModelHandlers();
                VModel.SetUserCredentials();

                if (ViewModel.StaticInfo.Project.fldProjectID == 0)
                    return;
                VModel.ModelUpdateAsync(ViewModel.StaticInfo.Pro
ject);
            }
        }

        protected void Page_Unloaded(object sender, EventArgs
e)
        {
            lock (VModelSync)
            {
                IsLoaded = false;
                VModel = null;
            }
        }

        protected void Page_Loaded(object sender, EventArgs e)
        {
            lock (VModelSync)
            {
```

```
                IsLoaded = true;

                ResetViewData();
                ResetModel();
                InitializeModelHandlers();
                VModel.SetUserCredentials();

                if (ViewModel.StaticInfo.Project.fldProjectID == 0)
                   return;
                VModel.ModelUpdateAsync(ViewModel.StaticInfo.Pro
ject);
            }
        }
    }
}
```

Για παράδειγμα, η σελίδα διαχείρισης των χρηστών σχεδιάζεται από την κλάση PageUsers. Ο κώδικας XAML αυτής της κλάσης είναι:

```
<localview:EPage
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml
/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xa
ml"
        xmlns:d="http://schemas.microsoft.com/expression/ble
nd/2008"
        xmlns:mc="http://schemas.openxmlformats.org/marku
p-compatibility/2006"
        mc:Ignorable="d"
        xmlns:localview="clr-namespace:Isotrack3.View"
        xmlns:localviewmodel="clr-
namespace:Isotrack3.ViewModel"
        xmlns:dx="clr-
namespace:DevExpress.AgDataGrid;assembly=DevExpress.Ag
DataGrid.v8.2"
        xmlns:navigation="clr-
namespace:System.Windows.Controls;assembly=System.Win
dows.Controls.Navigation"
        xmlns:controls="clr-
namespace:System.Windows.Controls;assembly=System.Win
dows.Controls" xmlns:data="clr-
namespace:System.Windows.Controls;assembly=System.Win
dows.Controls.Data" x:Class="Isotrack3.View.PageUsers"
        d:DesignWidth="958" d:DesignHeight="480"
        HorizontalAlignment="Stretch"
       VerticalAlignment="Stretch"
```

```
        Title="Users Page">

    <localview:EPage.Resources>
        <localviewmodel:LoggedUserProxy x:Key="User" />
    </localview:EPage.Resources>

    <Grid x:Name="LayoutRoot"
HorizontalAlignment="Stretch"  VerticalAlignment="Stretch"
>
        <Grid.RowDefinitions>
            <RowDefinition Height="446" />
            <RowDefinition Height="34" />
        </Grid.RowDefinitions>
        <dx:AgDataGrid x:Name="usersdg"
                       AutoGenerateColumns="False"
                       ChangeOddRowsAppearance="True"
                       ColumnsAutoWidth="True"
                       ShowGroupPanel="Visible"
                       AllowColumnGrouping="True"
                       AllowColumnSorting="True"
                       AllowColumnResizing="True"
                       AllowColumnMoving="True"
                       SelectionMode="None">
            <dx:AgDataGrid.Columns>
                <dx:AgDataGridColumn>
                    <dx:AgDataGridColumn.CellDisplayTemplate>
                        <DataTemplate>
                            <Button Content="διαγραφή"
ClickMode="Release" Click="user_delete_b" />
                        </DataTemplate>
                    </dx:AgDataGridColumn.CellDisplayTemplate>
                </dx:AgDataGridColumn>
                <dx:AgDataGridTextColumn
FieldName="LoginName" HeaderContent="username" />
                <dx:AgDataGridTextColumn
FieldName="LoginPassword" HeaderContent="password" />
                <dx:AgDataGridTextColumn FieldName="Fullname"
HeaderContent="Ονοματεπώνυμο" />
                <dx:AgDataGridColumn FieldName="Role"
HeaderContent="Ρόλος"
                                     BeginEditing="AgDataGridColumn_Beg
inEditing"
                                     EndEditing="AgDataGridColumn_EndE
diting">
                    <dx:AgDataGridColumn.CellEditingTemplate>
                        <DataTemplate>
                            <ComboBox
                                DisplayMemberPath=""
                                ItemsSource="{Binding StaticRoles}"
```

```
                        />
                    </DataTemplate>
                </dx:AgDataGridColumn.CellEditingTemplate>
            </dx:AgDataGridColumn>
            <dx:AgDataGridTextColumn
FieldName="Telephone" HeaderContent="Τηλέφωνο" />
            <dx:AgDataGridTextColumn FieldName="Fax"
HeaderContent="Fax" />
            <dx:AgDataGridTextColumn FieldName="Mail"
HeaderContent="Email" />
            <dx:AgDataGridTextColumn FieldName="Address"
HeaderContent="Διεύθυνση" />
            <dx:AgDataGridTextColumn FieldName="PostCode"
HeaderContent="T.K." />
            <dx:AgDataGridCheckColumn FieldName="IsActive"
HeaderContent="Ενεργός" />

            <dx:AgDataGridTextColumn
                x:Name="createdupdated_col"
                HeaderContent="Δημιουργία/Τροποποίηση"
                AllowEditing="False"
                FieldName="Tooltip"
                />

        </dx:AgDataGrid.Columns>
    </dx:AgDataGrid>
    <Button x:Name="saveusers_b" Content="Αποθήκευση
Χρηστών" ClickMode="Release" Click="users_save_b"
Grid.Row="1" Margin="0,0,806,0" Width="152" />
  </Grid>

</localview:EPage>
```

Ο κώδικας C# (code-behind) παρέχει υλοποιήσεις για τους event handlers της σελίδας, και υλοποιεί τις abstract μεθόδους της EPage:

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
```

```
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Windows.Navigation;
using Isotrack3.ViewModel;

namespace Isotrack3.View
{
   public partial class PageUsers : EPage
   {
      private bool _HasLastRowBeenEdited;

      public PageUsers()
      {
         InitializeComponent();

         Loaded += new
RoutedEventHandler(PageUsers_Loaded);
         Unloaded += new
RoutedEventHandler(Page_Unloaded);

         ViewModel.StaticInfo.ProjectChanged += new
ViewModel.StaticInfo.ProjectChangedEventHandler(StaticInfo_
ProjectChanged);
         ViewModel.StaticInfo.User.UserCredentialsChanged
+= new
ViewModel.LoggedUser.UserCredentialsChangedEventHandler
(User_UserCredentialsChanged);
      }

      protected void PageUsers_Loaded(object sender,
EventArgs e)
      {
         lock (VModelSync)
         {
            IsLoaded = true;

            ResetViewData();
            ResetModel();
            InitializeModelHandlers();
            VModel.SetUserCredentials();

            VModel.ModelUpdateAsync(ViewModel.StaticInfo.Pro
ject);
         }
      }

      protected override void ResetModel()
      {
```

```
            VModel = new UsersViewModel();
        }

        protected override void ResetViewData()
        {
            _HasLastRowBeenEdited = false;
            usersdg.DataContext = null;
        }

        protected override void InitializeModelHandlers()
        {
            VModel.ModelUpdateCompleted += new
AbstractViewModel.ModelUpdateCompletedEventHandler(VMo
del_ModelUpdateCompleted);
        }

        void VModel_ModelUpdateCompleted(object sender)
        {
            lock (VModelSync)
            {
                if (sender != VModel)
                    return;
                UsersViewModel uvm = (UsersViewModel)VModel;

                serUsers.tblUsers NewDataUser = new
serUsers.tblUsers();
                Model.UserModel NewUser = new
Model.UserModel(NewDataUser);
                uvm.UsersList.Add(NewUser);
                _HasLastRowBeenEdited = false;
                usersdg.DataContext = uvm.UsersList;
                usersdg.DataSource = uvm.UsersList;
            }
        }

        private void user_delete_b(object sender,
RoutedEventArgs e)
        {
            lock (VModelSync)
            {
                UsersViewModel uvm = ((UsersViewModel)VModel);
                if ((usersdg.FocusedDataRow == null) ||
(usersdg.FocusedRowVisibleIndex == uvm.UsersList.Count -
1))
                    return;
                if (uvm.UsersList[usersdg.FocusedRowVisibleIndex -
1].Fullname.CompareTo(StaticInfo.User.User.flduserFullName)
== 0)
                    return;
```

```
            uvm.UsersList.RemoveAt(usersdg.FocusedRowVisibl
eIndex - 1);
         }
      }

      private void users_save_b(object sender,
RoutedEventArgs e)
      {
         lock (VModelSync)
         {
            UsersViewModel uvm = ((UsersViewModel)VModel);

            if (!_HasLastRowBeenEdited)
               uvm.UsersList.RemoveAt(uvm.UsersList.Count -
1);
            else
               _HasLastRowBeenEdited = false;
            uvm.SaveUsersList();
            uvm.UsersList.Add(new Model.UserModel(new
serUsers.tblUsers()));
         }
      }

      private void AgDataGridColumn_BeginEditing(object
sender, DevExpress.AgDataGrid.EditingEventArgs e)
      {
         lock (VModelSync)
            ((ComboBox)e.Editor).ItemsSource =
Model.UserModel.StaticRoles;
      }

      private void AgDataGridColumn_EndEditing(object
sender, DevExpress.AgDataGrid.EditingEventArgs e)
      {
         Model.UserModel SelectedUser =
usersdg.FocusedDataRow as Model.UserModel;
         string Role = ((ComboBox) e.Editor).SelectedItem as
string;
         if ((SelectedUser == null) || (Role == null))
            return;
         e.Value = Role;
      }
   }
}
```

Τα ViewModels των σελίδων κληρονομούν την abstract κλάση AbstractViewModel:

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace Isotrack3.ViewModel
{
   public abstract class AbstractViewModel
   {
      public enum RemoteServiceOperationEventType
      {
         Fail,
         Success,
         Timeout,
      };

      public delegate void
RemoteServiceOperationEventEventHandler(RemoteServiceOperatio
nEventType Type, string Description);
      public static event RemoteServiceOperationEventEventHandler
RemoteServiceOperationEvent;

      protected void
OnRemoteServiceOperationEvent(RemoteServiceOperationEventTyp
e Type, string Description)
      {
         if (RemoteServiceOperationEvent != null)
            RemoteServiceOperationEvent(Type, Description);
      }

      public delegate void
ModelUpdateCompletedEventHandler(object sender);
      public event ModelUpdateCompletedEventHandler
ModelUpdateCompleted;

      private object _SyncPendingUpdateOperations = new Object();
      private int __PendingUpdateOperations;
      private int _PendingUpdateOperations
      {
         get { lock (_SyncPendingUpdateOperations) return
__PendingUpdateOperations; }
```

```
            set { lock (_SyncPendingUpdateOperations)
__PendingUpdateOperations = value; }
        }

        protected int NumUpdateOperations { get; set; }

        protected virtual void UpdateOperationsCompeted()
        {
            ;
        }

        protected void DecPendingUpdateOperations()
        {
            if (_PendingUpdateOperations > 0)
                --_PendingUpdateOperations;
            if (_PendingUpdateOperations == 0)
            {
                UpdateOperationsCompeted();
                if (ModelUpdateCompleted != null)
                    ModelUpdateCompleted(this);
            }
        }

        virtual public void ModelUpdateAsync(serProj.tblE1 Project)
        {
            _PendingUpdateOperations = NumUpdateOperations;
        }

        abstract public void SetUserCredentials();
    }
}
```

Για παράδειγμα, το ViewModel της E43 σελίδας είναι το E43ViewModel:

```
using System;
using System.Collections.ObjectModel;

namespace Isotrack3.ViewModel
{
    public class E43ViewModel : AbstractViewModel
    {
        private serE43.svE43Client __SeE43 = new
serE43.svE43Client();
        public serE43.svE43Client _SeE43
        {
            get
```

```
            {
                return __SeE43;
            }
            set
            {
                __SeE43 = value;
            }
        }

        private serE2.svE2Client __SeE2 = new serE2.svE2Client();
        private serE2.svE2Client _SeE2
        {
            get
            {
                return __SeE2;
            }
            set
            {
                __SeE2 = value;
            }
        }

        public delegate void
E43YearsSaveCompletedEventsHandler(E43ViewModel Sender, bool
Result);
        public event E43YearsSaveCompletedEventsHandler
E43YearsSaveCompleted;

        private ObservableCollection<serE43.tblE43Years> _tblYears =
new ObservableCollection<serE43.tblE43Years>();
        private ObservableCollection<Model.E43YearsExtended>
_E43Years = new
ObservableCollection<Model.E43YearsExtended>();
        public ObservableCollection<Model.E43YearsExtended>
E43Years
        {
            get
            {
                return _E43Years;
            }
        }

        public E43ViewModel()
        {
            NumUpdateOperations = 2;
            InitializeHandlers();
        }

        private void InitializeHandlers()
```

```
        {
            _SeE43.GetE43YearsCompleted += new
EventHandler<serE43.GetE43YearsCompletedEventArgs>(_SeE43_G
etE43YearsCompleted);
            _SeE43.DeleteAllE43YearsCompleted += new
EventHandler<serE43.DeleteAllE43YearsCompletedEventArgs>(_Se
E43_DeleteAllE43YearsCompleted);
            _SeE43.SaveE43YearsListCompleted += new
EventHandler<serE43.SaveE43YearsListCompletedEventArgs>(_SeE
43_SaveE43YearsListCompleted);
            _SeE2.GetUsersCompleted += new
EventHandler<serE2.GetUsersCompletedEventArgs>(_SeE2_GetUse
rsCompleted);
        }

        protected override void UpdateOperationsCompeted()
        {
            base.UpdateOperationsCompeted();

            _E43Years.Clear();
            if (_tblYears != null)
                foreach (serE43.tblE43Years Year in _tblYears)
                    _E43Years.Add(new Model.E43YearsExtended(Year));
        }

        public override void SetUserCredentials()
        {
            if (_SeE2.ClientCredentials.UserName.UserName != null)
            {
                _SeE2.CloseAsync();
                _SeE43.CloseAsync();
                _SeE2 = new serE2.svE2Client();
                _SeE43 = new serE43.svE43Client();
                InitializeHandlers();
            }

            _SeE2.ClientCredentials.UserName.UserName =
StaticInfo.User.Username;
            _SeE2.ClientCredentials.UserName.Password =
StaticInfo.User.Password;
            _SeE43.ClientCredentials.UserName.UserName =
StaticInfo.User.Username;
            _SeE43.ClientCredentials.UserName.Password =
StaticInfo.User.Password;
        }

        void _SeE2_GetUsersCompleted(object sender,
serE2.GetUsersCompletedEventArgs e)
        {
```

```
            Model.E43YearsExtended.StaticProjectUsers = e.Result;
            DecPendingUpdateOperations();
        }


        void _SeE43_GetE43YearsCompleted(object sender,
serE43.GetE43YearsCompletedEventArgs e)
        {
            _tblYears = e.Result;
            DecPendingUpdateOperations();
        }

        void _SeE43_SaveE43YearsListCompleted(object sender,
serE43.SaveE43YearsListCompletedEventArgs e)
        {
            if (e.Result)
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Success, "τα E43Years αποθηκεύθηκαν");
            else
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Fail, "δεν ήταν δυνατή η αποθήκευση των E43Years");
        }

        void _SeE43_DeleteAllE43YearsCompleted(object sender,
serE43.DeleteAllE43YearsCompletedEventArgs e)
        {
            if (e.Result)
            {
                if (E43YearsSaveCompleted != null)
                    E43YearsSaveCompleted(this, true);
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Success, "τα E43Years αποθηκεύθηκαν");
            }
            else
            {
                if (E43YearsSaveCompleted != null)
                    E43YearsSaveCompleted(this, false);
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Fail, "δεν ήταν δυνατή η αποθήκευση των E43Years");
            }
        }

        public override void ModelUpdateAsync(serProj.tblE1 Project)
        {
            base.ModelUpdateAsync(Project);

            _SeE2.GetUsersAsync(Project.fldProjectID);
            _SeE43.GetE43YearsAsync(Project.fldProjectID);
        }
```

```
        public void SaveE43Years(long ProjectID)
        {
            if (E43Years.Count == 0)
                _SeE43.DeleteAllE43YearsAsync(ProjectID);
            else
            {
                ObservableCollection<serE43.tblE43Years> Years = new
ObservableCollection<serE43.tblE43Years>();
                foreach (Model.E43YearsExtended ExtendedYear in
E43Years)
                    Years.Add(ExtendedYear.Data);
                _SeE43.SaveE43YearsListAsync(ProjectID, Years);
            }
        }
    }
}
```

Η E43ViewModel μετατρέπει την λίστα αντικειμένων tblE43Years που τις δίνει ο εξυπηρετητής, σε λίστα αντικειμένων κλάσης κατάλληλης για χρήση από την PageE4_3:

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Collections.ObjectModel;

namespace Isotrack3.Model
{
    public class E43YearsExtended : Isotrack3.Model.Model
    {
        public static ObservableCollection<serE2.tblUsers>
StaticProjectUsers = new ObservableCollection<serE2.tblUsers>();

        public ObservableCollection<serE2.tblUsers> ProjectUsers
        {
            get
            {
                return StaticProjectUsers;
            }
```

```
}

public E43YearsExtended(serE43.tblE43Years Year)
{
   Data = Year;
   AdjustUserFullName();
}

public long E43YearUserID
{
   get
   {
      return Data.fldE4YearUserID;
   }
   set
   {
      Data.fldE4YearUserID = value;
      OnPropertyChanged("E43YearUserID");
   }
}

public long UserID
{
   get
   {
      return Data.fldUserID;
   }
   set
   {
      if (Data.fldUserID == value)
         return;
      Data.fldUserID = value;
      OnPropertyChanged("UserID");
      AdjustUserFullName();
   }
}

private void AdjustUserFullName()
{
   foreach (serE2.tblUsers User in ProjectUsers)
      if (User.fldUserID == Data.fldUserID)
      {
         _UserFullName = User.flduserFullName;
         OnPropertyChanged("UserFullName");
         return;
      }
}

public long ProjectID
```

```
{
   get
   {
      return Data.fldProjectID;
   }
   set
   {
      Data.fldProjectID = value;
      OnPropertyChanged("ProjectID");
   }
}

public long YearID
{
   get
   {
      return Data.fldYearID;
   }
   set
   {
      Data.fldYearID = value;
      OnPropertyChanged("YearID");
   }
}

public float JanMM
{
   get
   {
      return Data.fldMonthManDays1;
   }
   set
   {
      Data.fldMonthManDays1 = value;
      OnPropertyChanged("JanMM");
   }
}

public float FebMM
{
   get
   {
      return Data.fldMonthManDays2;
   }
   set
   {
      Data.fldMonthManDays2 = value;
      OnPropertyChanged("FebMM");
   }
```

```
}

public float MarMM
{
   get
   {
      return Data.fldMonthManDays3;
   }
   set
   {
      Data.fldMonthManDays3 = value;
      OnPropertyChanged("MarMM");
   }
}

public float AprMM
{
   get
   {
      return Data.fldMonthManDays4;
   }
   set
   {
      Data.fldMonthManDays4 = value;
      OnPropertyChanged("AprMM");
   }
}

public float MayMM
{
   get
   {
      return Data.fldMonthManDays5;
   }
   set
   {
      Data.fldMonthManDays5 = value;
      OnPropertyChanged("MayMM");
   }
}

public float JunMM
{
   get
   {
      return Data.fldMonthManDays6;
   }
   set
   {
```

```
            Data.fldMonthManDays6 = value;
            OnPropertyChanged("JunMM");
        }
    }

    public float JulMM
    {
        get
        {
            return Data.fldMonthManDays7;
        }
        set
        {
            Data.fldMonthManDays7 = value;
            OnPropertyChanged("JulMM");
        }
    }

    public float AugMM
    {
        get
        {
            return Data.fldMonthManDays8;
        }
        set
        {
            Data.fldMonthManDays8 = value;
            OnPropertyChanged("AugMM");
        }
    }

    public float SepMM
    {
        get
        {
            return Data.fldMonthManDays9;
        }
        set
        {
            Data.fldMonthManDays9 = value;
            OnPropertyChanged("SepMM");
        }
    }

    public float OctMM
    {
        get
        {
            return Data.fldMonthManDays10;
```

```
    }
    set
    {
        Data.fldMonthManDays10 = value;
        OnPropertyChanged("OctMM");
    }
}

public float NovMM
{
    get
    {
        return Data.fldMonthManDays11;
    }
    set
    {
        Data.fldMonthManDays11 = value;
        OnPropertyChanged("NovMM");
    }
}

public float DecMM
{
    get
    {
        return Data.fldMonthManDays12;
    }
    set
    {
        Data.fldMonthManDays12 = value;
        OnPropertyChanged("DecMM");
    }
}

private object _DataSync = new object();
private serE43.tblE43Years _Data;
public serE43.tblE43Years Data
{
    get
    {
        lock (_DataSync)
            return _Data;
    }
    set
    {
        lock (_DataSync)
        {
            _Data = value;
```

```
                Data.PropertyChanged += new
System.ComponentModel.PropertyChangedEventHandler(_Data_Pro
pertyChanged);
            }
         }
      }

      public void _Data_PropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
      {
         Data.PropertyChanged -= new
System.ComponentModel.PropertyChangedEventHandler(_Data_Pro
pertyChanged);
         if (Data.fldE4YearUserID != 0)
            Data.fldProjectID = 0;
      }

      private string _UserFullName = "";
      public string UserFullName
      {
         get
         {
            return _UserFullName;
         }
         set
         {
            _UserFullName = value;
            OnPropertyChanged("UserFullName");
            foreach (serE2.tblUsers User in ProjectUsers)
               if (String.Compare(User.flduserFullName, value) == 0)
               {
                  Data.fldUserID = User.fldUserID;
                  OnPropertyChanged("UserID");
                  return;
               }
         }
      }
   }
}
```

Υπάρχουν δύο αντικείμενα που είναι αποθηκευμένα στατικά ώστε να είναι διαθέσιμα σε όλες τις σελίδες: Ένα αντικείμενο που περιγράφει το τρέχον έργο και ένα αντικείμενο που δίνει πληροφορίες για τον χρήστη (όνομα, username και password, ρόλοι).

Το αντικείμενο του τρέχοντος έργου είναι απλώς αντικείμενο της κλάσης tblE1. Ο συνδεδεμένος χρήστης είναι αντικείμενο της κλάσης LoggedUser:

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.ComponentModel;
using System.Collections.ObjectModel;

namespace Isotrack3.ViewModel
{
   public class LoggedUser : AbstractViewModel
//       : INotifyPropertyChanged
   {
      private serUsers.svUsersClient __SerUsers = new
serUsers.svUsersClient();
      private serUsers.svUsersClient _SerUsers
      {
         get
         {
            return __SerUsers;
         }
         set
         {
            __SerUsers = value;
         }
      }

      private serSecurity.svSecurityClient __SerSec = new
serSecurity.svSecurityClient();
      private serSecurity.svSecurityClient _SerSec
      {
         get
         {
            return __SerSec;
         }
         set
         {
            __SerSec = value;
         }
      }

      public event PropertyChangedEventHandler PropertyChanged;
```

```
        public delegate void
AuthenticationSuccededEventHandler(object sender, bool
WasSuccess);
        public event AuthenticationSuccededEventHandler
AuthenticationSucceded;

        private string _Username = String.Empty;
        public string Username
        {
           get
           {
              return _Username;
           }
           set
           {
              _Username = value;
//            if (PropertyChanged != null)
//                PropertyChanged(this, new
PropertyChangedEventArgs("Username"));
           }
        }

        private string _Password = String.Empty;
        public string Password
        {
           get
           {
              return _Password;
           }
           set
           {
              _Password = value;
//            if (PropertyChanged != null)
//                PropertyChanged(this, new
PropertyChangedEventArgs("Password"));
           }
        }

        private object _SyncRoles = new object();

        private bool _DirectorAccess = false;
        public bool DirectorAccess
        {
           get
           {
              lock (_SyncRoles)
                 return _DirectorAccess;
           }
           set
```

```
         {
            lock (_SyncRoles)
            {
               _DirectorAccess = value;
//               if (PropertyChanged != null)
//                  PropertyChanged(this, new
PropertyChangedEventArgs("DirectorAccess"));
            }
         }
      }

      private bool _ManagerAccess = false;
      public bool ManagerAccess
      {
         get
         {
            lock (_SyncRoles)
               return _ManagerAccess;
         }
         set
         {
            lock (_SyncRoles)
            {
               _ManagerAccess = value;
//               if (PropertyChanged != null)
//                  PropertyChanged(this, new
PropertyChangedEventArgs("ManagerAccess"));
            }
         }
      }

      private bool _MemberAccess = false;
      public bool MemberAccess
      {
         get
         {
            lock (_SyncRoles)
               return _MemberAccess;
         }
         set
         {
            lock (_SyncRoles)
            {
               _MemberAccess = value;
 //               if (PropertyChanged != null)
 //                  PropertyChanged(this, new
PropertyChangedEventArgs("MemberAccess"));
            }
         }
```

```
        }

        private bool _Administrator = false;
        public bool Administrator
        {
           get
           {
              lock (_SyncRoles)
                 return _Administrator;
           }
           set
           {
              lock (_SyncRoles)
                 _Administrator = value;
           }
        }

        private bool _ProjectCreator = false;
        public bool ProjectCreator
        {
           get
           {
              lock (_SyncRoles)
                 return _ProjectCreator;
           }
           set
           {
              lock (_SyncRoles)
                 _ProjectCreator = value;
           }
        }

        public delegate void
UserUpdateCompletedEventHandler(object sender, bool Result);
        public event UserUpdateCompletedEventHandler
UserUpdateCompleted;

        public delegate void
UserCredentialsChangedEventHandler(object sender);
        public event UserCredentialsChangedEventHandler
UserCredentialsChanged;

        public delegate void
AccessInfoRetrieveCompletedEventHandler(object sender);
        public event AccessInfoRetrieveCompletedEventHandler
AccessInfoRetrieveCompleted;

        private object _SyncAccessInfo = new object();
        private short _AccessInfoPendingOperations = 0;
```

```
        private void _IncAccessInfoPendingOperations()
        {
            ++_AccessInfoPendingOperations;
        }
        private void _DecAccessInfoPendingOperations()
        {
            if ((--_AccessInfoPendingOperations == 0) &&
(AccessInfoRetrieveCompleted != null))
                AccessInfoRetrieveCompleted(this);
        }

        public serUsers.tblUsers User = new serUsers.tblUsers();

        public LoggedUser()
        {
            InitializeHandlers();
        }

        private void InitializeHandlers()
        {
            _SerSec.DirectorAccessInProjectCompleted += (s, e) =>
                {
                    DirectorAccess = e.Result;
                    _DecAccessInfoPendingOperations();
                };

            _SerSec.ManagerAccessInProjectCompleted += (s, e) =>
                {
                    ManagerAccess = e.Result;
                    _DecAccessInfoPendingOperations();
                };

            _SerSec.TeamMemberAccessInProjectCompleted += (s, e)
=>
                {
                    MemberAccess = e.Result;
                    _DecAccessInfoPendingOperations();
                };

            _SerUsers.UpdateUserCompleted += new
EventHandler<serUsers.UpdateUserCompletedEventArgs>(_SerUser
s_UpdateUserCompleted);
            _SerUsers.UserInfoCompleted += new
EventHandler<serUsers.UserInfoCompletedEventArgs>(_SerUsers_U
serInfoCompleted);
        }

        void _SerUsers_UserInfoCompleted(object sender,
serUsers.UserInfoCompletedEventArgs e)
```

```
        {
            if ((e.Result == null) || (e.Result.fldUserID == 0))
            {
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Fail, "δεν ήταν δυνατή η ταυτοποίηση του χρήστη");
                if (AuthenticationSucceded != null)
                    AuthenticationSucceded(this, false);
                return;
            }
            User = e.Result;
            var Urls = e.Result.tblUserRoles_Link;
            if ((Urls != null) && (Urls.Count != 0))
            {
                Administrator = (Urls[0].fldRoleID == 1);
                ProjectCreator = (Urls[0].fldRoleID < 3);
            }
            else
            {
                Administrator = false;
                ProjectCreator = false;
            }
            OnRemoteServiceOperationEvent(RemoteServiceOperationE
ventType.Success, "η ταυτοποίση του χρήστη έγινε με επιτυχία");
            if (AuthenticationSucceded != null)
                AuthenticationSucceded(this, true);
        }

        void _SerUsers_UpdateUserCompleted(object sender,
serUsers.UpdateUserCompletedEventArgs e)
        {
            if (e.Result)
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Success, "η αποθήκευση των στοιχείων του χρήστη
έγινε με επιτυχία");
            else
                OnRemoteServiceOperationEvent(RemoteServiceOperatio
nEventType.Fail, "δεν ήταν δυνατή η αποθήκευση των στοιχείων
του χρήστη");

            if (UserUpdateCompleted != null)
                UserUpdateCompleted(this, e.Result);
        }

        public void UserInfoAsync()
        {
            _SerUsers.UserInfoAsync();
        }

        public override void SetUserCredentials()
```

```
        {
            if (_SerUsers.ClientCredentials.UserName.UserName != null)
            {
                _SerSec.CloseAsync();
                _SerUsers.CloseAsync();
                _SerSec = new serSecurity.svSecurityClient();
                _SerUsers = new serUsers.svUsersClient();
                InitializeHandlers();
            }

            _SerSec.ClientCredentials.UserName.UserName =
Username;
            _SerSec.ClientCredentials.UserName.Password = Password;
            _SerUsers.ClientCredentials.UserName.UserName =
Username;
            _SerUsers.ClientCredentials.UserName.Password =
Password;
        }

        public void AccessInfoAsync(long ProjectID)
        {
            _IncAccessInfoPendingOperations();
            _IncAccessInfoPendingOperations();
            _IncAccessInfoPendingOperations();
            _SerSec.TeamMemberAccessInProjectAsync(ProjectID);
            _SerSec.DirectorAccessInProjectAsync(ProjectID);
            _SerSec.ManagerAccessInProjectAsync(ProjectID);
        }

        public void SaveUserInfo()
        {
            _SerUsers.UpdateUserAsync(User);
        }

        public void OnUserCredentialsChanged()
        {
            if (UserCredentialsChanged != null)
                UserCredentialsChanged(this);
        }
    }
}
```