



**ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΚΑΛΑΜΑΤΑΣ ΠΑΡΑΡΤΗΜΑ ΣΠΑΡΤΗΣ
ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**«MICROSOFT SYNC FRAMEWORK:
ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΠΡΟΤΥΠΗ ΥΛΟΠΟΙΗΣΗ»**



**ΕΙΣΗΓΗΤΗΣ:
ΤΡΙΤΑΚΗΣ ΠΑΝΑΓΙΩΤΗΣ**

**ΣΠΟΥΔΑΣΤΡΙΕΣ:
ΚΑΡΠΟΥΧΤΣΗ ΑΙΚΑΤΕΡΙΝΗ
ΦΩΤΙΑΔΟΥ ΕΛΕΝΗ**

ΣΠΑΡΤΗ 2012

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«MICROSOFT SYNC FRAMEWORK: ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΠΡΟΤΥΠΗ ΥΛΟΠΟΙΗΣΗ»



ΕΥΧΑΡΙΣΤΙΕΣ

Θέλουμε να ευχαριστήσουμε θερμά όλους όσους συμμετείχαν και συνέβαλαν στην ολοκλήρωση της παρούσας πτυχιακής εργασίας. Για την υλοποίηση αυτής της εργασίας, θεωρούμε υποχρέωσή μας να εκφράσουμε τις ειλικρινείς ευχαριστίες μας προς τον καθηγητή μας κ. Τριτάκη Παναγιώτη, ο οποίος αφ' ενός μεν μας έδωσε αυτή την ευκαιρία να ασχοληθούμε με ένα ενδιαφέρον θέμα και αφετέρου μας μετέδωσε τις απαραίτητες γνώσεις για να μπορέσουμε να την ολοκληρώσουμε επιτυχώς.

Επίσης, θα θέλαμε να ευχαριστήσουμε τις οικογένειές μας για την στήριξη που μας προσέφεραν σε όλη τη διάρκεια των σπουδών μας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ.....	11
---------------	----

ΚΕΦΑΛΑΙΟ 1^ο

ΜΟΝΤΕΛΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

1.1 Αρχιτεκτονική πελάτη / εξυπηρετητή	12
1.1.1 Εισαγωγή	12
1.1.2 Πλεονεκτήματα / Μειονεκτήματα αρχιτεκτονικής.....	19
1.2 Peer-to-peer.....	20
1.2.1 Ιστορική Αναδρομή	20
1.2.2 Μορφές Peer-to-Peer δικτύων	21
1.3 Ενημέρωση δεδομένων και συνδεσιμότητα	22

ΚΕΦΑΛΑΙΟ 2^ο

MICROSOFT SYNCHRONIZATION FRAMEWORK

2.1. Γενικά.....	24
2.2 Εισαγωγή στο Sync Framework Database Synchronization ...	24
2.3 Εισαγωγή στο Microsoft Synchronization framework	29
2.4 Participants.....	30
2.4.1 Participant Types.....	31

2.5 Λειτουργία του Microsoft Synchronization Framework.....	33
2.6 Ροή Synchronization.....	36
2.7 Παράδειγμα Synchronization	38
2.7.1 Παράδειγμα Σύγκρουσης	41
2.8 Τοπολογίες Multiple Synchronization.....	44
2.9 Τεχνικά Θέματα.....	46
2.9.1 Τεχνολογία Υλοποίησης.....	46
2.9.2 Επιλέγοντας ένα πρωτεύον κλειδί κατάλληλο για τον συγχρονισμό των πινάκων των βάσεων με την χρήση του Sync Framework	47
2.9.3 Στήλες αυτόματης-αύξησης (ταυτότητα) – Auto-Increment (Identity) Columns	48
2.9.4 GUIDs.....	49
2.9.5 Κλειδιά που περιλαμβάνουν ένα προσδιοριστικό κόμβων (Node Identifier).....	50

ΚΕΦΑΛΑΙΟ 3^ο

ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ

3.1 NET Framework.....	51
3.1.1 Ιστορική αναδρομή.....	53
3.1.2 Πλεονεκτήματα .NET Framework.....	54
3.1.3 Μειονεκτήματα .NET Framework (για εφαρμογές με μεγάλες απαιτήσεις από το υλικό).....	54
3.2 Microsoft Visual Studio 2008.....	55
3.2.1 Γενικά.....	55
3.2.2 Εισαγωγή στο Microsoft Visual Studio 2008.....	56

3.2.3 Περιβάλλον εργασίας του Microsoft Visual Studio 2008	57
3.3 Microsoft SQL Server 2008.....	63
3.3.1 Γενικά.....	63
3.3.2 Εισαγωγή στο Microsoft SQL Server 2008	64
3.3.3 Περιβάλλον εργασίας του Microsoft SQL Server 2008	65
3.4 Γλώσσα προγραμματισμού C#.....	69
3.4.1 Λόγοι Δημιουργίας.....	70
3.4.2 Γενικά Χαρακτηριστικά.....	71
3.4.3 Μειονεκτήματα	72
3.4.4 Υποστήριξη για Versioning.....	73
3.4.5 Web Programming.....	73

ΚΕΦΑΛΑΙΟ 4^ο

ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΑΣ

4.1 Πρόλογος εφαρμογής.....	75
4.2 Δημιουργία της βάσης δεδομένων.....	75
4.3 Δημιουργία της εφαρμογής.....	81
4.4 Εγχειρίδιο χρήσης.....	86
4.4.1 Βασικά κουμπιά της εφαρμογής.....	86
4.4.2 Επεξήγηση της λειτουργίας της εφαρμογής.....	87
4.5 Δημιουργία Query's	92
4.6 Ορισμός της απομακρυσμένης / εκσυγχρονιζόμενης βάσης	93

Παράρτημα Α: Κώδικας Εφαρμογής..... 96

ΒΙΒΛΙΟΓΡΑΦΙΑ 112

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Κλάσεις Πελάτη-Εξυπηρετητή	13
Εικόνα 2: Διάγραμμα Ακολουθίας Πελάτη-Εξυπηρετητή	13
Εικόνα 3: 2-tier αρχιτεκτονική πελάτη/εξυπηρετητή.....	14
Εικόνα 4: 3-tier αρχιτεκτονική πελάτη/εξυπηρετητή.....	15
Εικόνα 5: 3-tier αρχιτεκτονική πελάτη/εξυπηρετητή.....	16
Εικόνα 6: 1ο Επίπεδο, Database Server.....	17
Εικόνα 7: 2ο Επίπεδο, Application Server.....	17
Εικόνα 8: 3ο Επίπεδο, Client.....	18
Εικόνα 9: Sync Framework Database Synchronization.....	25
Εικόνα 10: Standalone Database Applications.....	27
Εικόνα 11: Remote Offices with Workgroup Servers	27
Εικόνα 12: Δίκτυο peer-to-peer	28
Εικόνα 13: Συγχρονισμός συσκευών	29
Εικόνα 14: Παράδειγμα αρχείων συγχρονισμού	30
Εικόνα 15: Full participants.....	31
Εικόνα 16: Partial Participants	32
Εικόνα 17: Simple Participants	33
Εικόνα 18: Συνεδρία Συγχρονισμού μεταξύ δυο παρόχων.....	34
Εικόνα 19: Ροή Συγχρονισμού	37
Εικόνα 20: Λειτουργία εταιρίας με μοντέλο πελάτη/ εξυπηρετητή.....	44
Εικόνα 21: Λειτουργία εταιρίας με μοντέλο peer-to-peer.....	45
Εικόνα 22: Τα components που υλοποιούν τον συγχρονισμό του τοπικού SQL Server και του απομακρυσμένου με χρήση Υπηρεσιών	47
Εικόνα 23: Παράδειγμα εκτέλεσης κώδικα μέσω του CLR	52
Εικόνα 24: Το .NET στοίβα πλαίσιο.....	53
Εικόνα 25: Αρχική σελίδα του Visual Studio	57
Εικόνα 26: New Project.....	58
Εικόνα 27: Περιβάλλον του Visual Studio.....	59

Εικόνα 28: Μπάρα menu και μπάρα εργαλείων του Visual Studio.....	59
Εικόνα 29: Toolbox Explorer.....	60
Εικόνα 30: Κενή winForm	61
Εικόνα 31: Το Solution Explorer.....	61
Εικόνα 32: Property Explorer	62
Εικόνα 33: Είσοδος στον SQL Server	65
Εικόνα 34: Περιβάλλον εργασίας του SQL Server	66
Εικόνα 35: Η γραμμή του menu με τη μπάρα εργαλείων.....	66
Εικόνα 36: Registered Servers	67
Εικόνα 37: Object Explorer	67
Εικόνα 38: Object Explorer Details	68
Εικόνα 39: Properties Explorer	69
Εικόνα 40. Δημιουργία νέας βάσης δεδομένων	76
Εικόνα 41. Συμπλήρωση ενός πίνακα	76
Εικόνα 42: Relationships των πινάκων	80
Εικόνα 43: Σύνδεση της βάσης δεδομένων με το Visual Studio.....	81
Εικόνα 44: Add Connection	82
Εικόνα 45: Data Source	83
Εικόνα 46: Εμφάνιση του Dataset	84
Εικόνα 47: WinForm Εμφάνιση Ατζέντας	85
Εικόνα 48: Αρίθμηση βασικών κουμπιών	86
Εικόνα 49: Αρχική φόρμα εισαγωγής βασικών στοιχείων στην ηλεκτρονική ατζέντα.....	87
Εικόνα 50: Add Contact	88
Εικόνα 51: Edit Contact	89
Εικόνα 52: Message box	89
Εικόνα 53: Φόρμα Κατηγοριών.....	90
Εικόνα 54: Φόρμα Συγχρονισμού	91
Εικόνα 55: Εκτελεσμένη Φόρμα Συγχρονισμού.....	91
Εικόνα 56: Application Settings.....	93
Εικόνα 57: Connection Properties.....	94

ΠΡΟΛΟΓΟΣ

Με την πάροδο του χρόνου η τεχνολογία και συγκεκριμένα ο κλάδος της πληροφορικής έχει αναπτυχθεί με ραγδαίους ρυθμούς. Στις μέρες μας είναι επιτακτική ανάγκη να μπορούμε να χρησιμοποιήσουμε τους υπολογιστές αντί για το μολύβι και το χαρτί. Για αυτό το λόγο δημιουργήσαμε αυτήν την εφαρμογή.

Όλοι έχουμε χρησιμοποιήσει κατά καιρούς ατζέντες για να φυλάμε στοιχεία ανθρώπων που μας είναι απαραίτητα. Η εφαρμογή αυτή λοιπόν είναι μια απλή ατζέντα χρήσιμη για έναν άνθρωπο ή και μια επιχείρηση στην οποία είναι καταχωρημένα αναλυτικά στοιχεία επαφών όπως προσωπικά δεδομένα και φωτογραφίες χωρισμένα σε κατηγορίες για μεγαλύτερη ευκολία.

Ωστόσο, η πτυχιακή μας βασίζεται στο συγχρονισμό βάσεων δεδομένων ανάμεσα σε πολλαπλές συσκευές. Ο συγχρονισμός επιτρέπει σε πολλαπλούς αποσυνδεδεμένους χρήστες να χρησιμοποιούν μια κοινή ατζέντα.

Η εφαρμογή αυτή αναπτύχθηκε στο περιβάλλον του Microsoft Visual Studio 2008 χρησιμοποιώντας ως γλώσσα προγραμματισμού την C#. Η επιλογή του περιβάλλοντος εργασίας και της γλώσσας προγραμματισμού έγινε με γνώμονα την ταχύτητα και την ευελιξία που μας προσφέρει η χρήση του .Net Framework.

ΚΕΦΑΛΑΙΟ 1^ο

ΜΟΝΤΕΛΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

Με την πάροδο των χρόνων στον κλάδο της πληροφορικής έχουμε παρατηρήσει μεγάλη ανάπτυξη. Στο κεφάλαιο αυτό θα αναφερθούμε στη δημιουργία μοντέλων επικοινωνίας. Σταδιακά, έχουν δημιουργηθεί ποικίλα μοντέλα που καθιστούν ευκολότερη την επικοινωνία. Στις επόμενες ενότητες θα κάνουμε αναφορά στην αρχιτεκτονική πελάτη / εξυπηρετητή και στο μοντέλο peer-to-peer.

1.1 Αρχιτεκτονική πελάτη / εξυπηρετητή

1.1.1 Εισαγωγή

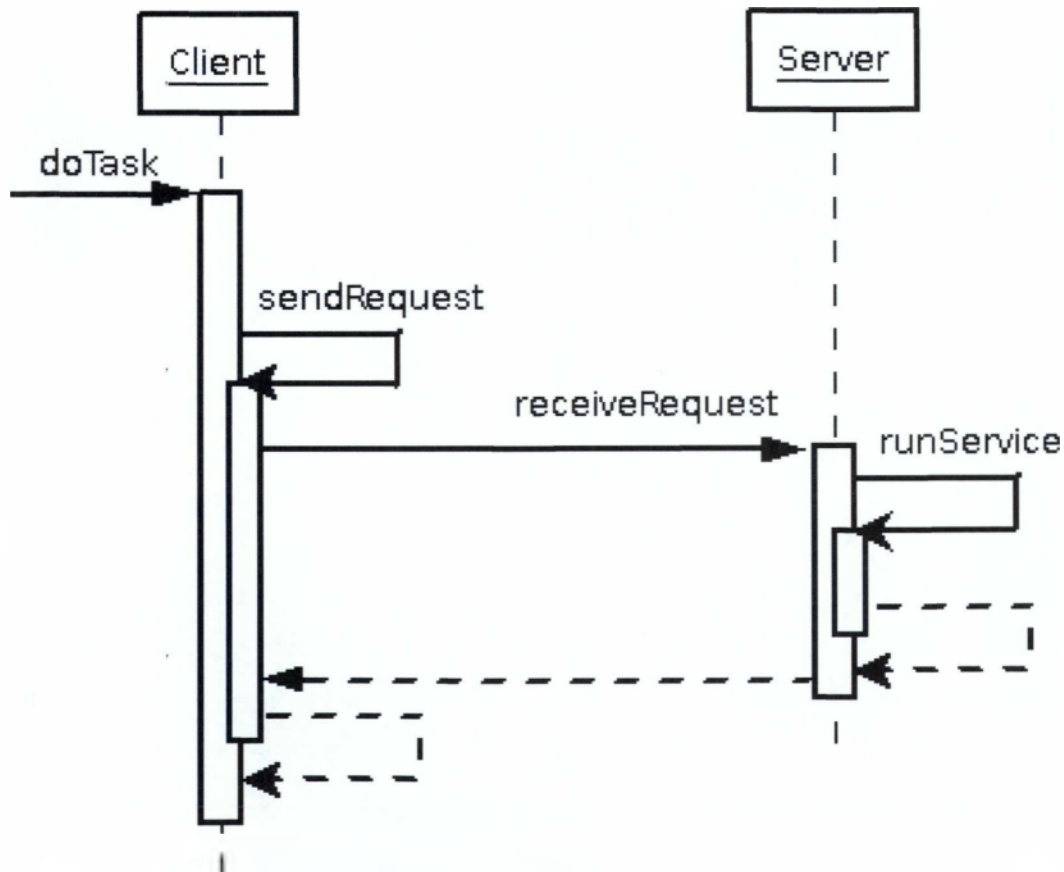
Η τεράστια ανάπτυξη του διαδικτύου την δεκαετία του '90 δημιούργησε ένα νέο πεδίο στην επιστήμη των υπολογιστών και κυρίως τη μηχανική λογισμικού, την ανάπτυξη διαδικτυακών εφαρμογών. Η ανάπτυξη διαδικτυακών εφαρμογών διαφέρει σε πολλά σημεία από την ανάπτυξη λογισμικού σε κλασικότερες μορφές, όπως εφαρμογών για χρήση σε desktop περιβάλλον. Οι διαφορές μιας διαδικτυακής σε σύγκριση με τις κλασικότερες μορφές οφείλονται κατά κύριο λόγο στο γεγονός ότι η πλατφόρμα εγκατάστασης και εφαρμογής της είναι το διαδίκτυο.

Το διαδίκτυο στηρίζεται στην αρχιτεκτονική πελάτη/εξυπηρετητή (client /server). Κάθε διαδικτυακή εφαρμογή λειτουργεί με βάση αυτό το πρότυπο. Ουσιαστικά ορίζει ότι τα συνεργαζόμενα κομμάτια μιας διαδικτυακής εφαρμογής μπορούν να κατηγοριοποιηθούν ως πελάτης και ως εξυπηρετητής. Ο πελάτης ζητά υπηρεσίες και δεδομένα από τον εξυπηρετητή και αυτός απαντάει στις αιτήσεις του. Στην περίπτωση που ένας web browser (π.χ. Mozilla Firefox, Internet Explorer)

επικοινωνεί με το web server χρησιμοποιώντας κάποιο πρωτόκολλο (HTTP πρωτόκολλο) κάνει κάποια αίτηση για δεδομένα ή υπηρεσία. Ο web server λαμβάνει αυτή την αίτηση του web browser και φροντίζει να την εξυπηρετήσει στέλνοντας τα δεδομένα (π.χ. στέλνοντας ένα HTML αρχείο στο browser) ή τρέχοντας κάποιο πρόγραμμα στον εξυπηρετητή και αποστέλλοντας τα αποτελέσματα κάποιας επεξεργασίας (π.χ. η άντληση δεδομένων από μια βάση δεδομένων).



Εικόνα 1: Κλάσεις Πελάτη-Εξυπηρετητή



Εικόνα 2: Διάγραμμα Ακολουθίας Πελάτη-Εξυπηρετητή

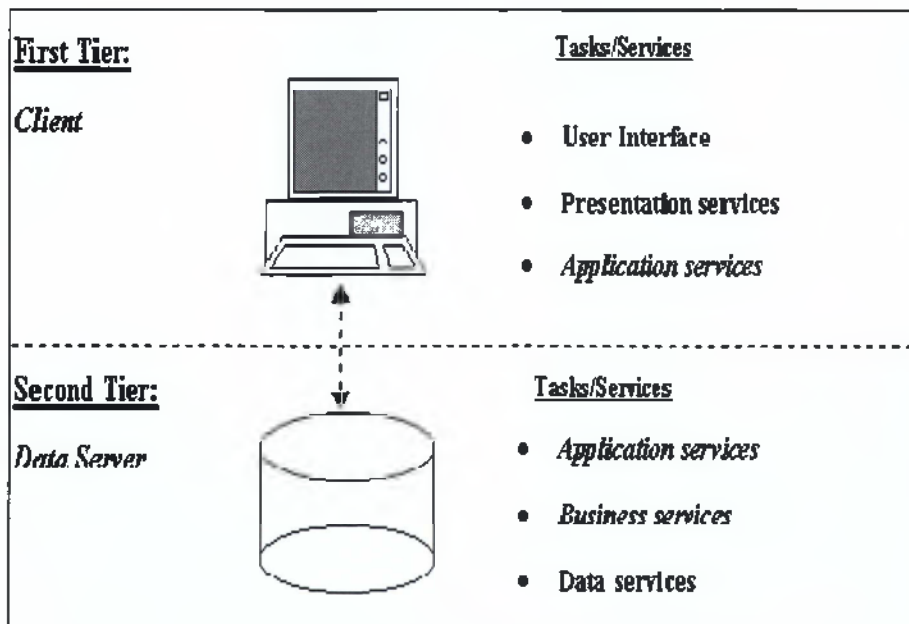
Η αρχιτεκτονική πελάτη-εξυπηρετητή συναντάται σε:

- I. Βάσεις δεδομένων
- II. Παγκόσμιος Ιστός (World Wide Web)
- III. Πληροφοριακά Συστήματα (Information Systems)

Παραλλαγές

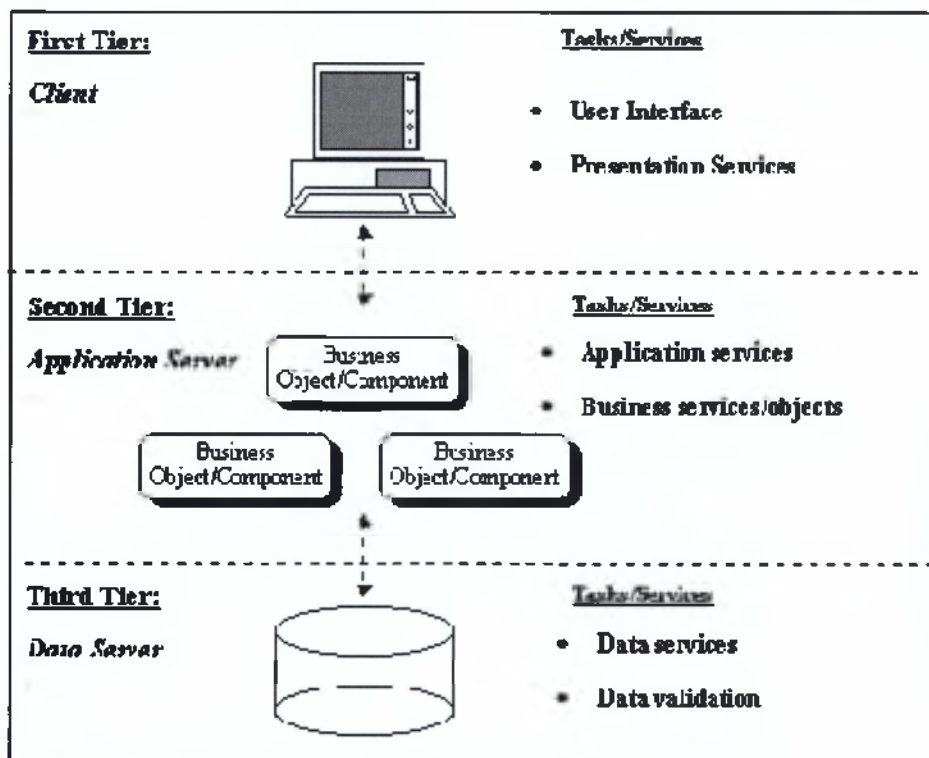
- I. *thin client*, όπου ο πελάτης είναι υπεύθυνος μόνο για την διεπαφή με τον χρήστη. Χαρακτηριστική περίπτωση όταν ο πελάτης χρησιμοποιεί το browser.
- II. *thick client*, όπου ο πελάτης είναι υπεύθυνος για την επιχειρησιακή λογική και για την διεπαφή με τον χρήστη, ενώ ο εξυπηρετητής είναι υπεύθυνος μόνο για την επεξεργασία δεδομένων.

Το απλό πρότυπο *client/server* όταν χρησιμοποιεί δύο επίπεδα που παίζουν κάποιο ρόλο στην λειτουργία ενός συστήματος ονομάζεται 2-tier αρχιτεκτονική (Εικόνα 1).



Εικόνα 3: 2-tier αρχιτεκτονική πελάτη/εξυπηρετητή

Η αρχιτεκτονική 2-tier σε μεγάλες εφαρμογές παρουσιάζει ορισμένα μειονεκτήματα. Υπάρχει η περίπτωση της υπερφόρτωσης του εξυπηρετητή από τις αιτήσεις του μεγάλου αριθμού πελατών που μπορεί να υπάρχουν. Επίσης, πιθανές αλλαγές στην επιχειρησιακή λογική που υλοποιεί μια εφαρμογή, απαιτούν υψηλό κόστος για αλλαγή σε όλη την σειρά διαδικτυακών εφαρμογών όπου αυτή υλοποιείται. Η αντιμετώπιση αυτών των μειονεκτημάτων γίνεται με την εφαρμογή 3-tier ή n-tier (Εικόνα 2) αρχιτεκτονικών όπου η υλοποίηση της επιχειρησιακής λογικής και η διαχείριση των δεδομένων γίνεται από δύο ή περισσότερα διαφορετικά συστατικά στοιχεία της διαδικτυακής εφαρμογής.

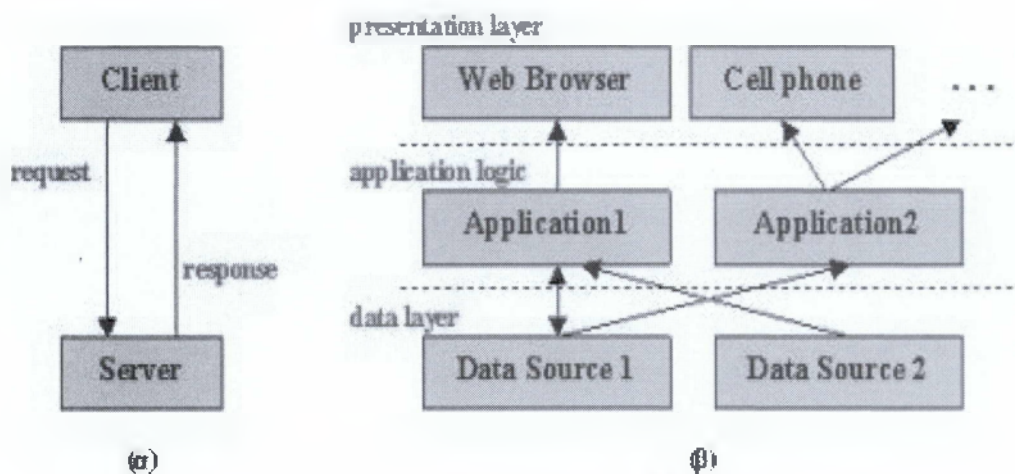


Εικόνα 4: 3-tier αρχιτεκτονική πελάτη/εξυπηρετητή

Σημείωση: Ο όρος ‘εφαρμογή n-tier’ είναι ένας τρόπος αναφοράς σε κατακευματισμένες εφαρμογές που χωρίζονται σε επίπεδα και ένας τύπος του και μάλλον ο πιο συχνός είναι ο 3-tier. Οι 3-tier είναι εφαρμογές τριών επιπέδων, αλλά εάν τα επίπεδα δεν είναι αρκετά, χωρίζονται σε περισσότερα. Εξίσου και ο όρος n-tier.

Η αρχιτεκτονική τριών επιπέδων (3-Tier Architecture) είναι μια αρχιτεκτονική Client-Server η οποία έχει ως χαρακτηριστικό της την διαμόρφωση των μερών εκείνων που αποτελούν την συνολική εφαρμογή σε επίπεδα (Layers ή αλλιώς Tiers). Οι έννοιες του στρώματος (layer) και της βαθμίδας (tier) συχνά χρησιμοποιούνται εναλλακτικά. Ωστόσο, μια αρκετά κοινή άποψη είναι ότι υπάρχει πράγματι μια διαφορά, και ότι ένα στρώμα είναι ένας λογικός μηχανισμός για τη διάρθρωση των στοιχείων που συνθέτουν τη λύση λογισμικού, ενώ μια βαθμίδα είναι ένας φυσικός μηχανισμός διάρθρωσης για την υποδομή του συστήματος.

Μια n-tier αρχιτεκτονική παρέχει ένα μοντέλο στους προγραμματιστές προκειμένου να δημιουργήσουν μια ευέλικτη και επαναχρησιμοποιήσιμη εφαρμογή. Με την διάσπαση μιας αίτησης σε κατηγορίες, οι προγραμματιστές πρέπει μόνο να τροποποιήσουν ή να προσθέσουν ένα συγκεκριμένο στρώμα, αντί να πρέπει να ξαναγράψουν το σύνολο της αίτησης.

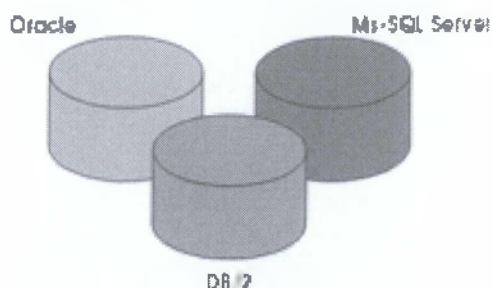


Εικόνα 5: 3-tier αρχιτεκτονική πελάτη/εξυπηρετητή

Συγκεκριμένα, τα επίπεδα αυτά όπως φαίνεται στην Εικόνα 2 είναι τα εξής:

1ο Επίπεδο – Data Layer – Database Server

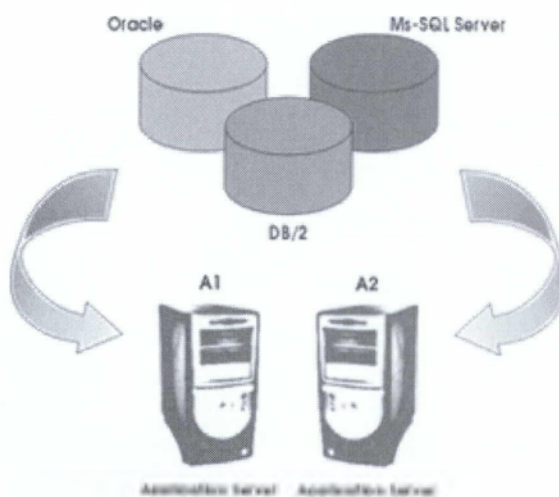
Αποτελώντας το βασικότερο επίπεδο του συστήματος, ο Database Server παρέχει όλες τις απαραίτητες λειτουργίες για την αποθήκευση, ανάκτηση, ενημέρωση και συντήρηση των δεδομένων του συστήματος καθώς επίσης και όλους τους απαραίτητους μηχανισμούς για την ακεραιότητα των δεδομένων (Data Integrity).



Εικόνα 6: 1ο Επίπεδο, Database Server

2ο Επίπεδο – Business Logic Layer – Application Server

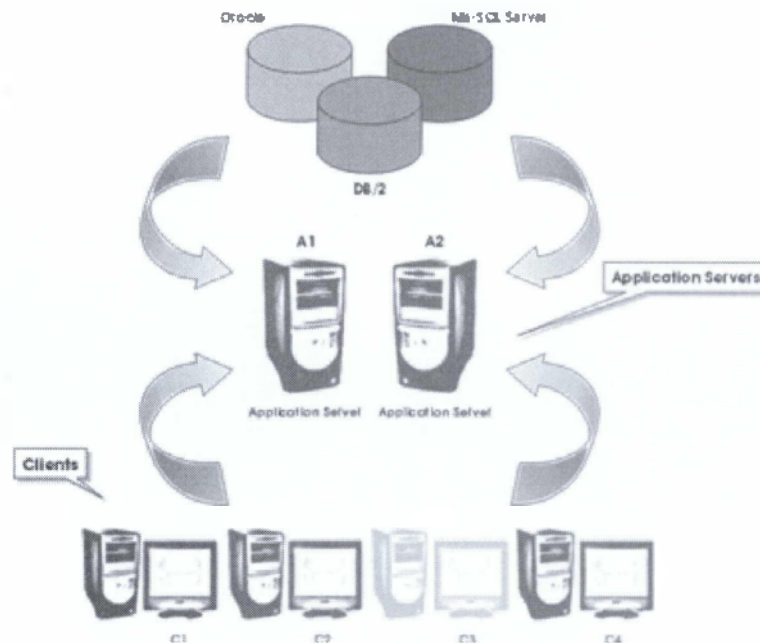
Αποτελεί το κύριο τμήμα του λογισμικού, Στο επίπεδο αυτό υλοποιούνται όλες εκείνες οι διαδικασίες που είναι υπεύθυνες για την διαχείριση των δεδομένων που φυλάσσονται στη βάση του 1ο επιπέδου καθώς και οι κανόνες που χαρακτηρίζουν την επιχειρηματική λογική της εφαρμογής. Υπάρχει δυνατότητα εγκατάστασης περισσότερων του ενός Application Servers σε διαφορετικά μηχανήματα, αξιοποιώντας, με τον τρόπο αυτό, οποιαδήποτε διαθέσιμη υπολογιστική ισχύ και εξασφαλίζοντας εξαιρετικά αποτελέσματα ανταπόκρισης, αξιοπιστίας και επεκτασιμότητας.



Εικόνα 7: 2ο Επίπεδο, Application Server

3ο Επίπεδο – Presentation Layer – User Interface

Το τρίτο επίπεδο του λογισμικού αποτελεί τη επαφή του χρήστη με το σύστημα (User Interface). Στο επίπεδο αυτό, πραγματοποιείται η διαχείριση των Οθονών Εργασίας (User Screens) καθώς επίσης και η μορφοποίηση των δεδομένων που εμφανίζονται. Η επικοινωνία του Client με τον Application ή τους Application Servers πραγματοποιείται κάνοντας χρήση ενός μόνο πακέτου δεδομένων κάθε φορά. Έτσι, επιτυγχάνεται ο βέλτιστος χρόνος απόκρισης μεταξύ του Client και του Application Server, δεδομένου ότι τα δυο αυτά επίπεδα μπορούν να λειτουργήσουν πάνω σε μια τηλεπικοινωνιακή γραμμή (Leased Line, Dialup, Internet Connection), εξασφαλίζοντας έτσι μικρούς χρόνους απόκρισης σε όλο το σύστημα.



Εικόνα 8: 3ο Επίπεδο, Client

Η συγκρότηση του συστήματος σε τρία επίπεδα εξασφαλίζει:

I. Την ελαχιστοποίηση της επιβάρυνσης του δικτύου λόγω μεταφοράς μεγάλου όγκου δεδομένων π.χ. η εκτέλεση ενός Query για την ανάκτηση μερικών εγγραφών από έναν πίνακα με δεκάδες χιλιάδες εγγραφές γίνεται στο διακομιστή εφαρμογής (Application Server), από τον οποίο μεταφέρεται στο χρήστη μόνο το αποτέλεσμα

II. Τη δυνατότητα διαχωρισμού του διακομιστή δεδομένων (Database Server) από το διακομιστή ή τούς διακομιστές εφαρμογής (Application Servers), ώστε να εκτελούνται σε διαφορετικά μηχανήματα. Κατά συνέπεια, ο καθορισμός των κρίσιμων μεγεθών απόδοσης των αντίστοιχων μηχανών (sizing) μπορεί να γίνεται ανεξάρτητα, ενώ παράλληλα εξασφαλίζεται απεριόριστη επεκτασιμότητα, χωρίς ανακατασκευή, του λογισμικού.

III. Τη μέγιστη ευελιξία στην επιλογή του διακομιστή δεδομένων, καθώς επιτρέπεται η χρήση οποιουδήποτε μηχανήματος με οποιοδήποτε λειτουργικό σύστημα (π.χ. Windows NT ή UNIX κλπ), με μοναδική απαίτηση τη δυνατότητα επικοινωνίας δια μέσου TCP/IP πρωτοκόλλου. Έτσι, είναι δυνατή η μεταγενέστερη αναβάθμιση ως προς τη βάση δεδομένων με την αλλαγή / αναβάθμιση του μηχανήματος, χωρίς να επηρεάζεται το υπόλοιπο σύστημα.

1.1.2 Πλεονεκτήματα / Μειονεκτήματα αρχιτεκτονικής

Όπως είναι φυσικό κάθε μοντέλο παρουσιάζει κάποια πλεονεκτήματα και κάποια μειονεκτήματα. Άλλα πιο σημαντικά και άλλα όχι.

Το μοντέλο πελάτη/εξυπηρετητή έχει αρκετά πλεονεκτήματα:

- I. Μεταφερισιμότητα (*portability*)
- II. Απόδοση (*performance*)
- III. Διαχειρισιμότητα (*ease of administration*)
- IV. Ικανότητα κλιμάκωσης (*scalability*)
- V. Προσαρμοστικότητα στη διεπαφή με τον χρήστη (*user interface flexibility*)
- VI. Αξιοπιστία (*reliability*)
- VII. Ταυτόχρονη διαχείριση βάσης σε πραγματικό χρόνο.

Εκτός από πλεονεκτήματα, υπάρχουν και μειονεκτήματα στην αρχιτεκτονική πελάτη/εξυπηρετητή:

- I. Υπάρχουν αυξημένες ανάγκες μεταφοράς δεδομένων μέσω δικτύου.
- II. Προβλήματα λόγω καθυστερήσεων στην μεταφορά δεδομένων.
- III. Είναι αναμφίβολα ένα δύσκολο προγραμματιστικό μοντέλο.

1.2 Peer-to-peer

Ένα δίκτυο υπολογιστών peer-to-peer (ή P2P) είναι ένα δίκτυο που επιτρέπει σε δύο ή περισσότερους υπολογιστές να μοιράζονται τους πόρους τους ισοδύναμα. Το δίκτυο αυτό χρησιμοποιεί την επεξεργαστική ισχύ, τον αποθηκευτικό χώρο και το εύρος ζώνης (bandwidth) των κόμβων. Όλοι οι κόμβοι του δικτύου έχουν τον ίδιο ρόλο. Πληροφορίες που βρίσκονται στον έναν κόμβο, ανάλογα με τα δικαιώματα που καθορίζονται, μπορούν να διαβαστούν από όλους τους άλλους και αντίστροφα.

1.2.1 Ιστορική Αναδρομή

Στις αρχές του 1999 ο Σον Φάνινγκ (Shawn Fanning) ξεκίνησε την υλοποίηση μιας ιδέας, η οποία θα του έδινε τη δυνατότητα αυτός και οι φίλοι του να αναζητήσουν στο Διαδίκτυο μουσικά κομμάτια MP3 της προτίμησής τους. Μερικούς μήνες αργότερα, η Napster μετρούσε πάνω από 21 εκατομμύρια χρήστες. Σε καμία περίπτωση, όμως, ο 18χρονος τότε μαθητής δεν μπορούσε να φανταστεί ότι το δημιούργημά του θα άλλαζε τον τρόπο με τον οποίο απολαμβάνουμε πολυμεσικές εφαρμογές και γενικά να επικοινωνούμε.

Η βασική ιδέα πίσω από το Napster ήταν η δημιουργία μιας εφαρμογής-πρωτοκόλλου, η οποία θα συνδύαζε μια μηχανή αναζήτησης, ενός προγράμματος

ανταλλαγής αρχείων βασισμένης στα πρωτόκολλα διαμοιρασμού αρχείων των Windows και του UNIX και ενός προγράμματος IRC, ώστε να είναι εφικτή η συζήτηση μεταξύ των χρηστών που βρισκόταν εκείνη τη στιγμή online. Το όνομα της εφαρμογής προήλθε από το παρατσούκλι του Φάνινγκ στο σχολείο λόγω του περιέργου κουρέματός του. Η εφαρμογή του Φάνινγκ έγινε νούμερο 1 στις προτιμήσεις των χρηστών στον Διαδικτυακό τόπο download.com και άνοιξε το δρόμο για την επανάσταση των δικτύων Peer-to-Peer, η οποία συνεχίζεται ως τις μέρες μας.

1.2.2 Μορφές Peer-to-Peer δικτύων

Τα Peer-to-Peer δίκτυα χωρίζονται σε τρεις κατηγορίες:

Συγκεντρωτικά P2P δίκτυα

Πολλοί, όταν αναφέρονται σε αυτά, χρησιμοποιούν τη φράση «πρώτης γενιάς P2P δίκτυα». Εδώ, υπάρχει ένας κεντρικός Index Server στον οποίο αποθηκεύονται οι πληροφορίες για τα περιεχόμενα των καταλόγων που οι συμμετέχοντες επιθυμούν να μοιράζονται. Οι χρήστες μπορούν να αναζητήσουν στους Index Servers αυτούς τα αρχεία που ψάχνουν, χρησιμοποιώντας ένα κατάλληλο πρόγραμμα-πελάτη. Όταν το αρχείο βρεθεί, ανοίγει μια σύνδεση μεταξύ των δύο χρηστών για τη μεταφορά του. Σε αυτή τη κατηγορία ανήκουν το Napster το DC++ και το WinMX.

Αποκεντρωτικά P2P δίκτυα

Η φιλοσοφία εδώ είναι εντελώς διαφορετική. Μόλις κάποιος συνδεθεί μέσω ενός ανάλογου προγράμματος-πελάτη P2P, κάνει γνωστή την παρουσία του σε ένα μικρό αριθμό υπολογιστών ήδη συνδεδεμένων οι οποίοι με τη σειρά τους προωθούν τη δήλωση παρουσίας του σε ένα μεγαλύτερο δίκτυο υπολογιστών κ.λ.π. Πλέον ο χρήστης έχει τη δυνατότητα να αναζητήσει οποιαδήποτε πληροφορία μεταξύ των διαμοιραζόμενων αρχείων. Τα δίκτυα αυτά λέγονται και δεύτερης γενιάς. Η μεταφορά των αρχείων είναι όμοια με αυτή των συγκεντρωτικών P2P δικτύων. Σε αυτή τη κατηγορία ανήκουν το Kazaa, το Gnutella και το BearShare.

P2P δίκτυα τρίτης γενιάς

Είναι αυτά τα οποία διαθέτουν χαρακτηριστικά ανωνυμίας όπως το Freenet, το I2P και το Entropy. Είναι αποκεντρωτικού τύπου και η φιλοσοφία του βασίζεται εκτός από την ανωνυμία, στην υψηλή βιωσιμότητα του, στο συνεχή διαμοιρασμό των αρχείων και στην κωδικοποίησή τους έτσι ώστε κανείς να μην μπορέσει ποτέ να αποκτήσει κανένα είδος ελέγχου πάνω σε αυτό. Τα δίκτυα αυτού του τύπου είναι υπό ανάπτυξη και έχουν χαρακτηριστεί ως μικρά παγκόσμια δίκτυα.

1.3 Ενημέρωση δεδομένων και συνδεσιμότητα

Στα σύγχρονα συστήματα είναι πολύ συχνό το φαινόμενο μια εφαρμογή να χρησιμοποιείται ταυτόχρονα από πολλαπλούς χρήστες (ή από ένα χρήστη ο οποίος όμως την προσπελαύνει από διαφορετικές συσκευές). Σε τέτοια σενάρια χρήσης η αρχιτεκτονική πελάτη-εξυπηρετητή παρουσιάζει ένα πολύ σημαντικό πλεονέκτημα. Καθώς όλα τα δεδομένα αποθηκεύονται κεντρικά (Data Layer), ενώ επίσης κεντρικά μπορεί να υλοποιείται και μεγάλο μέρος της λογικής του συστήματος (Application Layer), όλοι οι χρήστες έχουν πρόσβαση στα ίδια ακριβώς δεδομένα και στις ίδιες λειτουργίες. Όταν ένας χρήστης κάνει μια αλλαγή, αυτή η αλλαγή είναι αυτόματα ορατή σε όλους τους υπόλοιπους χρήστες.

Το παραπάνω χαρακτηριστικό βέβαια, προϋποθέτει ότι όλοι οι χρήστες είναι διαρκώς συνδεδεμένοι με τον κεντρικό εξυπηρετητή. Στα κλασικά σενάρια εργασίας (π.χ. ενός γραφείου) κάτι τέτοιο δεν αποτελεί πρόβλημα. Ωστόσο, υπάρχουν περιπτώσεις (π.χ. πωλητές ή οδηγοί) όπου οι χρήστες μετακινούνται συνέχεια και άρα είναι πρακτικά αδύνατο να είναι διαρκώς συνδεδεμένοι με τον κεντρικό εξυπηρετητή. Τέτοιου είδους σενάρια εμφανίζονται ολοένα και συχνότερα με την αυξανόμενη τάση για απομακρυσμένη εργασία.

Προκειμένου να είναι δυνατόν να χρησιμοποιηθεί η εφαρμογή σε τέτοιου είδους περιβάλλοντα θα πρέπει:

- Να μπορεί να λειτουργεί αυτόνομα (χωρίς συνδεσιμότητα με τον κεντρικό εξυπηρετητή).
- Να υπάρχει κάποια διαδικασία ενημέρωσης των δεδομένων.

Το πρώτο από τα δύο αυτά σημεία είναι σχετικά απλό να υλοποιηθεί ενσωματώνοντας τον εξυπηρετητή (ή μια πιο ελαφριά έκδοσή του) στη συσκευή του απομακρυσμένου χρήστη. Το δεύτερο απαιτεί την ανάπτυξη διεργασιών (πρότυπα, πρωτόκολλα, κώδικας) συγχρονισμού ανάμεσα στον κεντρικό εξυπηρετητή και τον εξυπηρετητή που υπάρχει στη συσκευή του απομακρυσμένου χρήστη.

Έτσι, ένας απομακρυσμένος χρήστης μπορεί να χρησιμοποιεί την εφαρμογή χωρίς συνδεσιμότητα. Όταν η συνδεσιμότητα αποκατασταθεί (π.χ. βρεθεί εντός δικτύου) ενημερώνει τον κεντρικό εξυπηρετητή για τις αλλαγές που έκανε κι ενημερώνεται για τις αλλαγές που πιθανόν να έγιναν από άλλους χρήστες.

ΚΕΦΑΛΑΙΟ 2^ο

MICROSOFT SYNCHRONIZATION FRAMEWORK

2.1. Γενικά

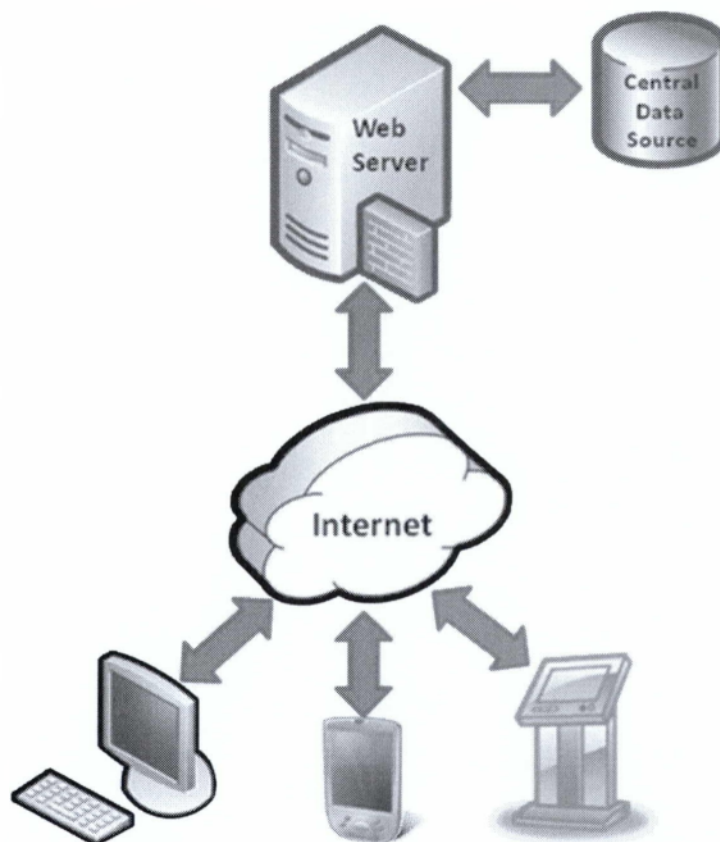
Το Sync Framework ή Υπηρεσίες Συγχρονισμού, μας δίνει την δυνατότητα να συγχρονίσουμε δεδομένα μεταξύ δυο όμοιων βάσεων δεδομένων (και όχι μόνο) με χρήση 2-tier, n-tier αρχιτεκτονικής ή με την χρήση Υπηρεσιών. Το Application Programming Interface (Προγραμματιστική Διασύνδεση) των Synchronization Services παρέχει ένα σετ από κλάσεις που δίνουν την δυνατότητα συγχρονισμού δεδομένων με χρήση των υπηρεσιών συγχρονισμού και μιας τοπικής βάσης αντί απλά να αντιγράφουν τα δεδομένα και το σχήμα μίας βάσης.

Όσον αφορά την εφαρμογή, ο χρήστης μπορεί να έχει πρόσβαση στα δεδομένα και να λειτουργεί το πρόγραμμα, ακόμα και όταν δεν υπάρχει διαθέσιμη σύνδεση στο τοπικό δίκτυο ή σύνδεση μέσω Internet.

Με τη χρήση των υπηρεσιών συγχρονισμού της Microsoft, δίνεται η δυνατότητα, όταν κάποιος βρίσκεται σε σύνδεση να συγχρονίζει τα δεδομένα της τοπικής βάσης του SQL Server Compact Edition με αυτά του κεντρικού SQL Server. Μπορεί να προβεί σε εισαγωγές, μεταβολές και διαγραφές στην τοπική βάση ενώ δεν υπάρχει σύνδεση και αργότερα πάλι όταν βρεθεί σε σύνδεση να προβεί σε έναν ακόμη συγχρονισμό και να ενημερώσει τις δύο βάσεις για τα πιο πρόσφατα δεδομένα.

2.2 Εισαγωγή στο Sync Framework Database Synchronization

Η ικανότητα να υποστηρίζουν κινητούς και απομακρυσμένους εργαζόμενους γίνεται όλο και περισσότερο σημαντική για τους οργανισμούς κάθε μέρα. Είναι σημαντικό ότι οι οργανισμοί διαβεβαιώνουν ότι οι χρήστες έχουν πρόσβαση στις ίδιες πληροφορίες που έχουν και όταν βρίσκονται στο γραφείο τους. Στις περισσότερες περιπτώσεις αυτοί οι εργαζόμενοι θα έχουν ένα είδους φορητό υπολογιστή, υπολογιστή γραφείου, ένα smartphone ή ένα PDA. Από αυτές τις συσκευές οι χρήστες θα είναι ικανοί να έχουν πρόσβαση στα δεδομένα τους απευθείας μέσω VPN συνδέσεων, Web servers ή κάποιου άλλου είδους μέθοδο σύνδεσης μέσα από το εταιρικό δίκτυο.



Εικόνα 9: Sync Framework Database Synchronization

Μερικά μειονεκτήματα που προκύπτουν σε περιπτώσεις όπου οι εργαζόμενοι δουλεύουν από απόσταση με μια εταιρία είναι τα ακόλουθα:

I. Οι απαιτήσεις δικτύου: Προκειμένου οι χρήστες να έχουν πρόσβαση στις πληροφορίες που τους ενδιαφέρουν, η κεντρική βάση πρέπει να έχει μια σταθερή

σύνδεση με το δίκτυο της εταιρίας και παράλληλα να έχουν πρόσβαση στα δεδομένα τους. Για κάποιους εργαζόμενους, όπως αυτοί που εργάζονται από το σπίτι, η έλλειψη πρόσβασης στο εταιρικό δίκτυο μπορεί να μην είναι πρόβλημα. Για τους εργαζόμενους όμως που είναι συνεχώς σε κίνηση, ενδεχομένως η διαδικασία αυτή μπορεί να είναι αρκετά περίπλοκη.

II. Ταχύτερη πρόσβαση στα δεδομένα: Σε ένα τυπικό μοντέλο πελάτη / εξυπηρετητή που υπάρχει σε μια εταιρία οι χρήστες έχουν πρόσβαση σε δίκτυα με υψηλή ταχύτητα που τους επιτρέπουν γρήγορη πρόσβαση σε όλες τις πληροφορίες. Όσο για τους εργαζόμενους που δουλεύουν από απόσταση, η σύνδεση τους είναι πιο αργή, εφόσον έχουν αναξιόπιστα ενσύρματα ή ασύρματα δίκτυα. Το γεγονός αυτό, αναγκάζει τους χρήστες να κατεβάζουν κάθε φορά ότι πληροφορία χρειάζονται, διότι δεν υπάρχει τρόπος να παραμένουν τα δεδομένα στη συσκευή.

III. Μοναδικό σημείο αποτυχίας: Σε αυτό το σημείο, όλοι οι χρήστες εξαρτώνται από έναν μόνο server. Αν η βάση δεδομένων καταστεί μη διαθέσιμη λόγω προγραμματισμένης διακοπής λειτουργίας του server ή λόγω του εκάστοτε προβλήματος που μπορεί να προκύψει, τότε όλοι οι εργαζόμενοι από απόσταση θα αποσυνδεθούν αυτόματα από τα δεδομένα τους.

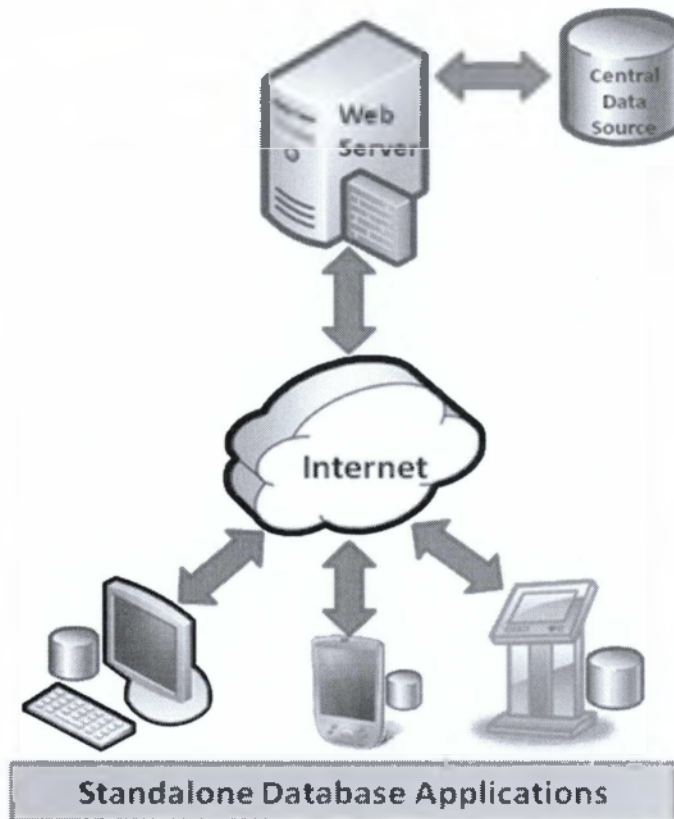
IV. Επεκτασιμότητα του server: Καθώς όλο και περισσότεροι εργαζόμενοι εργάζονται εξ' αποστάσεως, η απόδοση των servers της εταιρίας θα επηρεαστεί, με αποτέλεσμα να πρέπει να προσθέσουμε επιπλέον hardware.

Τα επόμενα διαγράμματα (εικόνα 10 και 11) απεικονίζουν μια εταιρία, όπου τα δεδομένα της αντιπροσωπεύονται από μια βάση δεδομένων και μεταφέρονται τοπικά στη συσκευή του απομακρυσμένου εργαζόμενου.

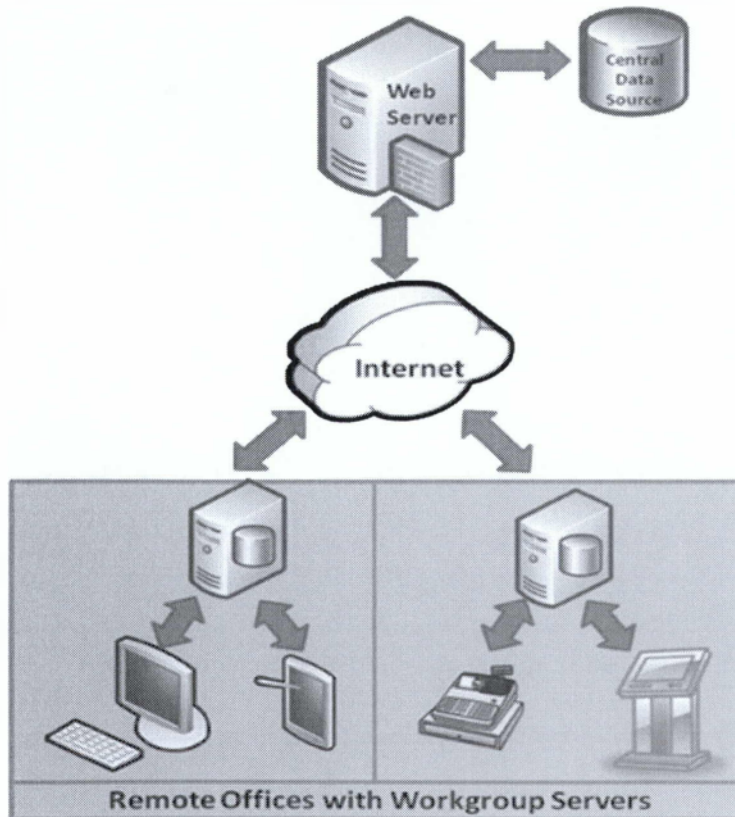
I. Στην εικόνα 10 η κεντρική βάση δεδομένων δεν είναι προσβάσιμη και υπάρχει ένα αυτόνομο σύστημα βάσης δεδομένων, όπου οι πληροφορίες αποθηκεύονται απευθείας στη συσκευή του εργαζόμενου.

II. Στην εικόνα 11 υπάρχει μια βάση δεδομένων σε κάθε απομακρυσμένο γραφείο, έτσι ώστε οι εργαζόμενοι να μπορούν να έχουν πρόσβαση στα δεδομένα.

Γενικά, και στις δύο περιπτώσεις πρέπει να υπάρχει τρόπος συγχρονισμού με την απομακρυσμένη βάση δεδομένων.

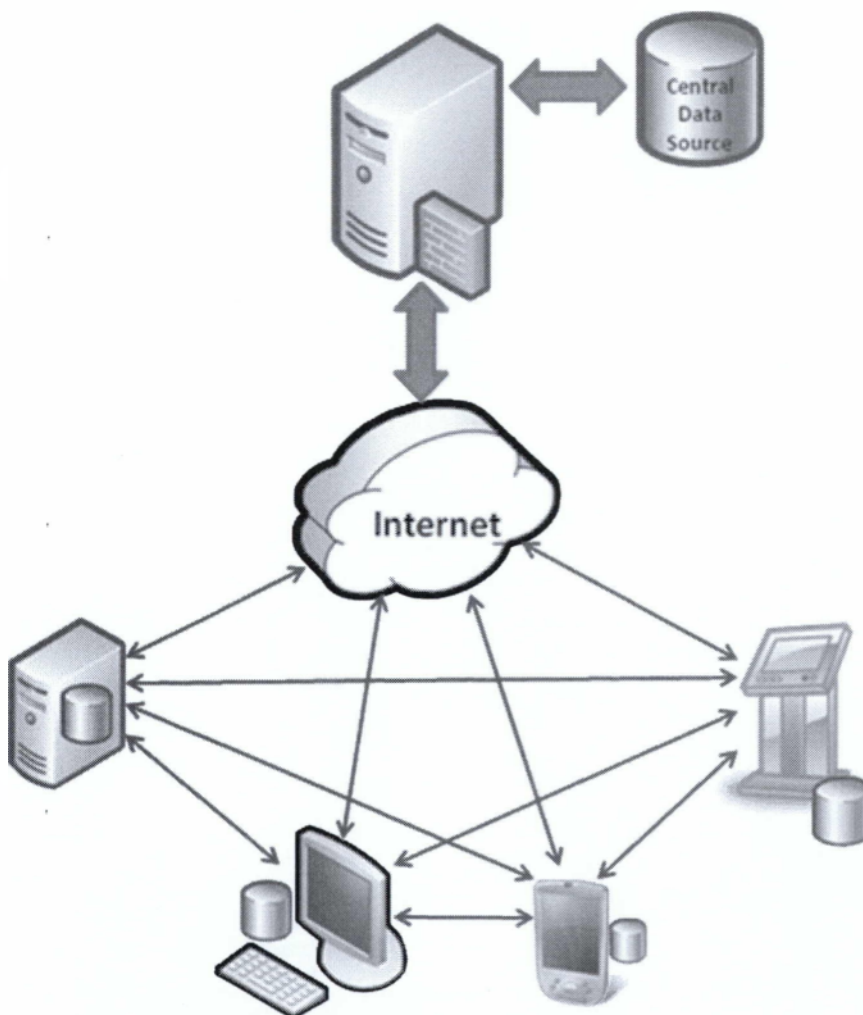


Εικόνα 10: Standalone Database Applications



Εικόνα 11: Remote Offices with Workgroup Servers

Όπως παρατηρούμε υπάρχει η δυνατότητα να υποστηριχτεί η συνεργασία των δεδομένων μεταξύ βάσεων δεδομένων. Όπως φαίνεται παρακάτω, μια απομακρυσμένη βάση δεδομένων είναι ελεύθερη να ανταλλάξει πληροφορίες με οποιαδήποτε άλλη βάση δεδομένων. Αυτός ο τρόπος λύσης είναι χρήσιμος για μια ομάδα ανθρώπων που εργάζονται σε απομακρυσμένες περιοχές και δεν έχουν πρόσβαση σε μια κεντρική βάση δεδομένων. Οι εργαζόμενοι αυτοί πρέπει συχνά να μοιράζονται πληροφορίες μεταξύ τους, αλλά δεδομένου ότι δεν έχουν σύνδεση με την κεντρική βάση δεδομένων είναι απαραίτητο να μοιράζονται πληροφορίες μέσω του δίκτυο peer-to-peer.



Εικόνα 12: Δίκτυο peer-to-peer

2.3 Εισαγωγή στο Microsoft Synchronization framework

Το Microsoft Sync Framework είναι μια πλατφόρμα συγχρονισμού που επιτρέπει την συνεργασία υπηρεσιών, συσκευών και εφαρμογών. Οι προγραμματιστές μπορούν να δημιουργήσουν συγχρονισμένα συστήματα που ενσωματώνονται σε κάθε εφαρμογή. Μπορούν να χρησιμοποιούν οποιοδήποτε δεδομένο απαιτείται κάθε φορά χρησιμοποιώντας το κατάλληλο πρωτόκολλο απ' όλα τα δίκτυο. Τα χαρακτηριστικά του Sync Framework προσαρμόζονται σε τεχνολογίες και εργαλεία που επιτρέπουν το roaming, την μεταφορά και την λήψη δεδομένων ενώ είναι offline.

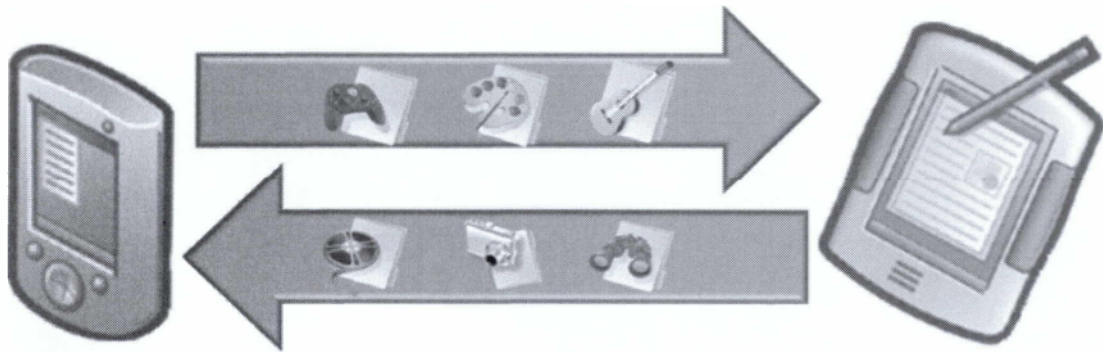
Μια βασική πτυχή του Sync Framework είναι η δυνατότητα να δημιουργεί προσαρμοσμένους παρόχους. Οι πάροχοι επιτρέπουν σε οποιοσδήποτε πηγές δεδομένων να συμμετέχουν στην διαδικασία συγχρονισμού στο Sync Framework, επιτρέποντας την υλοποίηση συγχρονισμού peer-to-peer ανάμεσα σε οποιοδήποτε τύπου δεδομένα (files, database, κ.λπ.) και σε οποιοδήποτε είδους συσκευές (notebooks, smart phones, ...).



Εικόνα 13: Συγχρονισμός συσκευών

Ένας αριθμός έτοιμων παρόχων συμπεριλαμβάνονται στο Sync Framework που υποστηρίζει πολλές κοινές πηγές δεδομένων. Αν και δεν απαιτείται, προτείνεται οι προγραμματιστές να χρησιμοποιούν αυτούς τους παρόχους όποτε είναι δυνατόν. Παρακάτω αναφέρονται αυτοί οι πάροχοι:

- I. Πάροχοι συγχρονισμού βάσης: Συγχρονισμός για ADO.NET που επιτρέπει τις πηγές βάσεων δεδομένων.
- II. Πάροχος συγχρονισμού αρχείων: Συγχρονισμός για αρχεία και φακέλους.
- III. Web συστατικά συγχρονισμού: Συγχρονισμός για FeedSync feeds, όπως RSS και Atom.



Εικόνα 14: Παράδειγμα αρχείων συγχρονισμού

Οι προγραμματιστές μπορούν να χρησιμοποιήσουν είτε έτοιμους είτε προσαρμοσμένους παρόχους προκειμένου να ανταλλάσουν πληροφορίες ανάμεσα σε συσκευές και εφαρμογές.

2.4 Participants (Συμμετέχοντες)

Βασιζόμενο στις δυνατότητες της συσκευής, ο τρόπος που ένας πάροχος ενσωματώνει τον συγχρονισμό ποικίλει.

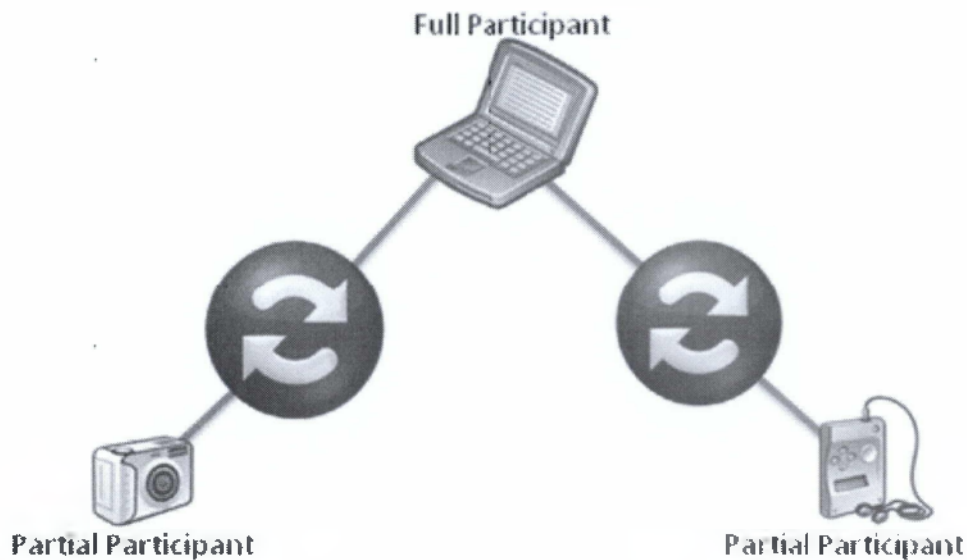
2.4.1 Participant Types (Τύποι Συμμετεχόντων)

I. Full participants: Είναι συσκευές που επιτρέπουν στους προγραμματιστές να δημιουργήσουν εφαρμογές και τα νέα δεδομένα αποθηκεύονται απευθείας στην συσκευή. Ένας φορητός υπολογιστής ή ένα smartphone είναι παραδείγματα full participants, επειδή οι νέες εφαρμογές μπορούν να εκτελεστούν απευθείας από την συσκευή και μπορούν επίσης να δημιουργήσουν νέα δεδομένα αποθήκευσης για να διατηρήσουν πληροφορίες εάν απαιτείται.



Εικόνα 15: Full participants

II. Partial Participants: Είναι συσκευές που έχουν την ικανότητα να αποθηκεύουν δεδομένα είτε σε έναν ήδη υπάρχοντα χώρο αποθήκευσης, είτε σε έναν άλλο χώρο αποθήκευσης που θα δημιουργήσουν στη συσκευή. Μερικά παραδείγματα αυτών των participants είναι thumb drives ή SD κάρτες. Αυτές οι συσκευές δρουν όπως ένας σκληρός δίσκος όπου οι πληροφορίες μπορούν να δημιουργηθούν, να ενημερωθούν και να διαγραφούν. Παρόλα αυτά, αυτοί δεν δίνουν μια διεπαφή που επιτρέπει στις εφαρμογές να εκτελούνται απ' αυτές απευθείας.



Εικόνα 16: Partial Participants

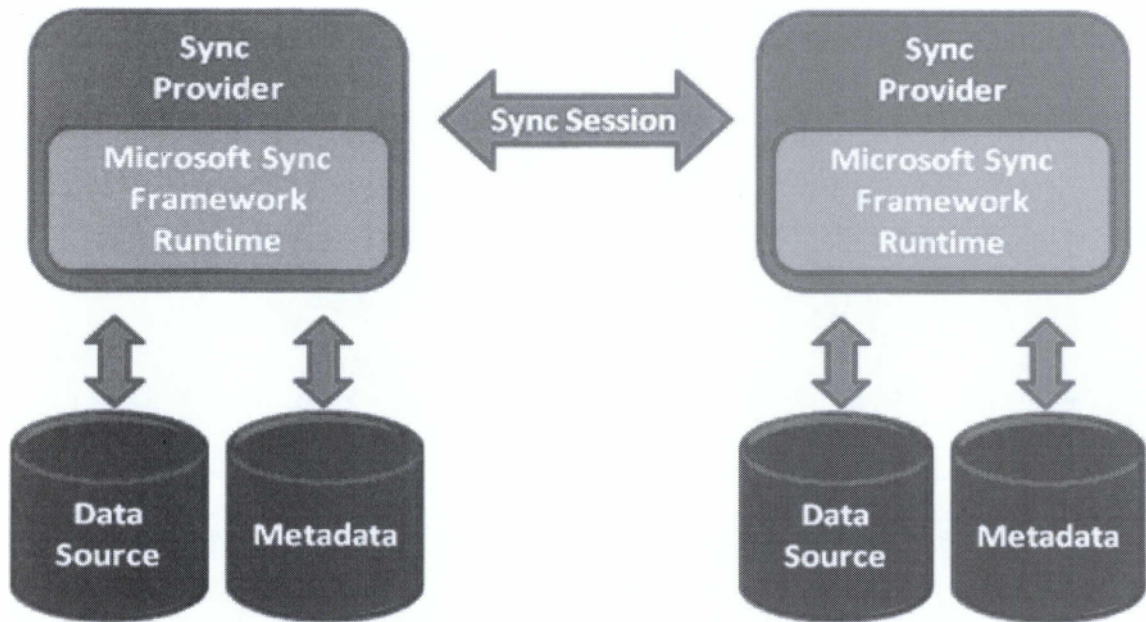
III. Simple Participants: Είναι συσκευές που είναι ικανές μόνο για να παρέχουν πληροφορίες όταν τους ζητούνται. Αυτές οι συσκευές δεν μπορούν να αποθηκεύσουν ή να διαχειριστούν νέα δεδομένα και δεν είναι ικανές να υποστηρίξουν την δημιουργία νέων εφαρμογών. RSS feeds και web υπηρεσίες που παρέχονται από έναν εξωτερικό οργανισμό, όπως το Amazon ή το Ebay είναι και τα δυο παραδείγματα simple participants. Αυτοί οι οργανισμοί δίνουν την δυνατότητα να εκτελούνται web υπηρεσίες. Όμως όπως είναι αναμενόμενο δεν επιτρέπουν να δημιουργήσει ο κάθε χρήστης το δικό του χώρο αποθήκευσης δεδομένων και επίσης δεν δίνουν την δυνατότητα να δημιουργεί ο καθένας τις δικές του εκτελέσιμες εφαρμογές μέσα από τους δικούς του web servers.



Εικόνα 17: Simple Participants

2.5 Λειτουργία του Microsoft Synchronization Framework

Πριν να εκτελεστεί ο συγχρονισμός και χρησιμοποιηθεί το Sync Framework πρέπει να κατανοήσουμε πρώτα τα βασικά στοιχεία ενός provider. Το παρακάτω διάγραμμα απεικονίζει πώς ένας συμμετέχοντας επικοινωνεί με μια πηγή δεδομένων και με τα μεταδεδομένα, δηλαδή τις πληροφορίες του κάθε αρχείου για μια κατάσταση. Αυτοί οι providers με τη σειρά τους επικοινωνούν με άλλους providers μέσω της συνεδρίας συγχρονισμού.



Εικόνα 18: Συνεδρία Συγχρονισμού μεταξύ δυο παρόχων

Η πηγή δεδομένων περιλαμβάνει όλα τα δεδομένα που χρειαζόμαστε για το συγχρονισμό. Προκειμένου όμως να είναι δυνατός ο συγχρονισμός, το Sync Framework χρειάζεται την ύπαρξη και πληροφοριών για τα δεδομένα. Μια πηγή δεδομένων θα μπορούσε να είναι μια σχεσιακή βάση δεδομένων, ένα αρχείο, μια Web υπηρεσία ή ακόμα και μια προσαρμοσμένη πηγή δεδομένων που περιλαμβάνεται σε μια επιχειρηματική εφαρμογή. Εφ' όσον είναι δυνατή η πρόσβαση μέσω προγραμματισμού στα δεδομένα μπορεί να συμμετέχει σε συγχρονισμό.

Η βασική λειτουργία ενός παρόχου είναι η ικανότητά του να αποθηκεύει δεδομένα που αφορούν πληροφορίες κατάστασης και αλλαγής. Τα μεταδεδομένα μπορούν να αποθηκευτούν σε ένα αρχείο, σε μια βάση δεδομένων ή σε ένα αρχείο δεδομένων προέλευσης, με τη βοήθεια του συγχρονισμού. Ως βοήθεια, το Sync Framework προσφέρει μια πλήρη αποθήκευση πληροφοριών για αποθηκευμένα δεδομένα που βρίσκονται σε μια βάση δεδομένων και λειτουργεί ταυτόχρονα με μια διεργασία. Τα μεταδεδομένα για την αποθήκευση των δεδομένων μπορούν να αναλυθούν σε πέντε βασικά στοιχεία:

- Versions (εκδόσεις)
- Knowledge (γνώση)
- Tick count (χρονικό σημείο ή χτύπος ρολογιού)
- Replica ID (ρέπλικα ID)
- Tombstones (επιτύμβιες στήλες)

Για κάθε στοιχείο το οποίο έχει υποστεί συγχρονισμό, μια μικρή ποσότητα από πληροφορίες είναι αποθηκευμένες και περιγράφουν το πού και πότε έχουν αλλάξει τα δεδομένα. Τα μεταδεδομένα αποτελούνται από δύο εκδόσεις: μια creation version(έκδοση δημιουργίας) και μια update version(έκδοση ανανέωσης). Μια έκδοση αποτελείται από δύο συστατικά: ένα tick count το οποίο είναι η χρονική στιγμή αποθήκευσης δεδομένων και μια replica ID που είναι ο πάροχος ο οποίος εκτελεί την αλλαγή. Όσον αφορά την ανανέωση των αρχείων, η creation tick count εφαρμόζεται στο στοιχείο κάθε φορά και ο update tick count αυξάνεται κατά την αποθήκευση των δεδομένων. Η replica ID είναι μια μοναδική τιμή που προσδιορίζει ένα συγκεκριμένο χώρο αποθήκευσης δεδομένων. Η creation version είναι ίδια με την update version, τη στιγμή που δημιουργείται ένα στοιχείο. Ενώ με τις επόμενες ανανεώσεις που θα γίνουν στο αρχείο η update version θα τροποποιηθεί.

Βασικοί τρόποι όπου μπορεί να εφαρμοστεί το versioning είναι οι εξής:

I. **Inline tracking** (Inline παρατήρηση αλλαγών): Σε αυτή τη μέθοδο γίνεται η αλλαγή στο αντικείμενο που συγχρονίζεται και αμέσως καταγράφεται η αλλαγή. Από τη στιγμή που θα πραγματοποιηθεί κάποια αλλαγή σε ένα αρχείο άμεσα θα ενημερωθεί και θα αλλάξουν τα μεταδεδομένα του αρχείου.

II. **Asynchronous tracking** (Ασύγχρονη παρατήρηση αλλαγών): Στη μέθοδο αυτή καταγράφεται η αλλαγή του αντικειμένου την ίδια στιγμή που συμβαίνει. Αυτή η διαδικασία μπορεί να εκτελεστεί πριν από το συγχρονισμό. Ένας συνήθης τρόπος για να ελέγξει για τυχόν αλλαγές είναι να αποθηκεύει κάποια πληροφορία κατάστασης του αρχείου (π.χ. ημερομηνία τροποποίησης).

Μια άλλη βασική έννοια είναι η έννοια της γνώσης. Η γνώση είναι μια συμπαγής αναπαράσταση των αλλαγών. Οι πάροχοι χρησιμοποιούν τη γνώση της replica για τα ακόλουθα:

I. Καταμέτρηση αλλαγών (προσδιορίζει ποιες αλλαγές δεν γνωρίζει η άλλη replica).

II. Εντοπισμός συγκρούσεων (καθορίζει τις εργασίες που έγιναν χωρίς να το γνωρίζει ο ένας για τον άλλον).

Κάθε replica πρέπει επίσης να διατηρεί tombstones για καθένα από τα αντικείμενα που έχουν διαγραφεί. Αυτό είναι σημαντικό επειδή όταν εκτελείται ο συγχρονισμός, εάν το στοιχείο δεν είναι πλέον εκεί, ο provider δε θα έχει κανένα τρόπο να αποδείξει ότι το σημείο αυτό έχει διαγραφεί και δεν θα μπορεί να διαδώσει την αλλαγή και στους άλλους providers. Μια tombstone πρέπει να περιέχει τις ακόλουθες πληροφορίες:

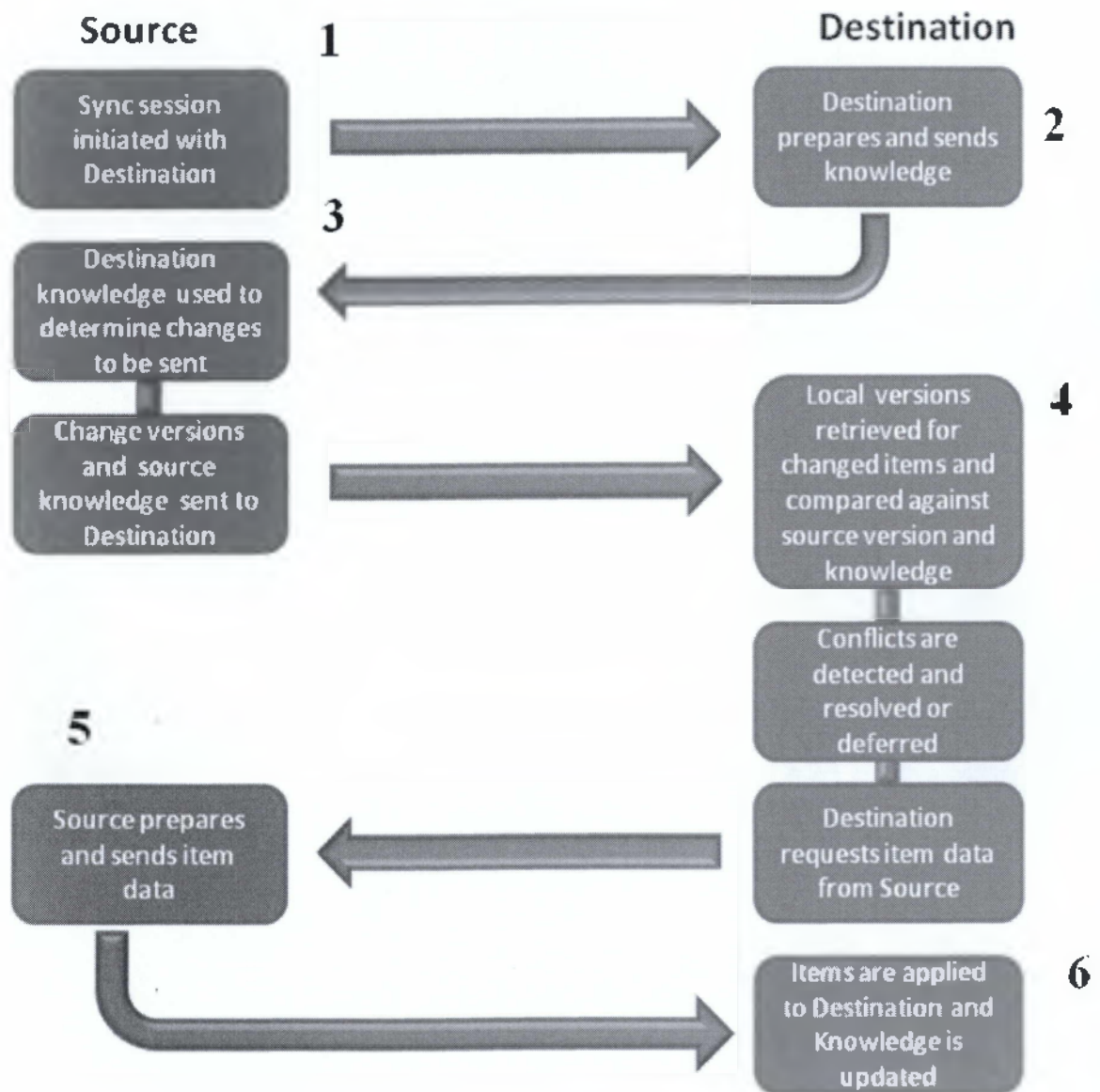
- Παγκόσμια ταυτότητα.
- Έκδοση Διαγραφής.
- Δημιουργία έκδοσης.

Επειδή ο αριθμός των tombstones θα αυξηθεί με την πάροδο του χρόνου, μπορεί να είναι φρόνιμο να δημιουργεί μια διαδικασία ώστε να καθαρίζονται οι διαγεγραμμένες πληροφορίες μετά από ένα χρονικό διάστημα, ώστε να εξοικονομείτε χώρο. Το Sync Framework παρέχει υποστήριξη για τη διαχείριση tombstones πληροφοριών.

2.6 Ποή Synchronization

Προκειμένου να ξεκινήσει ο συγχρονισμός έχουμε την source(πηγή), όπου αυτή συνδέεται με το destination(προορισμό). Στο παρακάτω διάγραμμα σκιαγραφείται η

ροή του συγχρονισμού. Σε περίπτωση αμφίδρομου συγχρονισμού η διαδικασία αυτή εκτελείται δυο φορές με αντεστραμμένους τους ρόλους πηγής και προορισμού.



Εικόνα 19: Ροή Συγχρονισμού

1. Στην αρχή αρχικοποιούμε τη φάση συγχρονισμού. Κατά τη διάρκεια αυτής της φάσης, η source (πηγή) ξεκινάει να επικοινωνεί με το destination (προορισμός). Η σχέση μεταξύ αυτών ονομάζεται Synchronization session (συνεδρία συγχρονισμού).

2. Στη συνέχεια, εφόσον, κάθε πάροχος έχει τη δική του μοναδική γνώση, ο destination προετοιμάζει να στείλει τη δική του γνώση στη source.
3. Στο σημείο αυτό η source λαμβάνει τη γνώση του και βλέπει ποιές μεταβολές της δεν γνωρίζει ο destination. Αφού τα ελέγξει και συγκρίνει τις δυο γνώσεις θα στείλει πίσω στο destination τις αλλαγές που έγιναν στη δική της γνώση.
4. Τώρα ο destination συγκρίνει τα τοπικά αρχεία με αυτά που έλαβε από τη source και βλέπει αν υπάρχουν αλλαγές. Επιπλέον, χρησιμοποιεί τις πληροφορίες που διαθέτει προκειμένου να εντοπίσει αν υπάρχουν συγκρούσεις. Σύγκρουση θα εντοπιζόταν εάν ένα αντικείμενο είχε αλλαχτεί τόσο στη source, όσο και στο destination, έτσι ώστε να μην ξέρουμε ποια έκδοση να κρατήσουμε. Βέβαια, ανεξάρτητα με το εάν υπάρχουν συγκρούσεις ή όχι, η ροή του συγχρονισμού συνεχίζεται κανονικά. Στην ουσία ο destination καθορίζει τα αρχεία που θέλει να λάβει προκειμένου να τα ζητήσει από τη source.
5. Στη φάση αυτή η source πρέπει να μεταφέρει τα πραγματικά αρχεία στο destination.
6. Τέλος, ο destination έλαβε τα δεδομένα και ανανέωσε τη γνώση του.

Σε περίπτωση που χαθεί η συνδεσιμότητα τα αρχεία θα εμφανιστούν ως exceptions και θα διορθωθούν με τον επόμενο συγχρονισμό.

2.7 Παράδειγμα Synchronization

Χρησιμοποιώντας τη ροή συγχρονισμού που περιγράψαμε στην προηγούμενη ενότητα θα αναλύσουμε με ένα παράδειγμα πως λειτουργεί το Sync Framework. Στο παράδειγμα αυτό θα υπάρχουν δυο αντίγραφα: η replica A και η replica B. Η replica A θα πάρει τη θέση της source και η B τη θέση του destination. Άρα, η replica A θα είναι αυτή που θα ξεκινήσει το συγχρονισμό.

Για παράδειγμα, φανταστείτε ότι θέλετε να συγχρονίσετε τα αντικείμενα αυτών των δυο συμμετεχόντων. Ένα αντικείμενο θα είναι το I₁ το οποίο θα κρατάει πληροφορίες, όπως το πότε δημιουργήθηκε ή ανανεώθηκε και από ποιόν.

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₁	1	A	1	A

Επομένως, το I₁ δημιουργήθηκε τη χρονική στιγμή 1, ανανεώθηκε, επίσης, τη χρονική στιγμή 1 από τη replica A. Εάν βέβαια ανανεωθεί ξανά το αντικείμενο τότε ο πίνακας θα έχει ως εξής:

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₁	5	A	1	A

Ωστόσο, μπορούν να δημιουργηθούν πολλαπλά αντικείμενα. Όσο μεγαλώνει ο αριθμός των αρχείων, τόσο μεγαλώνουν και οι πληροφορίες έκδοσης. Το Sync Framework δεν απαιτείται να γνωρίζει όλα τα προηγούμενα update που έχουν αποθηκευτεί. Το μόνο που θέλει να γνωρίζει είναι την πιο πρόσφατη.

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₂	3	A	2	A
I ₃	4	A	4	A
I ₁	5	A	1	A

Σύμφωνα με τα παραπάνω αν παρατηρήσουμε στον παραπάνω πίνακα η τρέχουσα κατάσταση έχει ως εξής: Η γνώση της replica A τη χρονική στιγμή 5 συμβολίζεται με A₅.

Στη replica B πάλι υπάρχει μια σειρά αρχείων όπως φαίνεται παρακάτω.

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₁₀₄	2	B	1	B
I ₁₀₅	4	B	3	B

Άρα, η γνώση της replica B τη χρονική στιγμή 4 είναι το B₄. Σε αυτή τη φάση ξεκινάει ο συγχρονισμός μεταξύ των δυο αντιγράφων. Ως source παίρνουμε τη replica A και ως destination τη replica B. Κατά τη διάρκεια του συγχρονισμού η replica B στέλνει στη replica A τη γνώση του χωρίς να γνωρίζει τη γνώση της replica A. Οι γνώσεις έχουν 5 εξής:

Replica A Knowledge = A₅

Replica B Knowledge = B₄

Η replica A λαμβάνει αυτή τη γνώση και είναι πλήρης ενημερωμένη για όλα τα αρχεία. Επομένως, τα ελέγχει και στέλνει πίσω στη replica B μόνο την καινούργια γνώση που δεν έχει η B.

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₂	3	A	2	A
I ₃	4	A	4	A
I ₁	5	A	1	A

Η replica B δέχεται τα versions, τα ελέγχει και προσπαθεί μέσω αυτών να καθορίσει ποια είναι τα στοιχεία που θα ζητηθούν από τη replica A. Επίσης, χρησιμοποιεί αυτές τις πληροφορίες για να διαπιστώσει εάν υπάρχουν συγκρούσεις

(για παράδειγμα, το ίδιο αρχείο να έχει ανανεωθεί και στα δύο αντίγραφα). Στη συνέχεια η replica A στέλνει στοιχεία τα οποία δεν γνωρίζει. Σε αυτήν την περίπτωση, η replica A θα στείλει τα αρχεία που σχετίζονται με το I_1 , I_2 και I_3 . Η replica B λαμβάνει αυτά τα αρχεία και τα προσθέτει στο φάκελό της. Άρα, στο σημείο αυτό και τα δυο αντίγραφα έχουν ενημερωθεί πλήρως επειδή έχουν στην κατοχή τους όλα τα αντικείμενα.

Στο τέλος της συνεδρίας συγχρονισμού, η διαδικασία αυτή εκτελείται μια ακόμη φορά με τη διαφορά ότι η source γίνεται destination και το αντίθετο. Αυτό επιτρέπει στη Replica A να λαμβάνει οποιοδήποτε αντικείμενο έχει δημιουργηθεί ή αλλάξει στη Replica B (I104 και I105).

Συμπεραίνουμε στο τέλος ότι και τα δυο αντίγραφα έχουν την ίδια γνώση.

$$\text{Replica A Knowledge} = A_5, B_4$$

$$\text{Replica B Knowledge} = A_5, B_4$$

2.7.1 Παράδειγμα Σύγκρουσης

Επεκτείνοντας το προηγούμενο παράδειγμα θα αναλύσουμε τι γίνεται σε περίπτωση που δημιουργηθεί κάποια σύγκρουση στη ροή συγχρονισμού. Αρχικά, έχουμε όλα τα παρακάτω αντικείμενα:

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₁₀₄	2	B	1	B
I ₁₀₅	4	B	3	B
I ₂	3	A	2	A
I ₃	4	A	4	A
I ₁	5	A	1	A

Ομοίως, η γνώση και για τα δύο αντίγραφα, έχει ως εξής:

Replica A Knowledge = A₅, B₄

Replica B Knowledge = A₅, B₄

Σε αυτό το σημείο και τα δύο αντίγραφα αποφασίζουν να ανανεώσουν το ίδιο αντικείμενο (στοιχείο I₂).

Στη replica A ο πίνακας ενημερώνεται ανάλογα:

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₁₀₄	2	B	1	B
I ₁₀₅	4	B	3	B
I ₂	6	A	2	A
I ₃	4	A	4	A
I ₁	5	A	1	A

Στη replica B ο πίνακας ενημερώνεται ανάλογα:

Item	Update Tick Count	Update Replica ID	Creation Tick Count	Creation Replica ID
I ₁₀₄	2	B	1	B
I ₁₀₅	4	B	3	B
I ₂	5	B	2	A
I ₃	4	A	4	A
I ₁	5	A	1	A

1. Επομένως, η replica A ξεκινάει το συγχρονισμό με τη replica B στέλνοντας της τη γνώση της η οποία έχει αλλάξει. Η replica B λαμβάνει τις αλλαγές: Η replica B βλέπει μια νέα αλλαγή στο στοιχείο I₂ το οποίο είναι:

Update Tick Count	Update Replica ID
6	A

2. Η replica B γνωρίζει τις αλλαγές που πραγματοποιήθηκαν στη replica A (A₆, B₄) και συνειδητοποιεί ότι η replica A δεν έχει επίγνωση των αλλαγών που έχουν γίνει στο ίδιο αντικείμενο I₂ της replica B.

Update Tick Count	Update Replica ID
5	B

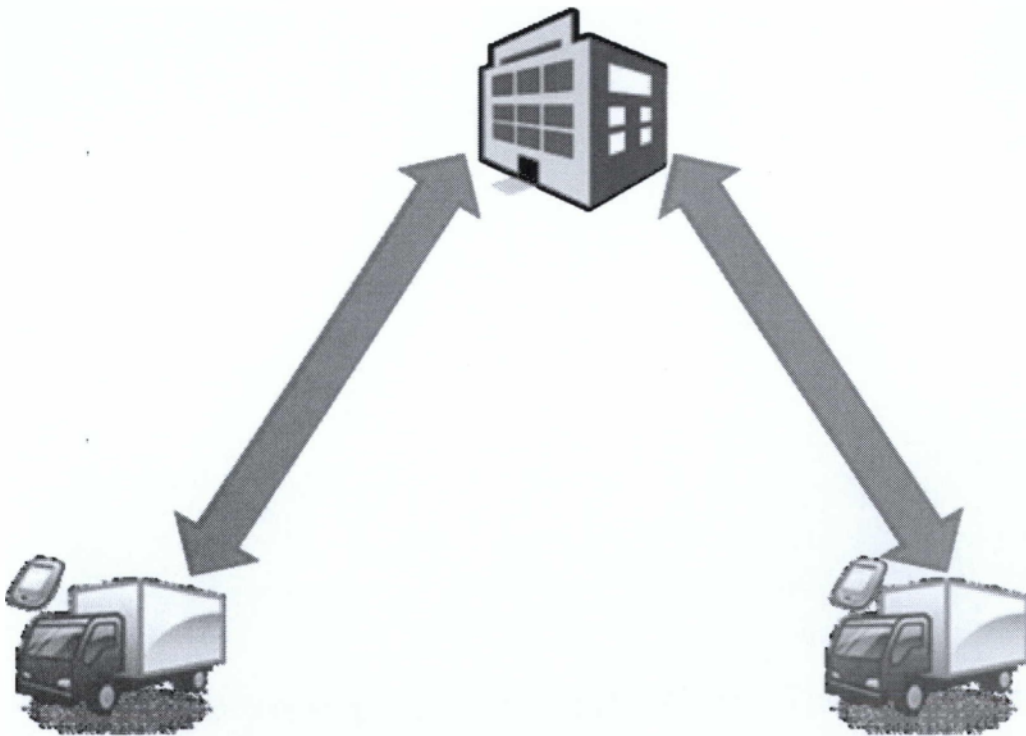
3. Εδώ εντοπίζεται η σύγκρουση, την οποία θα αντιμετωπίσει είτε ο χρήστης είτε μπορεί η ίδια η εφαρμογή να περιέχει λογική και να επιλέξει αυτόματα την έκδοση που επιθυμεί.

2.8 Τοπολογίες Multiple Synchronization

Στην ενότητα αυτή θα αναφερθούμε σε δυο βασικές τοπολογίες που υποστηρίζει το Sync Framework. Για ευκολότερη κατανόηση αναλύεται παρακάτω πως λειτουργεί μια εταιρία με κάθε μια από αυτές τις τοπολογίες.

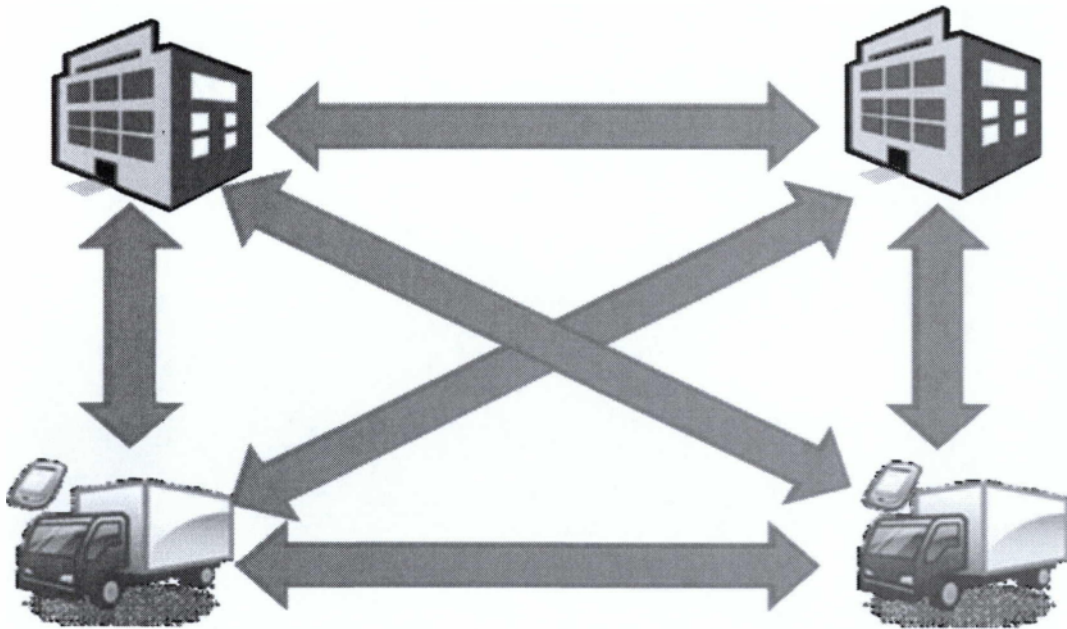
Στην ανάλυση που ακολουθεί έχουμε μια εταιρία η οποία έχει στην κατοχή της φορτηγά για να κάνει παραγγελίες σε όλη την Ελλάδα.

Στην πρώτη εικόνα έχουμε στα κεντρικά της εταιρίας εγκατεστημένο έναν κεντρικό server, ένα κοινό σημείο αναφοράς, ο οποίος επικοινωνεί με τους servers που είναι εγκατεστημένοι σε κάθε φορτηγό. Όλα τα φορτηγά συγχρονίζουν τις βάσεις τους αποκλειστικά με την κεντρική βάση η οποία ενημερώνει τους οδηγούς για τυχόν αλλαγές που έχουν πραγματοποιηθεί καθ' όλη τη διάρκεια της διαδρομής τους. Άρα, όταν θα συγχρονιστεί ένα φορτηγό θα έχει μια αρκετά πλήρη εικόνα του τι γίνεται στην εταιρία, εφόσον θα γνωρίζει όλες τις αλλαγές που έχουν πραγματοποιηθεί και στους υπόλοιπους οδηγούς.



Εικόνα 20: Λειτουργία εταιρίας με μοντέλο πελάτη/ εξυπηρετητή

Στην δεύτερη εικόνα, οι servers που υπάρχουν στην εταιρία επικοινωνούν με τους servers των οδηγών με την διάφορα ότι και οι οδηγοί πλέον επικοινωνούν και μεταξύ τους, γεγονός το οποίο προσφέρει μεγαλύτερη ευελιξία. Οι οδηγοί δεν είναι απαραίτητο να συγχρονίσουν τις βάσεις τους με κάποιο κεντρικό σημείο, αλλά μπορούν να τις συγχρονιστούν και μεταξύ τους. Στο σημείο αυτό, βέβαια, όταν το πρώτο φορτηγό συγχρονιστεί με το δεύτερο θα μάθει τις αλλαγές που έχουν γίνει στο φορτηγό αυτό και το αντίθετο, αλλά σε περίπτωση που υπήρχε και τρίτο φορτηγό για να μάθει τις αλλαγές του θα έπρεπε να συγχρονιστεί μαζί του.



Εικόνα 21: Λειτουργία εταιρίας με μοντέλο peer-to-peer

Το Sync Framework χρησιμοποιείται σχεδόν σε οποιονδήποτε τοπολογία συγχρονισμού. Τώρα οι εταιρίες δεν περιορίζονται πλέον σε μια μόνο τοπολογία, αλλά μπορούν να επιλέξουν μια ή και έναν συνδυασμό τοπολογιών. Αυτό σημαίνει ότι θα μπορούσαν να δημιουργήσουν συνδυασμούς offline και βασιζόμενες στη συνεργασία, αρχιτεκτονικές.

2.9 Τεχνικά Θέματα

2.9.1 Τεχνολογία Υλοποίησης

Σε μία n-Tier αρχιτεκτονική σχεδίασης για τις συσκευές, τα τμήματα συγχρονισμού χρησιμοποιούνται ως εξής:

I. Ο server synchronization provider (π.χ. DbServerSyncProvider) κατοικεί στον server ή σε κάποιο άλλο επίπεδο.

II. Ο server synchronization provider αθροίζει synchronization adapters (SyncAdapter), ο οποίος είναι ένα αντικείμενο το οποίο περιλαμβάνει τις δηλώσεις των εντολών για να επιλέξουμε, εισάγουμε, μεταβάλουμε και διαγράψουμε σειρές δεδομένων στην βάση δεδομένων του server. Χρησιμοποιούμε από έναν synchronization adapter για κάθε πίνακα που επιθυμούμε να συγχρονίσουμε.

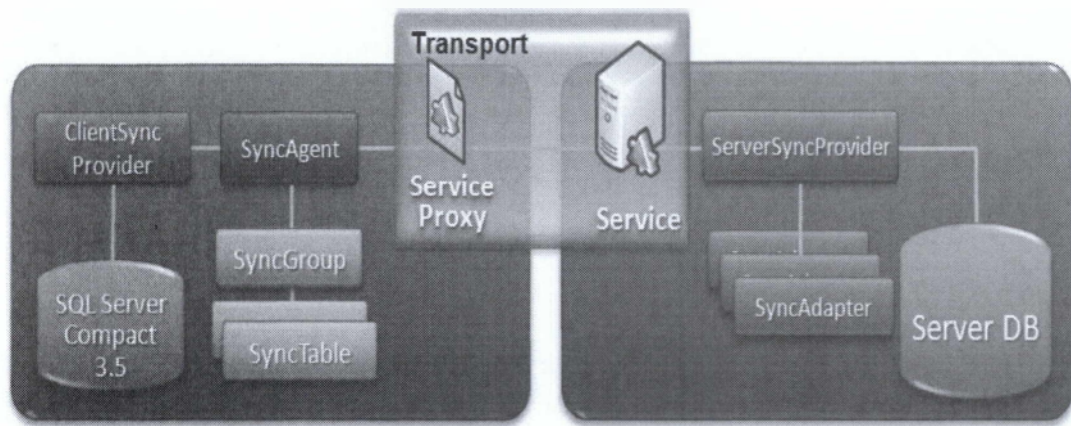
III. Ο client synchronization provider (π.χ. SqlCeClientSyncProvider) και ο synchronization agent (SyncAgent) κατοικούν και οι δυο στην κινητή συσκευή. Ο synchronization agent ενορχηστρώνει την όλη διαδικασία του συγχρονισμού. Ο client synchronization provider αλληλεπιδρά με την βάση του SQL Server Compact 3.5 SP1 έτσι ώστε να ενημερώσει τα δεδομένα και να αποθηκεύσει μεταδεδομένα συγχρονισμού.

IV. Ο SyncAgent χρησιμοποιεί μια συλλογή από synchronization table (SyncTable) αντικείμενα. Έχουμε από ένα SyncTable για κάθε πίνακα στην βάση δεδομένων που επιθυμούμε να συγχρονίσουμε. Ρυθμίζουμε τις ιδιότητες αυτών των αντικειμένων ώστε να προσδιορίσουμε το είδος του συγχρονισμού που επιθυμούμε να εκτελέσουμε.

V. Επίσης, μπορούμε να ομαδοποιήσουμε τα SyncTable αντικείμενα σε synchronization groups (SyncGroup), σε εκείνους τους πίνακες που είναι συσχετισμένοι με σχέσεις ξένων κλειδιών. Τα Synchronization Services συγχρονίζουν όλες τις αλλαγές μαζί στους πίνακες σε ένα SyncGroup έτσι ώστε να είναι σίγουρο

ότι δεν θα υπάρξουν μεταβολές στην ακεραιότητα των δεδομένων μεταξύ των πινάκων της ομάδας.

VI. Η επικοινωνία μεταξύ του server synchronization provider και του synchronization agent είναι διαχειρίσιμη από μια κλάση διαμεσολάβησης (ServerSyncProviderProxy) και μια Υπηρεσία. Οι Υπηρεσίες συγχρονισμού δεν προσδιορίζουν τον τύπο της υπηρεσίας που πρέπει να χρησιμοποιηθεί για αυτό τον σκοπό.



Εικόνα 22: Τα components που υλοποιούν τον συγχρονισμό του τοπικού SQL Server και του απομακρυσμένου με χρήση Υπηρεσιών.

2.9.2 Επιλέγοντας ένα πρωτεύον κλειδί κατάλληλο για τον συγχρονισμό των πινάκων των βάσεων με την χρήση του Sync Framework.

Για τους πίνακες που συμμετέχουν στον συγχρονισμό (Incremental Synchronization), το Sync Framework απαιτεί ότι κάθε γραμμή των πινάκων (Table Row) που θα συμμετέχουν σε αυτόν, πρέπει να προσδιορίζεται μοναδικά (Uniquely Identified). Αυτό δεν απαιτείται για το συγχρονισμό στιγμιότυπων (Snapshot Synchronization) πελατών και κεντρικών υπολογιστών. Τυπικά, οι γραμμές προσδιορίζονται από ένα πρωτεύον κλειδί που καθορίζεται στον κεντρικό υπολογιστή ή την ομότιμη (peer) βάση δεδομένων. Σε κατανομημένο περιβάλλον, πρέπει να είμαστε ιδιαίτερα προσεκτικοί όταν επιλέγουμε τον τύπο στήλης που θα

χρησιμοποιηθεί ως πρωτεύον κλειδί. Τα πρωτεύον κλειδιά πρέπει να είναι μοναδικά σε όλους τους κόμβους, και δεν πρέπει να επαναχρησιμοποιηθούν. Εάν μια σειρά διαγράφεται, το πρωτεύον κλειδί εκείνης της σειράς δεν πρέπει να χρησιμοποιηθεί για κάποια άλλη σειρά. Εάν ένα πρωτεύον κλειδί χρησιμοποιείται από περισσότερους από έναν κόμβους, μια σύγκρουση πρωτεύον κλειδιών (primary key collision) μπορεί να συμβεί. Αυτό μπορεί να συμβεί σε οποιοδήποτε είδος κατανεμημένου περιβάλλοντος και δεν είναι ένας περιορισμός του Sync Framework. Αυτή η ενότητα περιγράφει διάφορες επιλογές που έχουμε στην διάθεσή μας για την επιλογή πρωτεύον κλειδιού, και περιγράφει την καταλληλότητά τους για τα κατανεμημένα περιβάλλοντα.

2.9.3 Στήλες αυτόματης-αύξησης (ταυτότητα) – Auto-Increment (Identity) Columns

Οι αρχιτέκτονες βάσεων δεδομένων επιλέγουν συχνά μια στήλη αυτόματης-αύξησης για να χρησιμεύσουν ως το πρωτεύον κλειδί. Αυτή η ιδιότητα αυτόματης-αύξησης (η ιδιότητα Ταυτότητα (IDENTITY) στον SQL Server) παράγει μια νέα τιμή για κάθε γραμμή που εισάγεται σε έναν πίνακα. Αυτή η νέα τιμή παράγεται με την αύξηση ή τη μείωση της τρέχουσας τιμής από ένα σταθερό ποσό (Increment) και την ανάθεση του αποτελέσματος στη γραμμή που εισάγεται. Οι στήλες αυτόματης-αύξησης χρησιμοποιούν τύπους στοιχείων όπως οι ακέραιοι αριθμοί (Integers). Αυτό έχει ως αποτέλεσμα να οδηγήσουν σε έναν συμπαγέστερο συγκροτημένο ευρετήριο (compact clustered index), αποδοτικότερες συζεύξεις (Joins), και μικρότερη ροή δεδομένων όταν εκτελούνται ερωτήματα (Queries) στον πίνακα.

Οι ιδιότητες αρχικής τιμής και αύξησης του κλειδιού καθορίζονται και μπορούν να επιλεχθούν από έναν πεπερασμένο αριθμό πιθανών τιμών, η πιθανότητα μιας σύγκρουσης είναι πολύ υψηλή. Αυτός ο τύπος κλειδιού είναι κατάλληλος μόνο για σενάρια μεταφόρτωσης (Download-Only Scenarios). Σε αυτά τα σενάρια, ο κεντρικός υπολογιστής ή ένας οριζόμενος κόμβος πρέπει να είναι ο μόνος κόμβος που παράγει

τις νέες τιμές στο πρωτεύον κλειδί. Επομένως, αυτές οι τιμές είναι εγγυημένες για να είναι μοναδικές σε όλους τους κόμβους στην τοπολογία.

Οι στήλες αυτόματης-αύξησης είναι επίσης κατάλληλες για τα Upload-Only και αμφίδρομα (Bidirectional) σενάρια εάν η διαδικασία εισαγωγής δεδομένων πραγματοποιείται μόνο σε έναν κόμβο. Σε αυτά τα σενάρια, οι διαδικασίες εισαγωγής πραγματοποιείται μόνο στον κεντρικό υπολογιστή ή έναν οριζόμενο κόμβο και οι διαδικασίες μεταβολής δεδομένων (Updates), και διαγραφών (Deletes) ενδεχομένως να πραγματοποιούνται σε έναν ή περισσότερους πελάτες. Επειδή για τις ανάγκες της παρούσας εφαρμογής απαιτείται η διαδικασία εισαγωγής σε περισσότερους από έναν κόμβους, πρέπει να χρησιμοποιήσουμε μια από τις άλλες προσεγγίσεις που περιγράφονται παρακάτω.

2.9.4 GUIDs

Η χρησιμοποίηση ενός GUID, ενός μοναδικού προσδιοριστή (Unique Identifier) στον SQL Server ως πρωτεύον κλειδί, εγγυάται την μοναδικότητά του ανάμεσα σε οποιοδήποτε αριθμό κόμβων και αποφεύγει τις συγκρούσεις που είναι πιθανές με τις στήλες αυτόματης-αύξησης.

Εντούτοις, η χρησιμοποίηση ενός GUID στο πρωτεύον κλειδί έχει τις ακόλουθες συνέπειες:

I. Ο μεγάλος τύπος στοιχείων (16 αλφαριθμητικά) αυξάνει το μέγεθος του συγκροτημένου ευρετήριου (Clustered Index), που μπορεί να έχει επιπτώσεις στις συχνές διαδικασίες, όπως τις ενώσεις (Joins).

II. Η άτακτη παραγωγή GUIDs αναγκάζει τις νέες γραμμές να παρεμβάλλονται στις τυχαίες θέσεις στο συγκροτημένο ευρετήριο. Αυτό μπορεί στη συνέχεια να προκαλέσει έναν τεμαχισμένο του συγκεντρωμένου δείκτη (Fragmented Clustered Index). Αυτό μπορεί να έχει επιπτώσεις στο μέγεθος της ροής δεδομένων που απαιτείται για την εκτέλεση ερωτημάτων στον πίνακα.

Στην έκδοση 2005 του SQL Server και στις πιο πρόσφατες εκδόσεις, μπορούμε να χρησιμοποιήσουμε τη λειτουργία NEWSEQUENTIALID με την οποία παράγονται GUIDs διαδοχικά και να αποφύγουμε αυτόν τον τεμαχισμό.

2.9.5 Κλειδιά που περιλαμβάνουν ένα προσδιοριστικό κόμβων (Node Identifier).

Σε αυτήν την προσέγγιση, χρησιμοποιείτε ένα κλειδί που συνδυάζει μια τιμή που είναι μοναδική στον κεντρικό υπολογιστή ή στους πελάτες με μια τιμή που είναι μοναδική σε ολόκληρη την τοπολογία. Παραδείγματος χάριν, στο συγχρονισμό πελατών και κεντρικών υπολογιστών θα μπορούσαμε να χρησιμοποιήσουμε μια στήλη αυτόματος-αύξησης (μοναδική στον κόμβο) που συνδυάστηκε με μια στήλη που αποθηκεύει ένα hash της ταυτότητας που το Sync Framework ορίζει σε κάθε πελάτη. (Αυτό είναι το ClientId που είναι μοναδικό για όλη από την τοπολογία.).

ΚΕΦΑΛΑΙΟ 3^ο

ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ

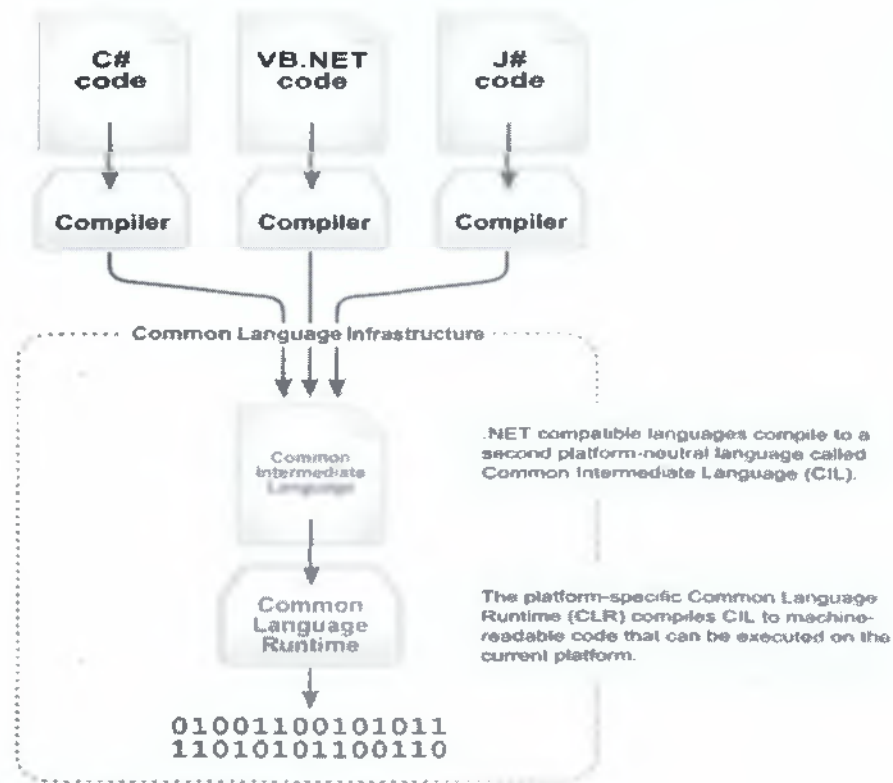
Η υλοποίηση της εφαρμογής έγινε σε C# χρησιμοποιώντας την πλατφόρμα .NET Framework. Η ανάπτυξη έγινε στο περιβάλλον του Microsoft Visual Studio, ενώ οι Βάσεις Δεδομένων υλοποιήθηκαν πάνω σε Microsoft SQL Server 2008. Είναι προγράμματα που χρησιμοποιούνται ευρέως για ένα σύνολο από εφαρμογές όπως η δική μας. Μπορούν να χρησιμοποιηθούν για την κατασκευή μικρών προγραμμάτων αλλά και μεγάλων εμπορικών εφαρμογών. Παρακάτω παρουσιάζουμε λίγες πληροφορίες γι' αυτές τις τεχνολογίες και τα προϊόντα.

3.1 NET Framework

Το .NET Framework της Microsoft είναι μία βιβλιοθήκη που μπορεί να εγκατασταθεί στους υπολογιστές που τρέχουν τα λειτουργικά συστήματα του Microsoft Windows. Περιλαμβάνει μια μεγάλη βιβλιοθήκη και μια εικονική μηχανή που διαχειρίζεται την εκτέλεση των προγραμμάτων που γράφονται συγκεκριμένα για το Framework. Το .NET Framework είναι μια προσφορά της Microsoft και προορίζεται να χρησιμοποιηθεί από τις περισσότερες νέες αιτήσεις που δημιουργούνται για την πλατφόρμα παραθύρων.

Το .NET Framework (προφέρεται dot net) είναι ένα πλαίσιο λογισμικού που τρέχει κυρίως σε Microsoft Windows. Περιλαμβάνει μια μεγάλη βιβλιοθήκη και υποστηρίζει πολλές γλώσσες προγραμματισμού καθώς επίσης επιτρέπει σε κάθε γλώσσα να χρησιμοποιεί κώδικα που γράφεται σε άλλες γλώσσες. Προγράμματα που έχουν γραφτεί για το .NET Framework εκτελούνται σε περιβάλλον λογισμικού (ως αντίθεση προς το περιβάλλον υλικό), γνωστό ως Common Language Runtime (CLR),

μια εφαρμογή εικονικής μηχανής που παρέχει σημαντικές υπηρεσίες, όπως η ασφάλεια, η διαχείριση μνήμης και ο χειρισμός εξαίρεσης. Η βιβλιοθήκη και η CLR αποτελούν μαζί το .NET Framework.

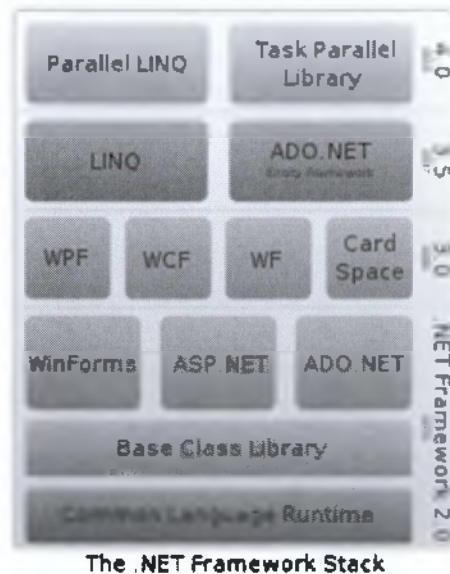


Εικόνα 23: Παράδειγμα εκτέλεσης κώδικα μέσω του CLR

Το .NET Framework είναι Βάση Βιβλιοθήκη Class που παρέχει διεπαφή χρήστη, πρόσβαση σε δεδομένα, σύνδεσης με βάσεις δεδομένων, κρυπτογράφηση, web εφαρμογή ανάπτυξης, αριθμητική αλγορίθμων, καθώς και δίκτυο επικοινωνιών. Οι προγραμματιστές παράγουν λογισμικό συνδυάζοντας το δικό τους κώδικα με το .NET Framework και άλλες βιβλιοθήκες. Το .NET Framework προορίζεται να χρησιμοποιείται από τις περισσότερες νέες εφαρμογές που δημιουργούνται για την πλατφόρμα των Windows. Η Microsoft παράγει επίσης ένα δημοφιλές ολοκληρωμένο περιβάλλον ανάπτυξης σε μεγάλο βαθμό για το .NET λογισμικού που ονομάζεται Visual Studio.

3.1.1 Ιστορική αναδρομή

Η Microsoft ξεκίνησε την ανάπτυξη για το .NET Framework στα τέλη της δεκαετίας του 1990 αρχικά με το όνομα της επόμενης γενιάς των Windows Services (NGWS). Μέχρι τα τέλη του 2000 κυκλοφόρησαν οι πρώτες εκδόσεις του NET 1.0.



Εικόνα 24: Το .NET στοίβα πλαίσιο

Η έκδοση 3.0 του .NET Framework περιλαμβάνεται στα Windows Server 2008 και Windows Vista. Η έκδοση 3.5 περιλαμβάνεται στα Windows 7, και μπορεί επίσης να εγκατασταθεί σε Windows XP και Windows Server 2003 που είναι οικογένεια των λειτουργικών συστημάτων. Στις 12 Απριλίου 2010, το .NET Framework 4 κυκλοφόρησε μαζί με το Visual Studio 2010.

Η οικογένεια του .NET Framework περιλαμβάνει, επίσης, δύο εκδόσεις για κινητά ή ενσωματωμένη χρήση της συσκευής. Μια μειωμένη έκδοση του framework, το .NET Compact Framework, είναι διαθέσιμο σε πλατφόρμες Windows CE, συμπεριλαμβανομένων των Windows Mobile συσκευών, όπως τα smartphones. Επιπλέον, το .NET Micro Framework απευθύνεται σε αυστηρά περιορισμένες συσκευές.

3.1.2 Πλεονεκτήματα .NET Framework

Το .NET έχει πολλά πλεονεκτήματα για την ανάπτυξη εφαρμογών:

I. Είναι εγγενώς αντικειμενοστραφές πλατφόρμα.

II. Είναι ανεξάρτητο από γλώσσα προγραμματισμού. Σε μια εφαρμογή ένας προγραμματιστής μπορεί να γράφει κώδικα σε C#, άλλος σε VB.NET και άλλος σε managed C++ και τα τμήματα που αναπτύσσει ο καθένας να συνεργάζονται μεταξύ τους χωρίς προβλήματα.

III. Η χρήση βιβλιοθηκών (assemblies) κάνει πολύ εύκολη την επαναχρησιμοποίηση κώδικα.

IV. Παρέχει πολύ εύκολη εγκατάσταση. Αρκεί να αντιγράψουμε το κατάλογο της εφαρμογής σε ένα άλλο υπολογιστή και αυτή θα τρέξει άμεσα. Δεν υπάρχει installation, δεν πειράζει το registry.

V. Παρέχει πληθώρα έτοιμων λειτουργιών που κάνουν την ανάπτυξη κώδικα πολύ εύκολη.

VI. Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

3.1.3 Μειονεκτήματα .NET Framework (για εφαρμογές με μεγάλες απαιτήσεις από το υλικό)

Το .NET έχει δυο μειονεκτήματα που αφορούν σε μια απαιτητική εφαρμογή όπως είναι η ανάπτυξη βιντεοπαιχνιδιών:

I. Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

II. Το CLR εισάγει μια (μικρή ίσως) καθυστέρηση στην εκτέλεση της εφαρμογής.

Το πρώτο, αν και είναι πολύ χρήσιμο χαρακτηριστικό για γενικό προγραμματισμό, σε ένα βιντεοπαιχνίδι μπορεί να δημιουργήσει πρόβλημα. Η απελευθέρωση μνήμης που δεν χρησιμοποιείται πλέον είναι μια ακριβή εργασία (από πλευράς χρόνου), η οποία επιπλέον είναι μη-ντετερμινιστική, μπορεί να συμβεί δηλαδή οποτεδήποτε. Αυτό μπορεί να έχει σαν αποτέλεσμα ένα παιχνίδι που τρέχει σε ένα σταθερό ρυθμό 60 καρέ το δευτερόλεπτο, να δει μια δραματική πτώση στο ρυθμό ανανέωσης για 1 δευτερόλεπτο, πράγμα ανεπίτρεπτο στην ανάπτυξη βιντεοπαιχνιδιών. Το δεύτερο αφορά τη Just in Time μεταγλώττιση που υποστηρίζει το CLR. Από τη μία εισάγει μια καθυστέρηση στην εκκίνηση του παιχνιδιού μιας και πρέπει να μεταγλωττιστεί ο κώδικας από την άλλη ο μεταγλωττιστής ο ίδιος δεν είναι βέλτιστος με την έννοια ότι δεν παράγει το καλύτερο δυνατό native κώδικα για Windows. Και τα δυο προβλήματα αυτά μπορούν να αντιμετωπιστούν αν τα λάβουμε υπόψη κατά την ανάπτυξη της εφαρμογής.

3.2 Microsoft Visual Studio 2008

3.2.1 Γενικά

Το Visual Studio IDE (Integrated Development Environment) είναι ένα επαγγελματικό εργαλείο ανάπτυξης λογισμικού που περιέχει ένα πλούσιο σύνολο από σχεδιαστικά εργαλεία, debugging εργαλεία και IntelliSense το οποίο ελέγχει για σφάλματα και προσφέρει λύσεις κατά την πληκτρολόγηση του κώδικα. Το Visual Studio έχει χαρακτηριστικά που προσφέρουν επιπλέον ευκολίες από την διαχείριση κώδικα. Τα πιο σημαντικά από αυτά φαίνονται στην παρακάτω λίστα.

I. Περιέχει έναν ενσωματωμένο web server: Για να λειτουργήσει μία ASP.NET διαδικτυακή εφαρμογή θα πρέπει να εκτελεστεί από ένα web server σαν τον IIS, ο οποίος περιμένει για web αιτήσεις και παρέχει τις κατάλληλες σελίδες. Για την

εγκατάσταση του IIS δεν υπάρχει δυσκολία αλλά μπορεί να μην είναι επιθυμητό. Χάρη στον ενσωματωμένο web server μπορεί ο προγραμματιστής να εκτελέσει μία διαδικτυακή εφαρμογή απευθείας από το σχεδιαστικό περιβάλλον. Επιπλέον, αυτή η προσέγγιση παρέχει ασφάλεια διότι ο web server εξυπηρετεί μόνο αιτήσεις από τον υπολογιστή που τρέχει η εφαρμογή.

II. Πολυγλωσσική ανάπτυξη: Το Visual Studio δίνει την δυνατότητα στον προγραμματιστή να χρησιμοποιήσει στην γλώσσα ή γλώσσες που επιθυμεί στο ίδιο IDE. Επιπλέον, είναι δυνατή η δημιουργία ξεχωριστών σελίδων σε διαφορετικές γλώσσες και να περιέχονται στην ίδια διαδικτυακή εφαρμογή. Ο μόνος περιορισμός είναι όχι δεν πρέπει να χρησιμοποιηθεί παραπάνω από μία γλώσσα σε κάθε σελίδα.

III. Χρειάζεται λιγότερος κώδικας να γραφεί από τον προγραμματιστή: Οι περισσότερες εφαρμογές χρειάζονται κάποια κομμάτια κώδικα που είναι πάντα ίδια. Επιπλέον, και το visual studio τα προσφέρει έτοιμα με τη μορφή προτύπων ή γραφικών εργαλείων.

3.2.2 Εισαγωγή στο Microsoft Visual Studio 2008

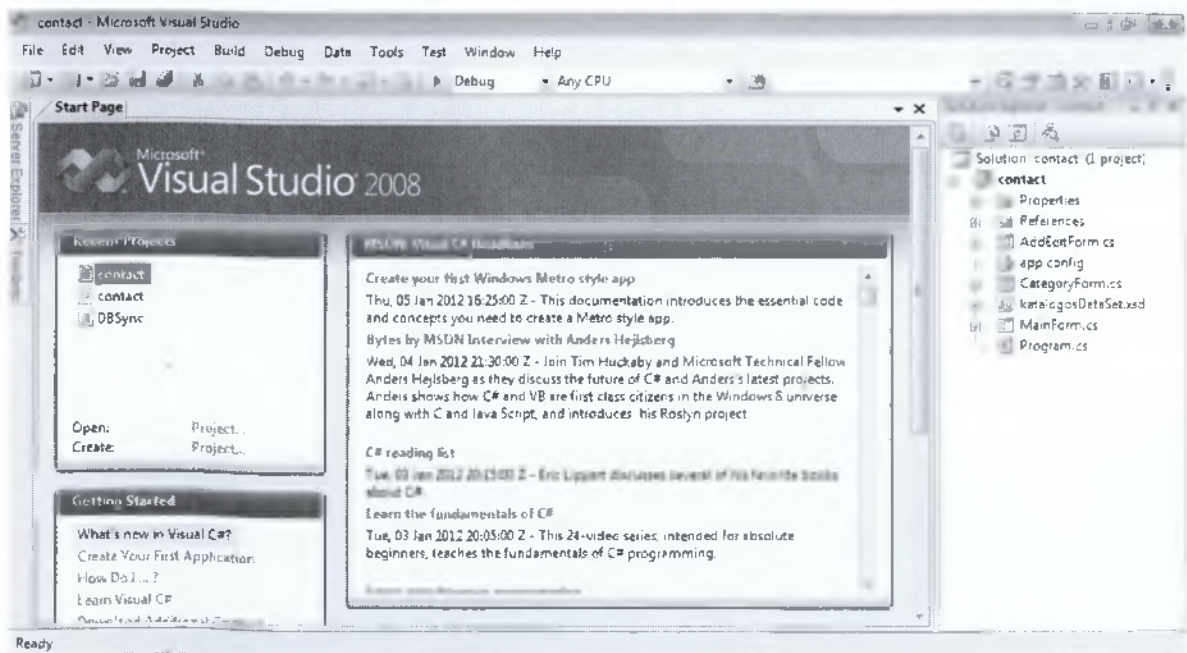
Το Microsoft Visual Studio αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) από τη Microsoft. Μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών κονσόλας, γραφικών εφαρμογών, ιστοσελίδων, Web εφαρμογών και υπηρεσιών για όλες τις πλατφόρμες που υποστηρίζονται από τα Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework και Microsoft Silverlight.

Το Microsoft Visual Studio περιλαμβάνει έναν συντάκτη κώδικα που υποστηρίζει IntelliSense καθώς επίσης και κώδικα. Ο ενσωματωμένος διορθωτής (debugger) λειτουργεί και σαν διορθωτής πηγής (source-level debugger) αλλά και σαν διορθωτής μηχανής (machine-level debugger). Άλλα ενσωματωμένα εργαλεία περιλαμβάνουν έναν σχεδιαστή φορμών για την σχεδίαση των γραφικών εφαρμογών

(Graphical User Interface - GUI), το σχεδιαστή ιστού, το σχεδιαστή κλάσεων, και το σχεδιαστή σχημάτων βάσεων δεδομένων.

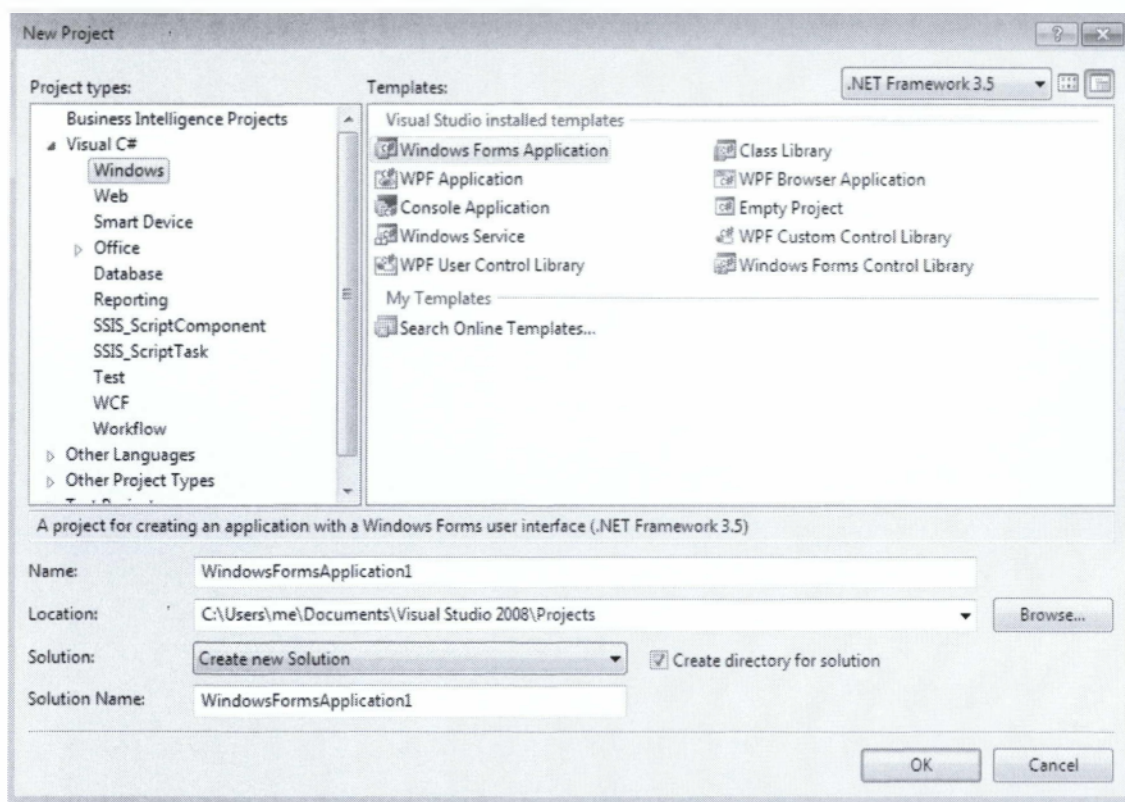
Το Microsoft Visual Studio υποστηρίζει διάφορες γλώσσες προγραμματισμού, με τη έννοια των γλωσσικών υπηρεσιών, οι οποίες επιτρέπουν σε οποιαδήποτε γλώσσα προγραμματισμού να υποστηριχθεί (σε διαφορετικό βαθμό) από το συντάκτη και το διορθωτή κώδικα, υπό τον όρο ότι υπάρχει μια συγκεκριμένη γλωσσική υπηρεσία. Στις ενσωματωμένες γλώσσες περιλαμβάνονται οι C/C++ (μέσω Visual C++), VB.NET (μέσω Visual Basic .NET), και C# (μέσω Visual C#). Η υποστήριξη για άλλες γλώσσες όπως F#, το M, Python, και την Ruby μεταξύ των άλλων έχει παρασχεθεί μέσω των γλωσσικών υπηρεσιών που πρόκειται να εγκατασταθούν χωριστά. Υποστηρίζει, επίσης XML/XSLT, HTML/XHTML, JavaScript και CSS.

3.2.3 Περιβάλλον εργασίας του Microsoft Visual Studio 2008



Εικόνα 25: Αρχική σελίδα του Visual Studio

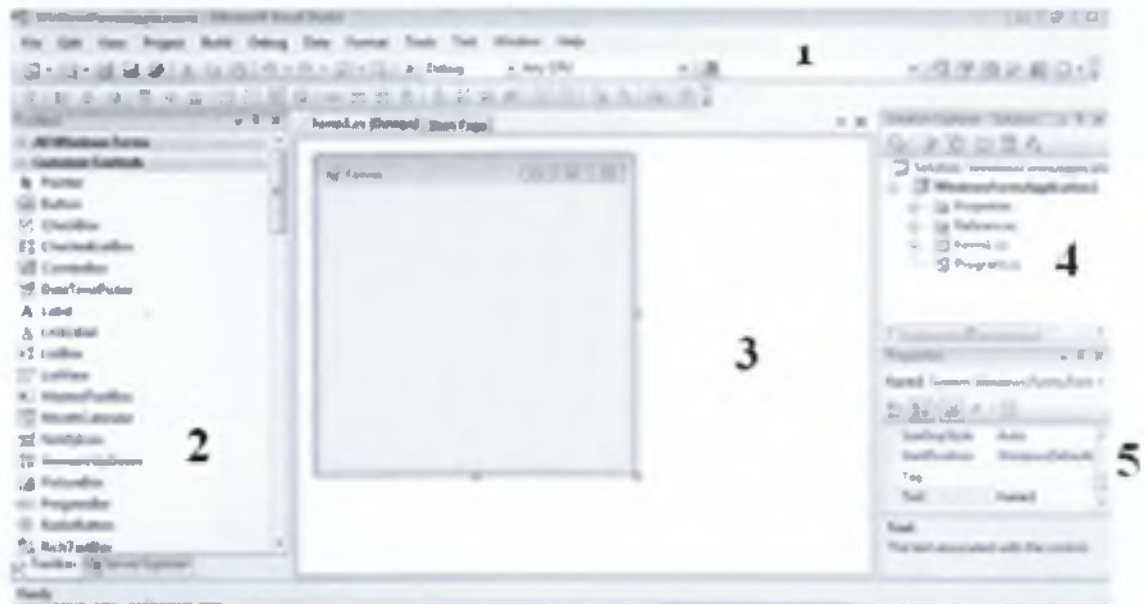
Ανοίγοντας το Visual Studio από το menu έναρξης των Windows εμφανίζεται η αρχική σελίδα. Στο σημείο που αξίζει να δώσουμε ιδιαίτερη προσοχή είναι το σημείο που γράφει Recent Projects (Εικόνα 25). Εκεί βρίσκουμε παλαιότερα project που έχουμε δημιουργήσει ή μπορούμε να δημιουργήσουμε ένα καινούριο project από την επιλογή Create Project. Επιλέγοντας Create Project θα εμφανιστεί ένα παράθυρο (Εικόνα 26) όπου εκεί θα διαλέξουμε το είδος του project που θέλουμε να δημιουργήσουμε, την γλώσσα προγραμματισμού που επιθυμούμε και το όνομα που θα του ορίσουμε.



Εικόνα 26: New Project

Όπως βλέπουμε στην περιοχή που ονομάζεται, Project Type μπορούμε να διαλέξουμε την γλώσσα προγραμματισμού που θέλουμε να δουλέψουμε πχ. Visual Basic ,C#, J# C++. Επιλέγοντας μια από αυτές, στην συγκεκριμένη περίπτωση εμείς διαλέξαμε την C#, μας δίνεται η δυνατότητα να διαλέξουμε αν θα δημιουργήσουμε μια εφαρμογή για Windows που θα τρέχει σε σταθερό υπολογιστή ή μια mobile εφαρμογή που θα τρέχει σε φορητές συσκευές, όπως Pocket PC και smartphone.

Στο δίπλα περιθώριο (Templates) διαλέγουμε πιο συγκεκριμένα την εφαρμογή που θέλουμε να δημιουργήσουμε, αν θα είναι ένα κενό Project ,μια Class Library κτλ. Συμπληρώνοντας το όνομα και την θέση που θέλουμε να αποθηκεύσουμε το Project, ανοίγει το βασικό περιβάλλον του Visual Studio (Εικόνα 27). Στην συγκεκριμένη περίπτωση εμείς έχουμε διαλέξει μια Windows Form Application με την γλώσσα C#.



Εικόνα 27: Περιβάλλον του Visual Studio

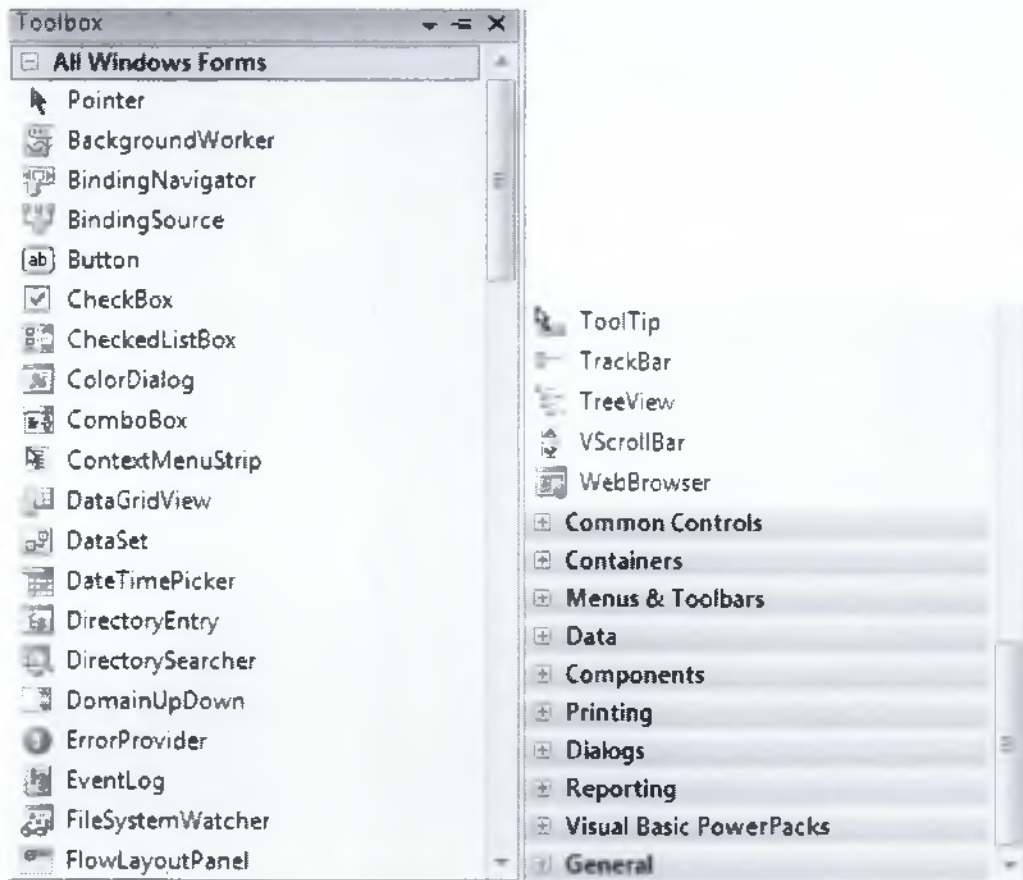
Όπως σε κάθε πλατφόρμα έτσι και σ' αυτήν υπάρχουν οι περιοχές εργασίας, σαν αυτή που έχουμε επιλέξει και στην παραπάνω φωτογραφία. Αρχίζοντας με την περιοχή 1 βλέπουμε την μπάρα του menu, με τις επιλογές file, open και γενικά όλα τα εργαλεία που υποστηρίζει η πλατφόρμα.(Εικόνα 28).



Εικόνα 28: Μπάρα menu και μπάρα εργαλείων του Visual Studio

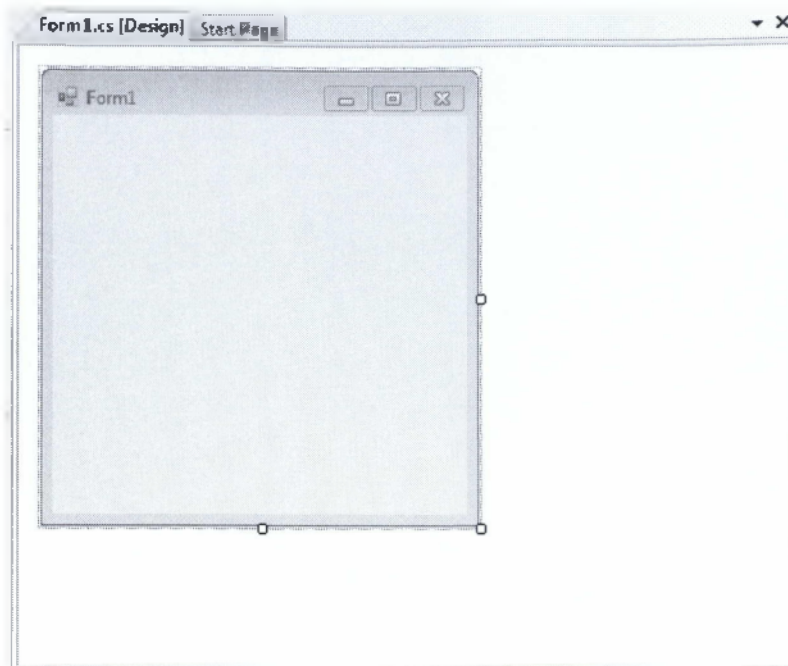
Από κάτω βρίσκεται η μπάρα εργαλείων, που από τα πιο χρήσιμα κουμπιά είναι αυτό του Debug. Το Debug κάνει Compile το πρόγραμμα που έχουμε δημιουργήσει

και το τρέχει ώστε να δει ο προγραμματιστής αν είναι σωστή η εφαρμογή του. Η δεύτερη περιοχή που έχουμε επιλέξει με το νούμερο 2 έχει όλα τα εργαλεία που χρειαζόμαστε για να υλοποιήσουμε μια Windows Form Application (Εικόνα 29).



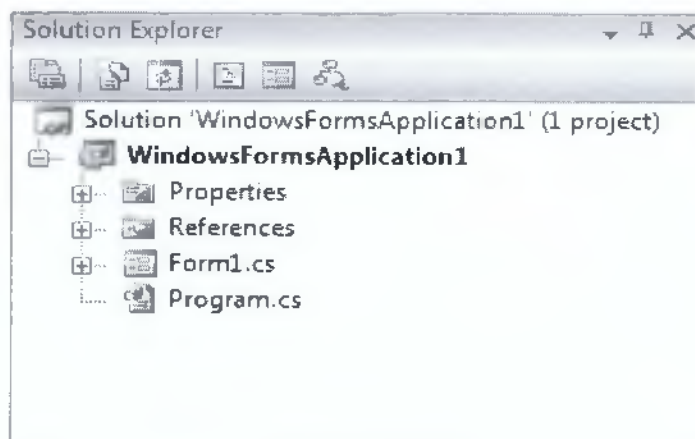
Εικόνα 29: Toolbox Explorer

Επιλέγοντας ένα από τα εργαλεία που θέλουμε όπως ένα button , ή ένα label και ότι μπορεί να υπάρχει σε μια Windows Form Application, με drag and drop μπορούμε να το προσθέσουμε στη δική μας φόρμα. Συνεχίζοντας στην περιοχή 3, το Visual Studio έχει δημιουργήσει μια κενή φόρμα όπου σε αυτήν εμείς θα προσθέσουμε τα κουμπιά τα label και γενικώς όποιο εργαλείο θέλουμε από το Toolbox Explorer. Ο σκοπός μας είναι να δημιουργήσουμε την δική μας Windows Form ανάλογα με τις απαιτήσεις μας (Εικόνα 30).



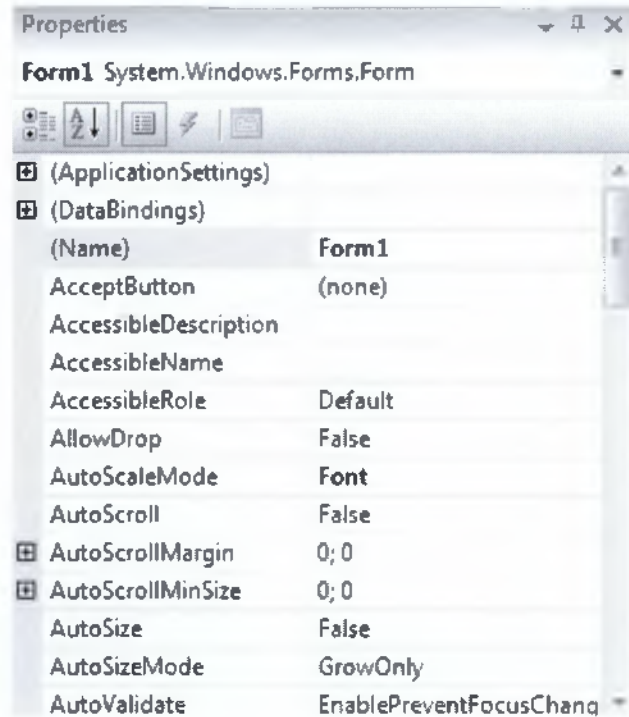
Εικόνα 30: Κενή winForm

Μόλις τοποθετήσουμε ένα εργαλείο μέσα στην φόρμα, το Visual Studio μας δίνει την δυνατότητα πατώντας διπλό κλικ πάνω σε αυτό, να βλέπουμε τον κώδικα που βρίσκεται από πίσω του, έτσι ώστε να έχουμε τον έλεγχο και να προγραμματίζουμε το κάθε εργαλείο ξεχωριστά. Πατώντας δυο φορές πάνω στην φόρμα ή εάν κάνουμε διπλό κλικ σε κάποιο σημείο της φόρμας εμφανίζεται όλος ο κώδικάς της.



Εικόνα 31: Το Solution Explorer

Δημιουργώντας ένα καινούριο project, το Visual Studio δημιουργεί ένα δέντρο με τις βιβλιοθήκες, τις κλάσεις και τις ιδιότητες του Project, φαίνεται στην περιοχή 4. Με λίγα λόγια το Solution Explorer (Εικόνα 31) μας διευκολύνει στην αναζήτηση του κώδικα, γιατί τακτοποιεί το project ανά κλάση, winForm και dataset.



Εικόνα 32: Property Explorer

Τέλος, στην περιοχή 5 βρίσκεται το Property Explorer (Εικόνα 32) όπου εκεί ανάλογα με το αντικείμενο που έχουμε επιλέξει μπορούμε να ρυθμίσουμε τις ιδιότητές του, το χρώμα, το όνομα και πολλά άλλα ανάλογα με το είδος του αντικειμένου.

3.3 Microsoft SQL Server 2008

3.3.1 Γενικά

Ο Microsoft SQL Server είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων που αναπτύχθηκε από την Microsoft. Οι βασικές γλώσσες για queries είναι η MS-SQL και η T-SQL.

Προσφέρει μια ασφαλή και αξιόπιστη αποθήκευση για τα δεδομένα. Συνδυάζει υπηρεσίες ανάλυσης, ενσωμάτωσης ενημέρωσης και είναι δυνατή η υλοποίηση και ανάπτυξη οικονομικά αποδοτικών λύσεων για τις ανάγκες των οργανισμών.

Ο SQL Server ανταγωνίζεται συστήματα βάσεων δεδομένων, όπως η Microsoft Access και το Microsoft Visual Fox Pro αλλά και συστήματα από άλλους κατασκευαστές όπως MySQL, Oracle, Sybase, IBM DB2, SAP DB, PostgreSQL και SwisSQL.

Η Microsoft Access υπάρχει σε πολλούς υπολογιστές σαν κομμάτι του Microsoft Office, αυτό είχε σαν αποτέλεσμα την χρησιμοποίησή του από πολλούς χρήστες. Δυστυχώς, όμως, υπάρχουν όρια στην επεκτασιμότητα, ταχύτητα και ευελιξία του συστήματος. Αυτό όμως δεν είναι εμπόδιο για μικρές εφαρμογές που τα παραπάνω δεν είναι αναγκαία.

Η MySQL είναι μια βάση δεδομένων ανοικτού κώδικα. Έχει υψηλή απόδοση και δυνατότητα κλιμάκωσης στις εφαρμογές των βάσεων δεδομένων.

Η Oracle δεν υπάρχει αμφιβολία ότι είναι ένα ισχυρό εργαλείο στο χώρο αλλά θεωρείτε ότι είναι περισσότερο πολύπλοκο στην εγκατάσταση και στη διαχείριση του. Πολλά χαρακτηριστικά της προσφέρουν επεκτασιμότητα και υψηλή απόδοση δεν είναι όμως τόσο φιλικό στην ανάπτυξη ολοκληρωμένων λύσεων για την αποθήκευση δεδομένων.

Αναμφίβολα όμως, ο SQL Server είναι δυνατή γλώσσα προγραμματισμού που παρέχει γραφικό περιβάλλον και παρόχους συγχρονισμού. Είναι η καλύτερη επιλογή για την ανάπτυξη του συστήματος που περιγράφεται στην πτυχιακή μας διότι παρέχει

ευκολία εγκατάστασης καθώς είναι ένα ολοκληρωμένο πακέτο χαρακτηριστικών. Επίσης, μπορεί να γίνει επέκταση της εφαρμογής χωρίς να υπάρχει περιορισμός στην ποσότητα των δεδομένων.

Ειδικότερα, όσο αφορά το κόστος είναι μηδαμινό για την χρησιμοποίησή του στη παρούσα πτυχιακή επειδή η Microsoft το προσφέρει δωρεάν για εκπαιδευτικούς σκοπούς. Επιπλέον, είναι βελτιστοποιημένος λόγω της ανάπτυξης .NET εφαρμογών και την χρησιμοποίησή του μέσω και του Visual Studio.

3.3.2 Εισαγωγή στο Microsoft SQL Server 2008

Ο Sql Server αποτελεί συνέχεια του Sybase Sql Server, που αρχικά αναπτυσσόταν μόνο για συστήματα Unix από την εταιρεία Sybase. Το 1988, οι εταιρείες Sybase, Microsoft και Ashton-Tate ολοκλήρωσαν από κοινού την ανάπτυξη του Sql Server 1.0 για το λειτουργικό σύστημα OS/2. Στη συνέχεια η απόσυρση της εταιρείας Ashton-Tate από την ανάπτυξη του προϊόντος καθώς και η μεταφορά του Sql Server στην πλατφόρμα των Windows NT, έδωσαν στη Microsoft τον πρώτο λόγο στη συνεργασία της με τη Sybase.

Η Microsoft και η Sybase εμπορεύονταν και υποστήριζαν από κοινού το προϊόν μέχρι την έκδοση 4.21. Το 1993 η συμφωνία από κοινού ανάπτυξης του προϊόντος μεταξύ της Microsoft και της Sybase έληξε και οι δύο εταιρείες διαχώρισαν τους δρόμους τους, συνεχίζοντας να αναπτύσσουν κάθε μια το δικό της πλέον προϊόν. Η πρώτη έκδοση του Sql Server χωρίς τη συμμετοχή της Sybase ήταν η έκδοση 6.0, η οποία ήταν σχεδιασμένη αποκλειστικά για τα Windows NT, ενώ οι εκδόσεις του Microsoft Sql Server μέχρι και το 1994 περιείχαν, βάση νομικής απόφασης, ενδείξεις πνευματικών δικαιωμάτων της Sybase, ως ένδειξη της προέλευσης τους.

Η έκδοση 7.0 του Sql Server ήταν το πρώτο πραγματικά γραφικά διαχειριζόμενο σύστημα διαχείρισης βάσεων δεδομένων, ενώ η έκδοση 2000 του Sql Server ήταν το πρώτο σύστημα διαχείρισης βάσεων δεδομένων για την IA64 αρχιτεκτονική της Intel.

Μια άλλη έκδοση του Microsoft SQL Server είναι η 2008, η οποία δόθηκε στην αγορά τον Αύγουστο του 2008. Στις 11 Ιουλίου 2011 κυκλοφόρησε για πρώτη φορά ο Microsoft SQL Server 2008 R2.

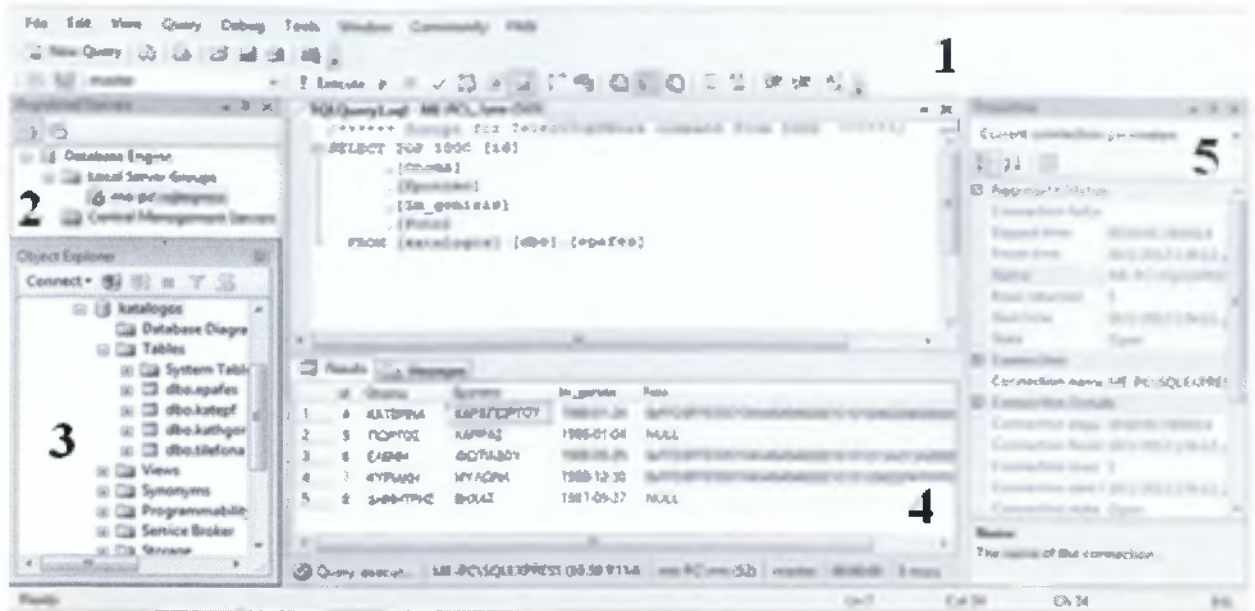
3.3.3 Περιβάλλον εργασίας του Microsoft SQL Server 2008

Πατώντας τον SQL Server Management Studio από το menu έναρξης των Windows θα εμφανιστεί η παρακάτω εικόνα:



Εικόνα 33: Είσοδος στον SQL Server

Στο πρώτο πεδίο (Server Type) επιλέγουμε τον τύπο server στον οποίο θέλουμε να συνδεθούμε. Οι επιλογές είναι οι ακόλουθες: Database Engine, Analysis Services, Reporting Services, SQL Server Compact Edition και Integration Service. Εμείς για το συγκεκριμένο project επειδή θέλουμε να δημιουργήσουμε μια βάση δεδομένων επιλέγουμε να συνδεθούμε με το Database Engine. Το δεύτερο πεδίο μας ζητάει να επιλέξουμε το όνομα του server. Στο τρίτο να ορίσουμε αν η πιστοποίηση θα γίνεται από τα Windows (Windows Authentication) ή από τον Sql Server (SQL Server Authentication). Πληκτρολογώντας το User Name και το Password ανοίγει το περιβάλλον εργασίας του SQL Server 2008 (Εικόνα 33). Το User Name και το Password είναι τα ίδια με αυτά που μας ζητήθηκε να ορίσουμε όταν κάναμε την εγκατάσταση του SQL Server.



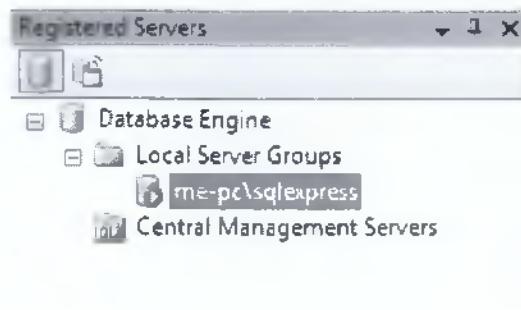
Εικόνα 34: Περιβάλλον εργασίας του SQL Server

Όπως βλέπουμε (Εικόνα 34) το περιβάλλον του SQL Server Management Studio (SSMS) αποτελείται από 5 βασικές περιοχές.



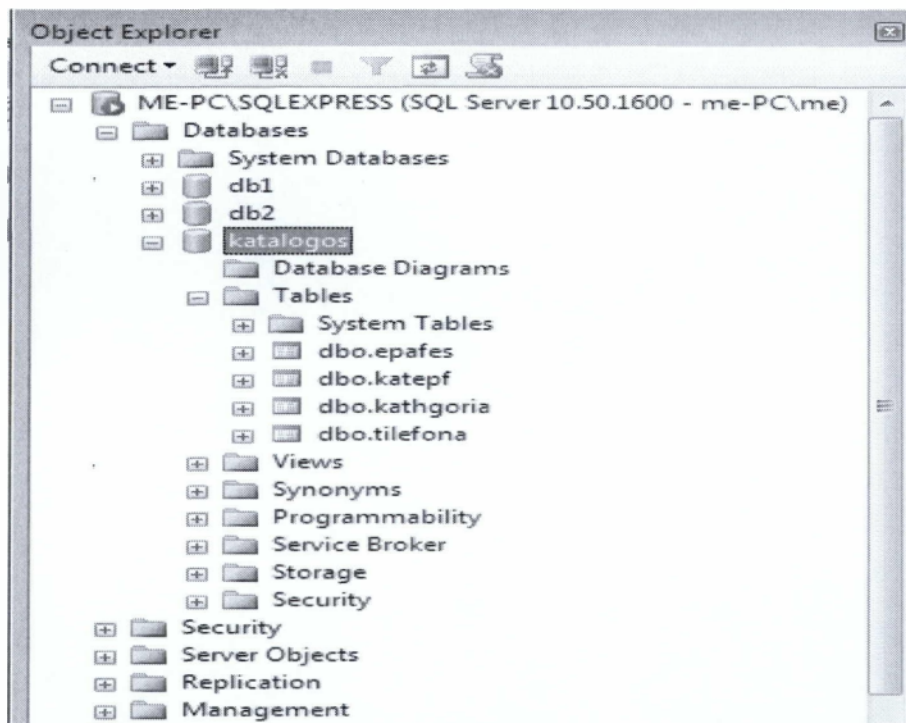
Εικόνα 35: Η γραμμή του menu με τη μπάρα εργαλείων.

Ξεκινάμε με την πρώτη περιοχή που έχουμε επιλέξει (Εικόνα 35). Είναι η μπάρα του menu μαζί με την μπάρα εργαλείων. Στη μπάρα του menu όπως και σε κάθε πρόγραμμα, βρίσκουμε όλες τις λειτουργίες του προγράμματος από το Save μέχρι και όλα τα εργαλεία που υποστηρίζει το SSMS. Ακριβώς από κάτω βρίσκεται η μπάρα εργαλείων που μας δίνει την δυνατότητα να εκτελούμε κάποιες ενέργειες άμεσα χωρίς να χάνουμε χρόνο ψάχνοντας μέσα στην γραμμή του menu. Ενδεικτικά μερικά από τα κουμπιά είναι η δημιουργία ερωτημάτων (Queries) για μια βάση δεδομένων που έχουμε δημιουργήσει και το κουμπί της αποθήκευσης. Εκτελώντας κάποια συγκεκριμένη εργασία προσθέτονται ή αφαιρούνται αυτόματα τα κατάλληλα κουμπιά ώστε να μας διευκολύνουν στην υλοποίηση της.



Εικόνα 36: Registered Servers

Η δεύτερη περιοχή που έχουμε επιλέξει (Εικόνα 36) μας βοηθά να αποθηκεύουμε και να διαλέγουμε τους server που χρησιμοποιούμε. Το SSMS έχει την δυνατότητα να συνδέεται με διαφόρους Server ώστε να μπορεί να επεξεργάζεται βάσεις που βρίσκονται στον καθένα από αυτούς. Στην εικόνα 36 βλέπουμε ότι υπάρχει καταχωρημένος ο Server με το όνομα me-pc\sql express.



Εικόνα 37: Object Explorer

Η τρίτη περιοχή που έχει επιλεγεί είναι μια από τις βασικότερες. Αυτή είναι ο Object Explorer (Εικόνα 37). Βλέπουμε ότι ο Object Explorer μας δείχνει τα περιεχόμενα ενός server. Στην περίπτωση αυτή ο server έχει το όνομα me-

rs\sqlexpress, που συνήθως είναι και το όνομα του υπολογιστή που είναι εγκατεστημένος ο SQL Server. Τα περιεχόμενα που θα μας απασχολήσουν περισσότερο στο project αυτό είναι οι βάσεις δεδομένων (Databases). Εκτός από τις Default βάσεις που έχει μπορούμε να δημιουργήσουμε και εμείς μια δική μας με τους πίνακες και τα πεδία που εμείς επιθυμούμε. Στην παραπάνω εικόνα την βάση δεδομένων katalogos την έχουμε δημιουργήσει εμείς για να υλοποιήσουμε την πτυχιακή άσκηση. Εκτός από τις βάσεις ο Object Explorer μας δίνει την δυνατότητα να επεξεργαστούμε τις ιδιότητες της κάθε βάσης, όπως τις παραμέτρους σύνδεσης της κάθε βάσης με άλλες, τις παραμέτρους ασφάλειας και τον τύπο επικοινωνίας αυτής με τις άλλες βάσεις.

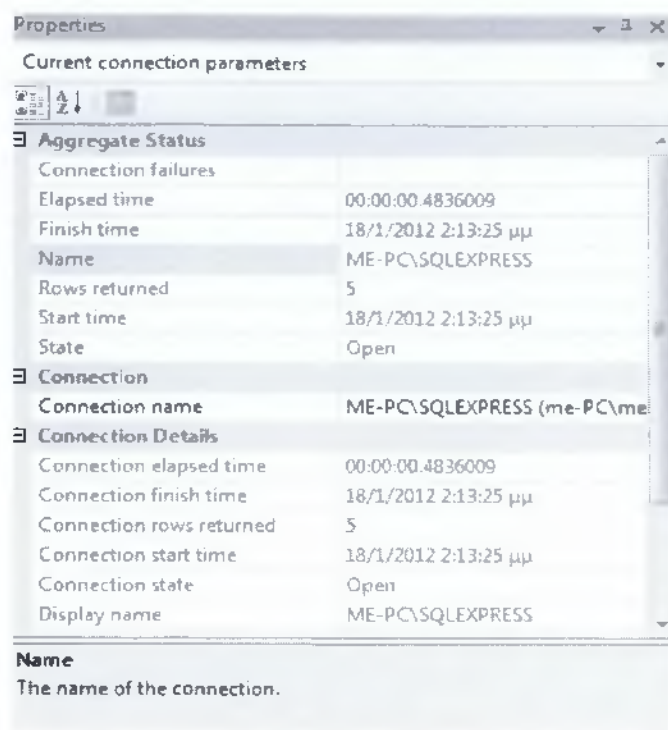
Σημαντικά είναι και τα κουμπιά του Object Explorer , που μας δίνουν την δυνατότητα να συνδεθούμε με διαφορετικούς τύπους server, ή να διακόψουμε την σύνδεση μας με το κουμπί Disconnect .

Ο Object Explorer Details που είναι το πεδίο 3 που έχουμε επιλέξει, ουσιαστικά είναι η περιοχή που δημιουργούμε και επεξεργαζόμαστε τις βάσεις μας και τις ιδιότητες τους. Με λίγα λόγια είναι η περιοχή εργασίας του SSMS.

	id	Όνομα	Επωνυμο	Im_genisis	Foto
1	4	ΚΑΤΕΡΙΝΑ	ΚΑΡΑΓΩΡΓΟΥ	1988-01-24	0xFFD8FFE000104A46494600010101006000600000FFD800...
2	5	ΠΩΡΓΟΣ	ΚΑΡΡΑΣ	1986-01-04	NULL
3	6	ΕΛΕΝΗ	ΦΩΤΙΑΔΟΥ	1988-05-29	0xFFD8FFE000104A46494600010101013A013A0000FFE11F...
4	7	ΚΥΡΙΑΚΗ	ΜΥΛΩΝΑ	1988-12-30	0xFFD8FFE000104A46494600010101006000600000FFD800...
5	8	ΔΗΜΗΤΡΗΣ	ΒΗΧΑΣ	1987-09-27	NULL

Εικόνα 38: Object Explorer Details.

Στην εικόνα 38 βλέπουμε ότι έχουμε ανοίξει το table με το όνομα erafes της βάσης katalogos και βλέπουμε τα δεδομένα. Αυτά μπορούμε να τα επεξεργαστούμε προσθέτοντας ή αφαιρώντας μια εγγραφή και ρυθμίζοντας ανάλογα τις ιδιότητες της.



Εικόνα 39: Properties Explorer

Τέλος , η περιοχή 5 (Εικόνα 39) που ονομάζεται Properties Explorer είναι αυτή που μας δίνει την δυνατότητα να βλέπουμε και να επεξεργαζόμαστε τις ιδιότητες του κάθε αντικειμένου που έχει επιλεγθεί..

3.4 Γλώσσα προγραμματισμού C#

Η C# είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού η οποία δημιουργήθηκε από τη Microsoft για το .NET Framework. Στη συνέχεια αναγνωρίστηκε σαν πρότυπο από τους οργανισμούς Ecma και ISO.

Η C# ενθαρρύνει την αντικειμενοστρέφεια αλλά μπορεί να χρησιμοποιηθεί και για συναρτησιακό ή δομημένο προγραμματισμό. Στόχος της Microsoft ήταν να φτιάξει μια γλώσσα απλή, μοντέρνα, γενικού-σκοπού και αντικειμενοστρεφή.

3.4.1 Λόγοι Δημιουργίας

Καθώς η Microsoft εδώ και μια δεκαετία προσθέτει συνεχώς νέα χαρακτηριστικά στο λειτουργικό της, και στις βιβλιοθήκες των Windows, οι γλώσσες Visual Basic και C++ επεκτείνονται και αυτές σε μια παράλληλη τροχιά. Αυτό είχε ως αποτέλεσμα και οι δύο γλώσσες να καταλήξουν με πανίσχυρα χαρακτηριστικά αλλά και με προβλήματα που προέκυψαν από τον τρόπο με τον οποία εξελίχθησαν.

Στην περίπτωση της Visual Basic και πριν την έκδοση 6, η δύναμη της γλώσσας αντλούνταν από την απλότητα της χρήσης της. Σε βάρος της απλότητας, πολλά προγραμματιστικά προβλήματα που είχαν σχέση με το Windows API και το COM component δεν ήταν εύκολο να λυθούν. Αυτό συνέβαινε γιατί η Visual Basic δεν ήταν ποτέ, πραγματικά αντικειμενοστρεφής γλώσσα με αποτέλεσμα όσο μεγάλωνε το μέγεθος της εφαρμογής να μειώνεται η οργάνωση του κώδικα. Παράλληλα γινόταν πολύ δύσκολη η διατήρηση της εφαρμογής. Σε συνδυασμό με αυτά, η γλώσσα αυτή λόγω της καταγωγής της (δεν προοριζόταν για ανάπτυξη μεγάλων εμπορικών εφαρμογών, αλλά κυρίως για εκμάθηση) διατήρησε κάποια χαρακτηριστικά της τα οποία όμως δεν την έκαναν μια ολοκληρωμένη αντικειμενοστρεφή γλώσσα προγραμματισμού.

Η C++ έχει τις ρίζες της στην Ansi C++. Βέβαια δεν είναι ακριβώς ίδιες, καθώς η Microsoft έφτιαξε πρώτα τον μεταγλωττιστή για τη C++ και στη συνέχεια έγινε επίσημο το Ansi πρότυπο αλλά μοιάζουν σε πολλά στοιχεία. Δυστυχώς όμως αυτό οδήγησε σε δύο προβλήματα. Αρχικά, η Ansi C++ έχει τις ρίζες της σε παλιές τεχνολογίες, και αυτό υποδηλώνει έλλειψη υποστήριξης για ένα σύνολο νέων χαρακτηριστικών όπως είναι τα unicode strings και η παραγωγή XML Documentation. Επίσης κάποια χαρακτηριστικά της γλώσσας είχαν σχεδιαστεί με βάση παλαιότερων μεταγλωττιστών. Κατά δεύτερον, η Microsoft προσπάθησε να εξελίξει την C++ σε μια γλώσσα προγραμματισμού η οποία θα χρησιμοποιούνταν για την ανάπτυξη, υψηλών προδιαγραφών και υψηλής απόδοσης, λογισμικού για την πλατφόρμα των Windows. Προκειμένου να γίνει αυτό η Microsoft αναγκάστηκε να προσθέσει ένα σύνολο βιβλιοθηκών και λέξεων κλειδιών με αποτέλεσμα η γλώσσα αυτή σε περιβάλλον Windows να είναι ένα "χάος". Ένα χαρακτηριστικό παράδειγμα

αυτού του γεγονότος είναι ο μέγανος αριθμός τύπων που υπάρχουν για τα αλφαριθμητικά (char, LPTSTR, string, CString, wchar_t, OLECHAR κτλπ).

Με τον ερχομό του .NET Framework η Microsoft αναγκάστηκε να προσθέσει και άλλα χαρακτηριστικά στις δύο προαναφερόμενες γλώσσες. Συγκεκριμένα στην C++ προστέθηκαν νέες λέξεις κλειδιά και η Visual Basic άλλαξε ριζικά, προκύπτοντας η γλώσσα προγραμματισμού Visual Basic .NET. Στη Visual Basic .NET διατηρήθηκε η σύνταξη της απλής Visual Basic αλλά ο σχεδιασμός του διέφερε εντελώς. Όλοι αυτοί οι λόγοι, συντέλεσαν στη δημιουργία μιας νέας γλώσσας από την Microsoft η οποία θα ήταν προορισμένη αποκλειστικά και μόνο για το .NET Framework. Και έτσι δημιουργήθηκε η C#.

3.4.2 Γενικά Χαρακτηριστικά

Η C# είναι μια γλώσσα απλή, μοντέρνα, αντικειμενοστρεφής και έχει στοιχεία από τις C, C++ και JAVA γλώσσες προγραμματισμού. Συντακτικά η C# μοιάζει πολύ στη C++ και στη JAVA καθώς πολλές λέξεις-κλειδιά είναι ίδιες. Επίσης μοιράζεται με αυτές, α) τη δομή των μπλοκ τα οποία ορίζονται με τα σύμβολα “{” και “}” αλλά και β) την οριοθέτηση των εντολών με το σύμβολο “;”. Μπορεί με την πρώτη ματιά ενός κομματιού κώδικα σε C# να πούμε ότι είναι εμφανής η ομοιότητα με τις γλώσσες C++ και JAVA αλλά θεωρείται ότι η C# είναι πιο εύκολη στην εκμάθηση από την πρώτη και παρόμοιας δυσκολίας με την δεύτερη. Ο σχεδιασμός της είναι συνυφασμένος με τα μοντέρνα προγραμματιστικά πρότυπα και παράλληλα έχει υιοθετήσει την απλότητα χρήσης της Visual Basic αλλά και την υψηλή απόδοση της C++ σε θέματα χαμηλού επιπέδου διαχείρισης της μνήμης, σε περίπτωση που χρειαστεί.

Τα κύρια χαρακτηριστικά της C# είναι τα εξής:

I. Πλήρης υποστήριξη για κλάσεις και αντικειμενοστραφή προγραμματισμό, η οποία περιέχει κληρονομικότητα διεπαφής (Interfaces) και υλοποίησης, εικονικές συναρτήσεις και υπερφόρτωση τελεστών.

II. Ένα συνεπές και καθορισμένο με σαφήνεια σύνολο βασικών τύπων μεταβλητών.

III. Δεν υπάρχουν καθολικές μεταβλητές και μέθοδοι. Όλες οι μέθοδοι πρέπει να δηλώνονται μέσα στις κλάσεις. Οι δημόσιες (public) κλάσεις μπορούν να αντικαταστήσουν τις καθολικές μεταβλητές και τις καθολικές συναρτήσεις.

IV. Ενσωματωμένη δυνατότητα για αυτόματη παραγωγή XML documentation.

V. Αυτόματο καθάρισμα της δυναμικά δεσμευμένης μνήμης.

VI. Δυνατότητα για μαρκάρισμα κλάσεων ή μεθόδων με κάποια συγκεκριμένα χαρακτηριστικά. Αυτό μπορεί να είναι χρήσιμο για την τεκμηρίωση της εφαρμογής αλλά και στην μεταγλώττιση της εφαρμογής. (για παράδειγμα μπορούμε να μαρκάρουμε κάποιες μεθόδους ώστε να μεταγλωττίζονται μόνο όταν η εφαρμογή είναι σε λειτουργία αποσφαλμάτωσης (debug mode)).

VII. Πλήρης πρόσβαση στη βασική βιβλιοθήκη του .NET Framework αλλά και εύκολη πρόσβαση στο Windows API (αν χρειαστεί).

VIII. Προσπέλαση της μνήμης με δείκτες ή απ' ευθείας (αν χρειαστεί).

3.4.3 Μειονεκτήματα

Πέρα από τη δύναμη C#, υπάρχουν και κάποιοι περιορισμοί οι οποίοι την κάνουν ακατάλληλη γλώσσα προγραμματισμού για τη συγγραφή κάποιων εφαρμογών. Το κύριο μειονέκτημά της είναι ότι δεν έχει σχεδιαστεί για τη συγγραφή προγραμμάτων τα οποία έχουν σαν πρώτη προτεραιότητα τις ακραίες επιδόσεις. Αν λοιπόν ενδιαφέρει τον προγραμματιστή αν ένα βρόγχο θα πάρει 1.050 κύκλους μηχανής αντί για 1.000, και αν κάθε δέκατο του δευτερολέπτου είναι σημαντικό για την ανάγκη που εξυπηρετεί μια εφαρμογή τότε η καλύτερη λύση μεταξύ των low-level γλωσσών παραμένει η C++. Παρ' όλα αυτά το σύνολο των εφαρμογών που ανήκουν σε αυτή την κατηγορία είναι πολύ μικρό.

3.4.4 Υποστήριξη για Versioning

Η διαδικασία ανανέωσης ενός ήδη υπάρχοντος λογισμικού είναι επίπονη και επιρρεπής σε λάθη. Οι αλλαγές στον κώδικα μιας εφαρμογής μπορεί να αλλάξουν τον τρόπο λειτουργίας της. Η C# δίνει λύση σε αυτή τη διαδικασία καθώς περιλαμβάνει υποστήριξη για versioning.

Συγκεκριμένα, πολλές εφαρμογές που τρέχουν σε Windows παρουσιάζουν προβλήματα όταν για κάποιο λόγο μια assembly που χρησιμοποιούν αντικατασταθεί από μία νεότερη. Το πρόβλημα αυτό είναι ιδιαίτερα συχνό στο κόσμο των προγραμματιστών και “μπελάς” για τον κόσμο των εταιριών καθώς αυξάνουν τα κόστη ανάπτυξης. Στο .NET Framework κάθε DLL αρχείο ή COM αντικείμενο περιέχει πληροφορία για την έκδοσή του.

Ως αποτέλεσμα κάθε εφαρμογή που έχει χτιστεί πάνω στο .NET Framework φορτώνει τις εκδόσεις των DLLs ή COM αντικειμένων με τις οποίες έχει δοκιμαστεί ότι τρέχει χωρίς σφάλματα.

Με τη βοήθεια αυτού του χαρακτηριστικού, επιτρέπεται σε σύνθετα πακέτα λογισμικού να αναπτύσσονται και να εξελίσσονται σε βάθος χρόνου κάνοντας τις νεότερες εκδόσεις του πιο σταθερές.

3.4.5 Web Programming

Ένα χαρακτηριστικό της C# είναι η πλήρης υποστήριξη που έχει για τα πρότυπα και πρωτόκολλα που υποστηρίζονται από τον παγκόσμιο ιστό. Η C# παρέχει υποστήριξη για τη μετατροπή οποιουδήποτε τμήματος κώδικα σε Web Service με αποτέλεσμα, συναρτήσεις και γενικότερα πολλές από τις λειτουργίες του προγράμματος να μπορούν να προσπελαστούν από άλλες πλατφόρμες προγραμματισμού. Ο τεχνικός τρόπος με τον οποίο γίνεται αυτό είναι ο εξής: ένα XML Web Service ουσιαστικά είναι μια ASP.NET σελίδα η οποία επιστρέφει XML

κώδικα στον πελάτη αντί για HTML όταν ο τελευταίος ζητήσει μια πληροφορία. Τέτοια προγράμματα παράγουν DLL αρχεία τα οποία περιέχουν κλάσεις οι οποίες προέρχονται από την Webservice κλάση του .NET Framework. Τα XML Web Services μπορούν να χρησιμοποιηθούν πάνω από HTTP δίκτυα δεδομένου το ότι είναι ένα μέσο για μετάδοση πληροφοριών. Η XML είναι μια γλώσσα αυτοπεριγραφής και μη ειδικευμένη η οποία είναι κατανοητή από όλες τις γλώσσες προγραμματισμού.

ΚΕΦΑΛΑΙΟ 4^ο

ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΑΣ

4.1 Πρόλογος εφαρμογής

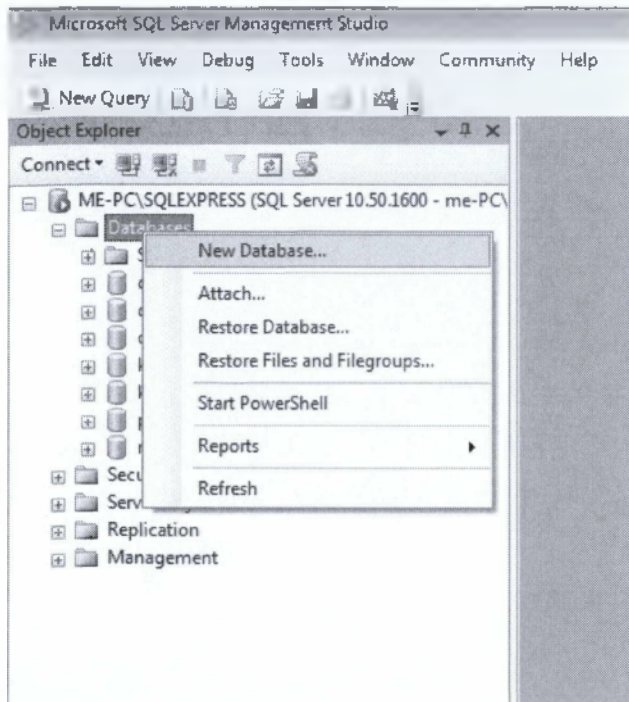
Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι ο σχεδιασμός και η υλοποίηση μιας εφαρμογής ηλεκτρονικής ατζέντας, στην οποία ο εκάστοτε χρήστης μπορεί να καταχωρήσει στοιχεία ανθρώπων που είναι χρήσιμα για αυτόν. Μαζί με την καταχώρηση των ονομάτων μπορούμε να πραγματοποιήσουμε εισαγωγή και άλλων προσωπικών δεδομένων κάθε ατόμου όπως ημερομηνία γέννησης, τηλέφωνα (σταθερά, κινητά, κτλ) και φωτογραφίες. Επιπλέον, δίνεται η δυνατότητα επεξεργασίας, (διαγραφή, αλλαγή) όλων των δεδομένων που έχουμε στην εφαρμογή μας.

Επίσης, συγχρονίζουμε την βάση δεδομένων με κάποια άλλη ίδια βάση δεδομένων που μπορεί να είναι σε οποιαδήποτε συσκευή (tablet, pc κτλ). Σημαντικό στοιχείο είναι ότι επιτρέπει το συγχρονισμό δεδομένων της εφαρμογής. Ο συγχρονισμός επιτρέπει να έχουμε κοινά δεδομένα και ο καθένας να έχει πρόσβαση όποτε το επιθυμεί. Υπάρχει η δυνατότητα να δουλεύουν οι χρήστες παράλληλα ακόμη και όταν δεν υπάρχει σύνδεση. Η βάση δεδομένων μας περιλαμβάνει όλα τα συστατικά που είναι απαραίτητα για την υλοποίηση μιας ηλεκτρονικής ατζέντας.

4.2 Δημιουργία της βάσης δεδομένων

Όπως αναφέραμε σε προηγούμενο κεφάλαιο η δημιουργία της βάσης δεδομένων γίνεται με την βοήθεια του προγράμματος Microsoft SQL Server 2008. Είναι σημαντικό πριν ξεκινήσουμε την υλοποίηση της βάσης, αλλά και ολόκληρης της εφαρμογής, να κάνουμε μια μελέτη που θα μας δώσει τις πληροφορίες που

χρειαζόμαστε ώστε η εφαρμογή να είναι λειτουργική. Πχ πρέπει να γνωρίζουμε τι στοιχεία πρέπει να έχει η ατζέντα μας προκειμένου η εφαρμογή να είναι πιο χρηστική. Αφού καταλάβουμε ποια πεδία θα πρέπει να έχει η βάση μας συνεχίζουμε στην υλοποίηση της.



Εικόνα 40. Δημιουργία νέας βάσης δεδομένων

Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
Onoma	nvarchar(50)	<input type="checkbox"/>
Eponimo	nvarchar(50)	<input type="checkbox"/>
In_geriss	date	<input checked="" type="checkbox"/>
Foto	varbinary(MAX)	<input checked="" type="checkbox"/>

Column Properties

(General)	
Name	id
Allow Nulls	No
Data Type	int
Default Value or Binding	
Table Designer	
Collation	collation_implicit
Computed Column Specification	
Condensed Data Type	int
Description	
Is Computed	Yes
(General)	

Εικόνα 41. Συμπλήρωση ενός πίνακα

Το όνομα της βάσης που έχουμε δημιουργήσει είναι 'Project' (Εικόνα 41). Αυτή αποτελείται από 4 πίνακες που τα ονόματά τους είναι:

- I. Erafes
- II. Thlefono
- III. Kathgoria
- IV. Keteprf

Όπως καταλαβαίνουμε ο πίνακα Erafes αποθηκεύει τα προσωπικά στοιχεία κάθε επαφής ενώ ο πίνακας thlefono αποθηκεύει τα τηλέφωνα και το είδος τους. Ο πίνακας Kathgoria αποθηκεύει τις κατηγορίες/ομάδες που ανήκει κάθε επαφή. Ενώ, ο τελευταίος πίνακας είναι ο πίνακας Keteprf που αποθηκεύει δεδομένα σύνδεσης ανάμεσα στους πίνακες erafes και kathgoria.

Πίνακας 1: Erafes

Όνομα Στήλης	Τύπος Δεδομένων	Επιτρέπεται κενή εγγραφή	Περιγραφή
Id	Int	Not Null	Πρωτεύον κλειδί . Είναι ο κωδικός της επαφής μοναδικός για κάθε επαφή μας. *Αυτόματη εισαγωγή αριθμού.
Onoma	Nvarchar(50)	Not Null	Όνομα επαφής
Eponymo	Nvarchar(50)	Not Null	Επίθετο επαφής
Im_genisis	date	Null	Ημερομηνία γέννησης επαφής
Foto	Varbinary(Max)	Null	Φωτογραφία επαφής

Πίνακας 2: Tilefona

Όνομα Στήλης	Τύπος Δεδομένων	Επιτρέπεται κενή εγγραφή	Περιγραφή
Id	Int	Not Null	Πρωτεύον κλειδί . Είναι ο κωδικός του τηλεφώνου μοναδικός για κάθε τηλέφωνο μας. *Αυτόματη εισαγωγή αριθμού.
Id_epafes	Int	Not Null	Δευτερεύον κλειδί συσχετιζόμενο με το id_epafes και τον πίνακα επαφές.
Telefono	Nchar(20)	Not Null	Αριθμός Τηλεφώνου
Typos	Nvarchar(20)	Null	Τύπος Τηλεφώνου

Πίνακας 3: Kathgoria

Όνομα Στήλης	Τύπος Δεδομένων	Επιτρέπεται κενή εγγραφή	Περιγραφή
Id_kat	Int	Not Null	Πρωτεύον κλειδί . Είναι ο κωδικός της κατηγορίας μοναδικός για κάθε κατηγορία μας. *Αυτόματη εισαγωγή αριθμού.
Onoma_kat	Nvarchar(20)	Not Null	Όνομα της κατηγορίας μας

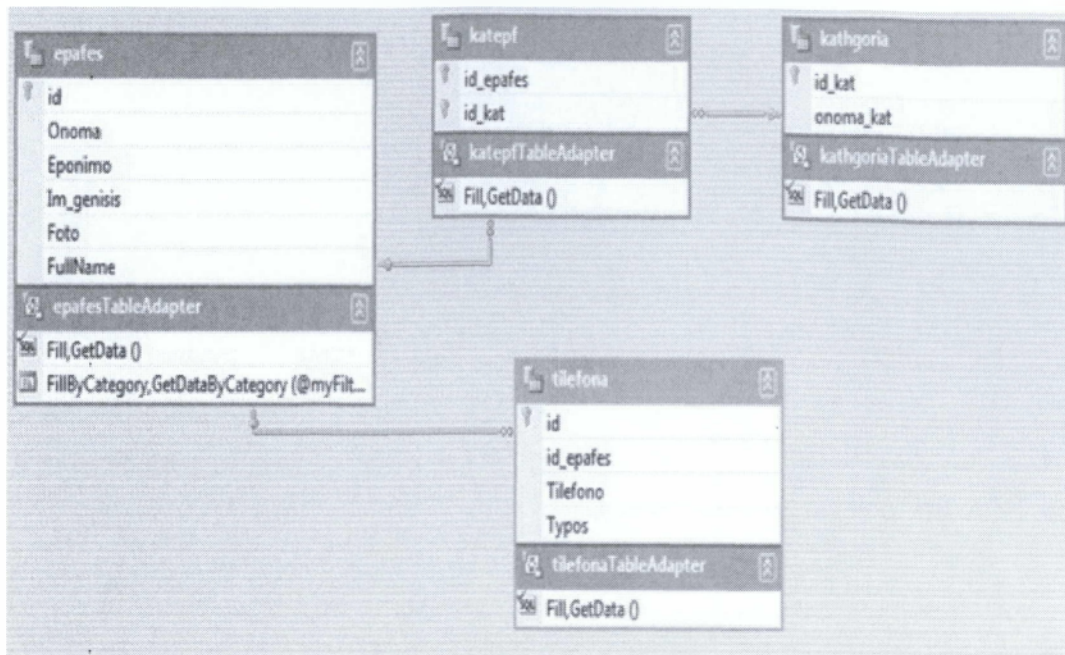
Πίνακας 4: Katepf

Όνομα Στήλης	Τύπος Δεδομένων	Επιτρέπεται κενή εγγραφή	Περιγραφή
Id_epafes	Int	Not Null	Δευτερεύον κλειδί συσχετιζόμενο με το id_epafes και τον πίνακα επαφές.
Id_kat	Int	Not Null	Δευτερεύον κλειδί συσχετιζόμενο με το id_kat και τον πίνακα kathgoria

Αρα, ο πίνακας katepf είναι ο πίνακας σύνδεσης που χρησιμοποιούμε για να δημιουργήσουμε μια σχέση πολλά-προς-πολλά. Μια επαφή μπορεί να αντιστοιχίζεται σε πολλές κατηγορίες, ενώ μια κατηγορία μπορεί να περιέχει πολλές επαφές. Περιέχει τα δυο πρωτεύον κλειδιά των πινάκων.

* Αυτόματη εισαγωγή αριθμού εννοούμε ότι στο Column Property έχουμε ρυθμίσει την επιλογή Identity Specification να είναι Yes και να έχει Identity Seed 1. Δηλαδή σε κάθε νέα εγγραφή που θα γίνεται στην στήλη που έχουμε επιλέξει, να καταχωρείτε αυτόματα ένα αριθμός και σε κάθε επομένη καταχώριση να προστίθεται συν 1 από τον προηγούμενο.

Τώρα θα πρέπει να ορίσουμε την έννοια του συσχετισμού μεταξύ των πινάκων (Relationships). Στους πίνακες μεταξύ τους δημιουργούμε έναν συσχετισμό έτσι ώστε τα κοινά τους πεδία να μην μπορούν να έχουν διαφορετικές τιμές. Αυτό γίνεται για τον λόγο ότι αν δυο πεδία σε δυο διαφορετικούς πίνακες έχουν διαφορετικές τιμές, δεν θα έχουμε την δυνατότητα να κάνουμε αναζήτηση των εγγραφών που έγιναν για ένα συγκεκριμένο πεδίο μιας άλλης εγγραφής. Πάντα κάθε πίνακας πρέπει να έχει ένα πρωτεύον κλειδί (σύνθετο ή μη) όπου αυτό δηλώνει ότι οι τιμές του πεδίου αυτού είναι μοναδικές. Το πρωτεύον κλειδί προσδιορίζει μονοσήμαντα μια οντότητα, ενώ ένα δευτερεύον κλειδί προσδιορίζει ένα υποσύνολο όλων των εγγραφών. Στην επόμενη εικόνα θα δούμε τις σχέσεις αυτών των πινάκων (Εικόνα 42).

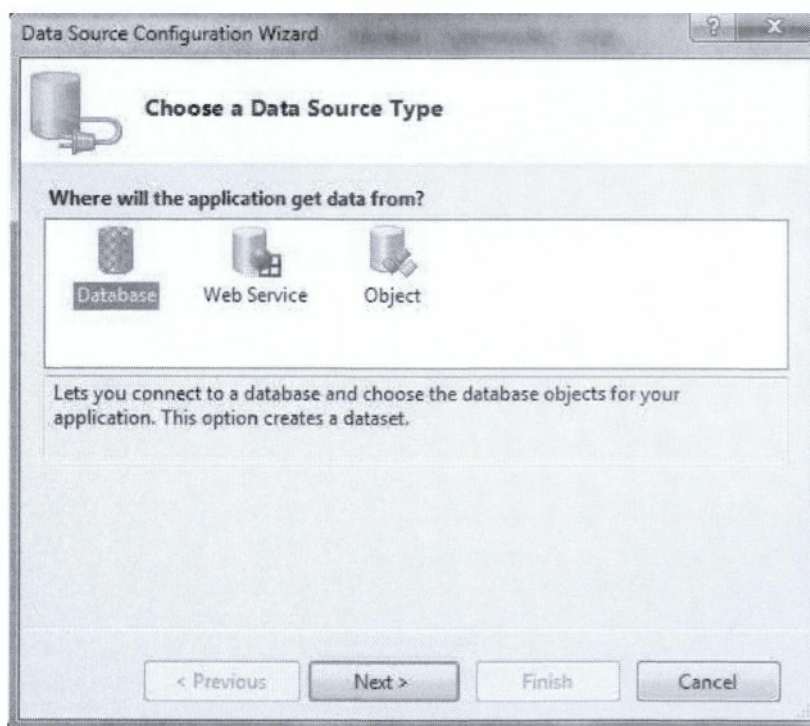


Εικόνα 42: Relationships των πινάκων.

Η εικόνα μας εξηγεί διαγραμματικά την σχέση που έχει ο ένας πίνακας με τον άλλον. Το κλειδί δείχνει ότι η εγγραφή είναι μοναδική ενώ το ∞ ότι οι εγγραφές είναι περισσότερες από μία. Για παράδειγμα, για μία εγγραφή του πεδίου 'id' του πίνακα epafes μπορούμε να έχουμε άπειρες εγγραφές του πεδίου 'id_epafes' για τον πίνακα tilefona. Έχοντας τελειώσει με την κατασκευή της βάσης δεδομένων θα μπορούσαμε να κάνουμε ερωτήματα (Query) για να κάνουμε αναζητήσεις συγκεκριμένων στοιχείων σε κάθε πίνακα.

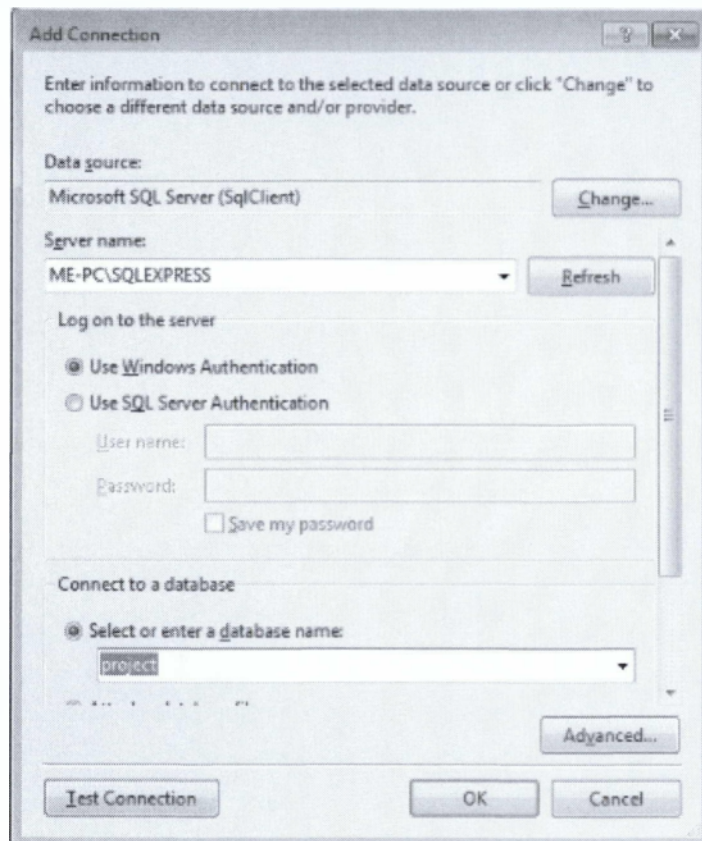
4.3 Δημιουργία της εφαρμογής

Σε αυτό το κεφάλαιο θα εξετάσουμε πως με το Visual Studio θα δημιουργήσουμε τις φόρμες που θα χρησιμοποιούν οι χρήστες. Με αυτές θα μπορεί ένας χρήστης να αποθηκεύει ή να κάνει αναζήτηση των στοιχείων που του είναι απαραίτητα και να συγχρονίζει τα δεδομένα με την εκάστοτε απομακρυσμένη βάση. Αρχικά ανοίγουμε την Visual Studio και επιλέγουμε να φτιάξουμε ένα νέο project. Στο Project type επιλέγουμε ότι η γλώσσα που θα χρησιμοποιήσουμε είναι η C# και το Template που θα δημιουργήσουμε είναι ένα Windows application. Το όνομα που του έχουμε δώσει είναι 'Contacts'. Πατώντας OK το Visual Studio θα δημιουργήσει ένα κενό Windows application (Εικόνα 28) που συναντήσαμε στην υποενότητα 3.2.3. Το επόμενο βήμα και πολύ σημαντικό είναι να συνδέσουμε την βάση δεδομένων που έχουμε δημιουργήσει, με το Visual Studio και την εφαρμογή. Από την menu bar επιλέγουμε Data -> Add new Data Sources και εμφανίζεται το παρακάτω (Εικόνα 23).



Εικόνα 43: Σύνδεση της βάσης δεδομένων με το Visual Studio

Μας δίνετε η δυνατότητα να επιλέξουμε τι θέλουμε να συνδέσουμε, Database, Web Service ή Object. Εμείς επιλέγουμε να συνδέσουμε την Database και πατάμε Next. Στην συνέχεια επιλέγουμε να κάνουμε νέα σύνδεση και πατώντας το κουμπί New Connection εμφανίζεται η εικόνα 44, όπου από εκεί θα διαλέξουμε τον Server που επιθυμούμε να συνδέσουμε.

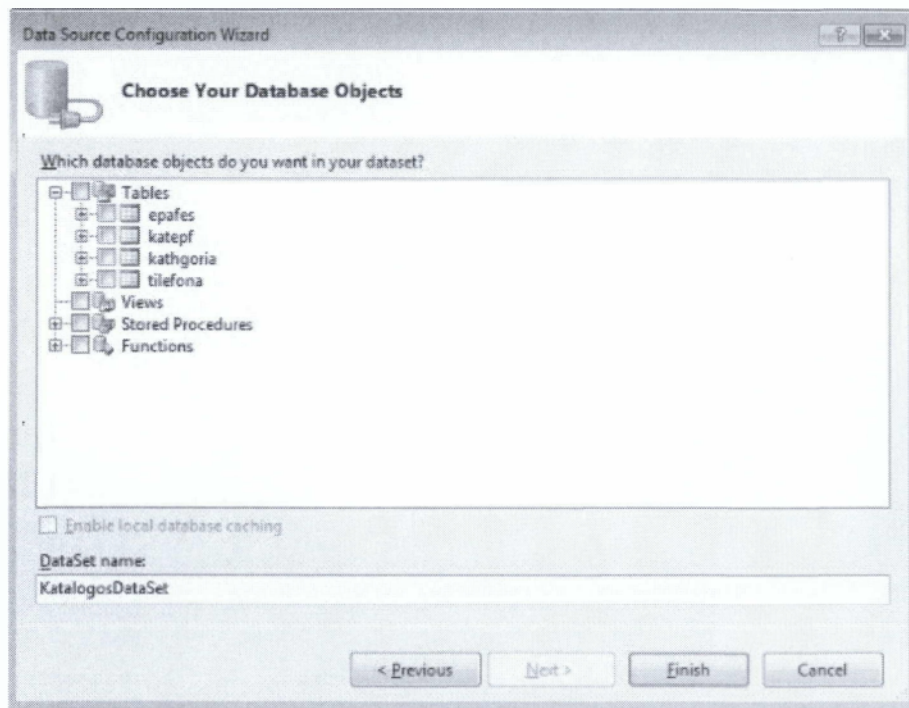


Εικόνα 44: Add Connection

Στο σημείο αυτό μας δίνονται οι πληροφορίες της σύνδεσης που θα έχει το project με τον Server. Στο πεδίο Data Source επιλέγουμε τον τύπο της βάσης με την οποία θα συνδεθεί η εφαρμογή. Επιλέγουμε Microsoft SQL Server. Πατώντας το κουμπί Change βλέπουμε τις άλλες επιλογές που έχουμε να διαλέξουμε. Στο Server Name βάζουμε το όνομα 'ME-PC\SQLEXPRESS' δηλαδή το όνομα που δώσαμε στον Server κατά την εγκατάσταση του SQL Server.

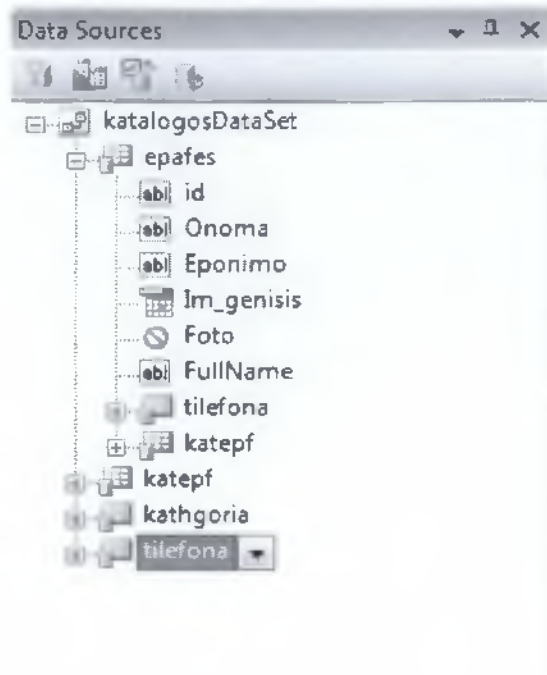
Από κάτω επιλέγουμε να χρησιμοποιείτε Windows Authentication, ενώ στο Select database Name τοποθετούμε το όνομα της βάσης που δημιουργήσαμε. Το όνομα της είναι 'project'. Πατώντας το κουμπί Test Connection μας δίνεται η δυνατότητα να ελέγξουμε αν η σύνδεση με την συγκεκριμένη βάση είναι εφικτή η

όχι. Πατάμε OK για να προχωρήσουμε. Στο επόμενο παράθυρο αφήνουμε τις ρυθμίσεις που υπάρχουν ήδη και πατάμε next. Κάνουμε το ίδιο αφήνοντας τις υπάρχοντες ρυθμίσεις και πατάμε next. Τέλος επιλέγουμε το “table” στο παράθυρο (εικόνα 25) αν θέλουμε να επιλεχθούν όλοι οι πίνακες και τα πεδία που έχουμε δημιουργήσει. Έχουμε την δυνατότητα να μην επιλέξουμε όλους τους πίνακες, αλλά κάποιους από αυτούς και μπορούμε να επιλέξουμε όλα τα procedures, τα views και τα functions που μας είναι απαραίτητα. Επίσης, πατάμε Finish και ολοκληρώνουμε την διαδικασία πρόσθεσης της βάσης στο project.



Εικόνα 45: Data Source

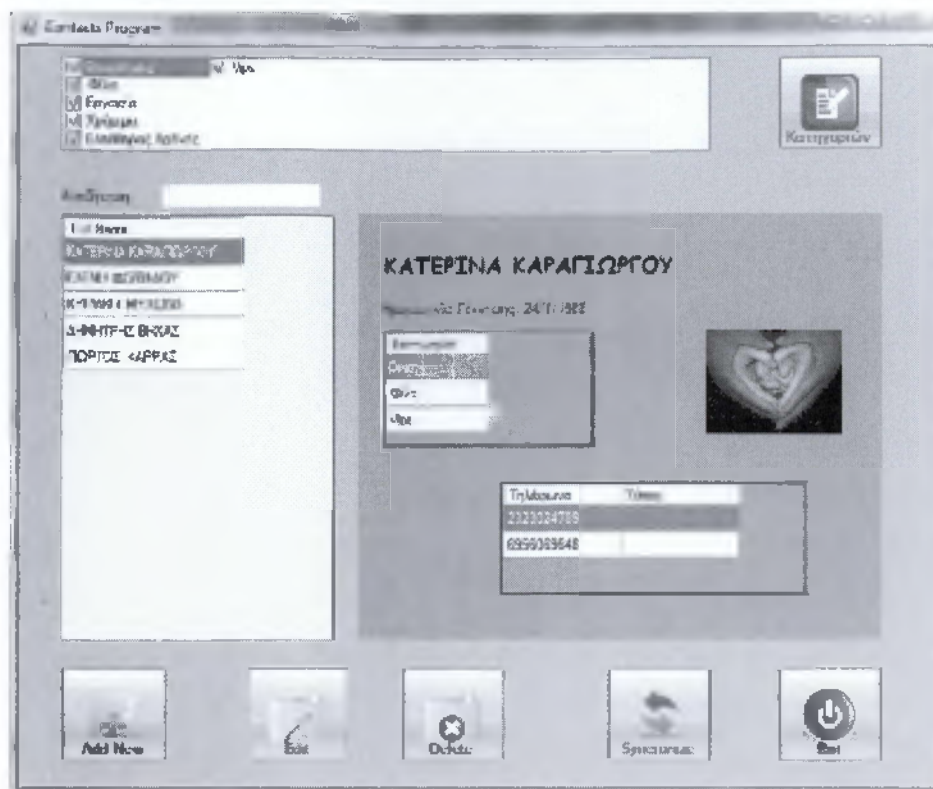
Εφόσον έχουμε ολοκληρώσει όλη την διαδικασία επιτυχώς στο Project έχουν προστεθεί τα απαραίτητα στοιχεία για την σύνδεση με τον Server και τη βάση ώστε να μπορούμε να επεξεργαζόμαστε τα δεδομένα του. Ήρθε η ώρα να φτιάξουμε το interface της εφαρμογής προσθέτοντας τα κατάλληλα κουμπιά και TextBox στην Windows Form μας. Αρχικά, από το menu bar επιλέγουμε Data → Show Data Source για εμφανιστεί το παράθυρο Data Source όπου μέσα βρίσκονται οι πίνακες και τα περιεχόμενα αυτών (Εικόνα 46).



Εικόνα 46: Εμφάνιση του Dataset.

Τώρα επιλέγουμε στο βελάκι δίπλα στον πίνακα epafes αν τα δεδομένα αυτού του πίνακα θα εμφανίζονται σαν DataGridView ή σαν Details. Εμείς διαλέγουμε την πρώτη. Το ίδιο κάνουμε με όλους τους πίνακες. Στην συνέχεια παίρνουμε τον πίνακα Epafes και με drag and drop το τοποθετούμε μέσα στην φόρμα. Βλέπουμε ότι στο Visual Studio δημιουργείτε ένας πίνακας σαν αυτό της βάσης δεδομένων (επιλέγουμε ποια κελιά θέλουμε με την βοήθεια του βέλους που υπάρχει δεξιά από το DataGridView που έχουμε στην φόρμα μας). Με την Navigator Bar μπορούμε να προσθέσουμε, να διαγράψουμε και να κινηθούμε σε όλες τις εγγραφές, όμως εμείς τη διαγράφουμε γιατί θα δημιουργήσουμε κουμπιά που θα κάνουν αυτές τις λειτουργίες.

Εκτός από τον πίνακα Epafes χρειάζεται να προσθέσουμε και τους άλλους μας πίνακες, γιατί η Main Form θα μας εμφανίσει στοιχεία και από τους άλλους πίνακες, όπως το τηλέφωνο, οι κατηγορίες κτλ.. Επίσης, θα χρησιμοποιήσουμε και άλλα εργαλεία στην WinForm, όπως κουμπιά και listbox, ώστε να γίνει λειτουργική. Όλα τα εργαλεία τα βρίσκουμε στο Toolbox Explorer (Εικόνα 29) όπου και αυτά με drag and drop προστίθενται στον κώδικα και τον πλαισιώνουν. Η πρώτη φόρμα που τα δημιουργούμε φαίνεται στην εικόνα 27.



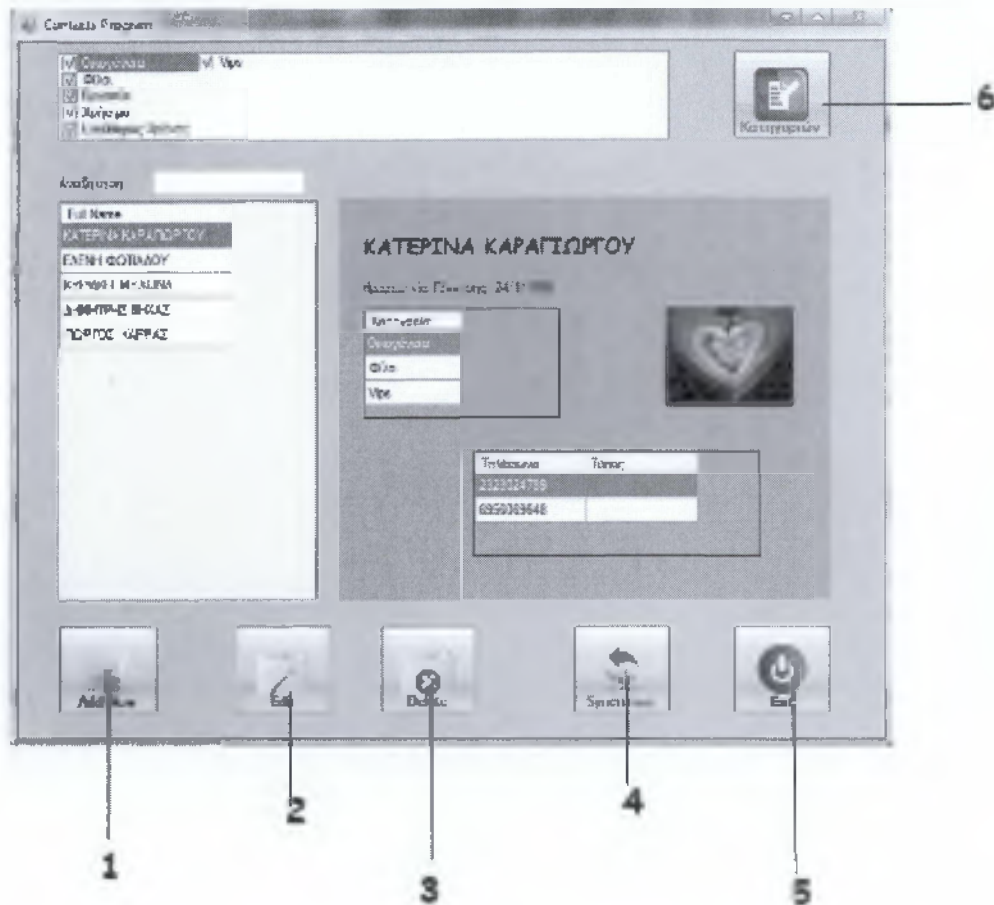
Εικόνα 47: WinForm Εμφάνιση Ατζέντας

Οι ιδιότητες του κάθε εργαλείου υπάρχουν στον Property Explorer αφού πρώτα έχουμε επιλέξει ένα από αυτά. Άρα, τα ονόματα των κουμπιών και της φόρμας τα επιλέγουμε καθώς και άλλες ιδιότητες (εικόνες, μέγεθος, χρώμα κτλ). Για να δούμε ή να επεξεργαστούμε τον ήδη υπάρχων κώδικα που δημιούργησε το Visual Studio με την προσθήκη των εργαλείων στην φόρμα, αρκεί να κάνουμε διπλό κλικ ή διαφορετικά πηγαίνουμε στο Solution Explorer κάνουμε δεξί κλικ στο όνομα της φόρμας και επιλέγουμε View Code. Και με τους δύο αυτούς τρόπους εμφανίζεται όχι μόνο ο κώδικας της φόρμας που έχουμε επιλέξει, αλλά και όλων των εργαλείων που έχουμε προσθέσει στην WinForm.

4.4 Εγχειρίδιο χρήσης

4.4.1 Βασικά κουμπιά της εφαρμογής

Τα βασικά κουμπιά της εφαρμογής φαίνονται στην επόμενη εικόνα:



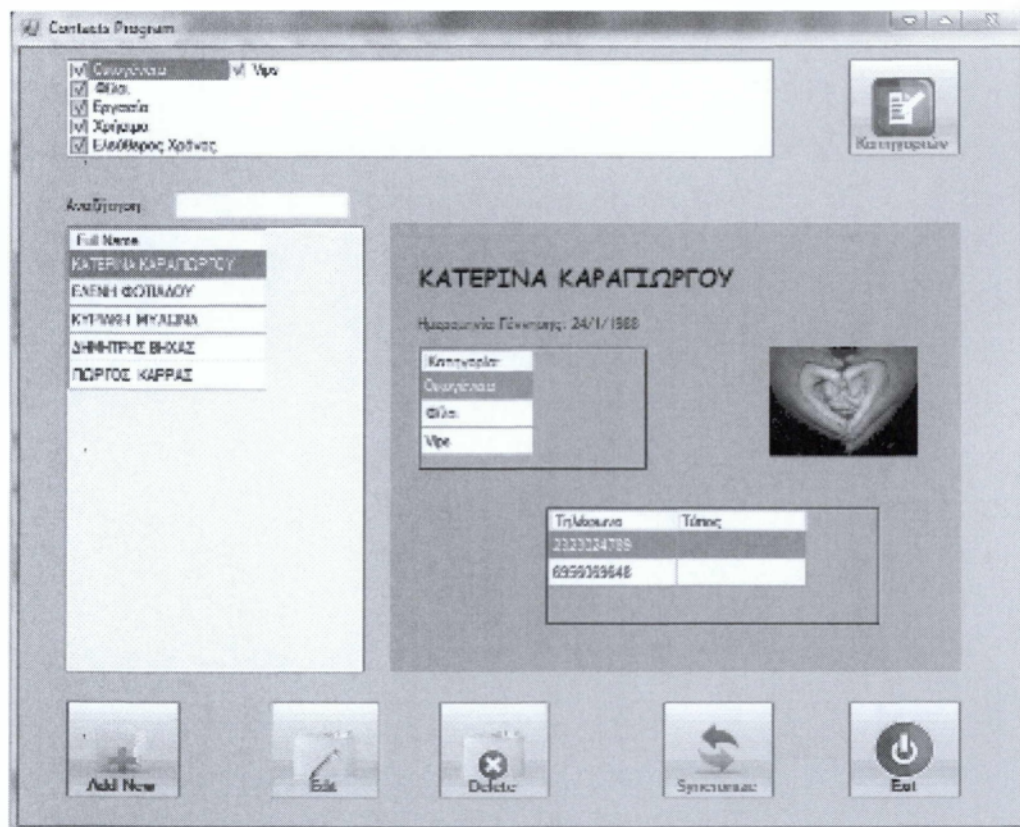
Εικόνα 48: Αρίθμηση βασικών κουμπιών

1. Πατώντας το κουμπί εισάγουμε μια νέα επαφή.
2. Το κουμπί αυτό μας βοηθάει να επεξεργαστούμε την εγγραφή που έχουμε επιλέξει.
3. Διαγράφουμε μια εγγραφή.
4. Πηγαίνουμε στην φόρμα συγχρονισμού.

5. Κλείνουμε την εφαρμογή.
6. Πατώντας το κουμπί πηγαίνουμε να επεξεργαστούμε τις κατηγορίες.

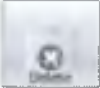
4.4.2 Επεξήγηση της λειτουργίας της εφαρμογής


Σε αυτό το κεφάλαιο θα δείξουμε στον χρήστη πως λειτουργεί αυτή η εφαρμογή. Αρχικά, αφού θα έχουμε εγκαταστήσει την εφαρμογή και την βάση δεδομένων στον σταθερό υπολογιστή του χρήστη και κάνουμε τις απαραίτητες ρυθμίσεις θα μπορεί να ξεκινήσει τη λειτουργία του.




Εικόνα 49: Αρχική φόρμα εισαγωγής βασικών στοιχείων στην ηλεκτρονική ατζέντα.

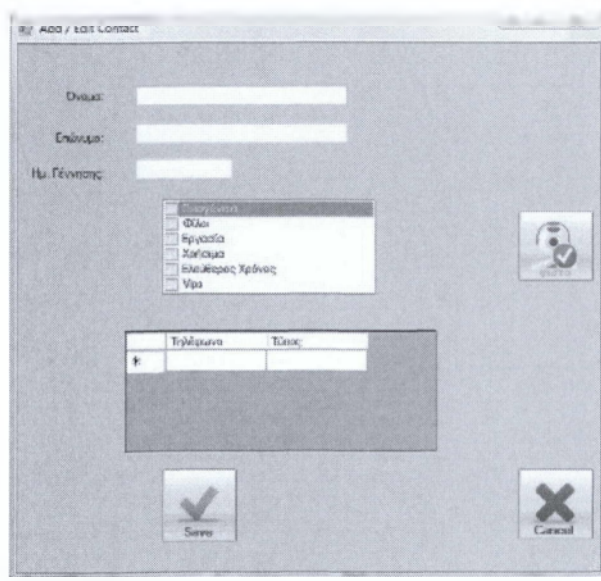
Μπαίνοντας στην εφαρμογή θα εμφανίζονται οι όποιες επαφές έχουν καταχωρηθεί. Υπάρχει η δυνατότητα να γίνει αναζήτηση των επαφών σύμφωνα με το επώνυμο (π.χ. όσους έχει με επίθετο Παπαδόπουλος), καθώς επίσης και να εμφανίσει στην λίστα του μόνο όσους ανήκουν σε κάποια ομάδα/κατηγορία για παράδειγμα την οικογένεια. Επιπλέον, έχει την επιλογή να σβήσει την εκάστοτε

επιλεγμένη επαφή πατώντας το κουμπί delete.  Πατώντας το κουμπί add

 έχει την δυνατότητα να προσθέσει μια επαφή (Εικόνα 50) ενώ με το κουμπί

edit  να επεξεργαστεί την επαφή που θέλει και έχει επιλέξει (Εικόνα 51).

Τέλος, το exit  είναι το κουμπί που τερματίζει τη φόρμα.




Εικόνα 50: Add Contact

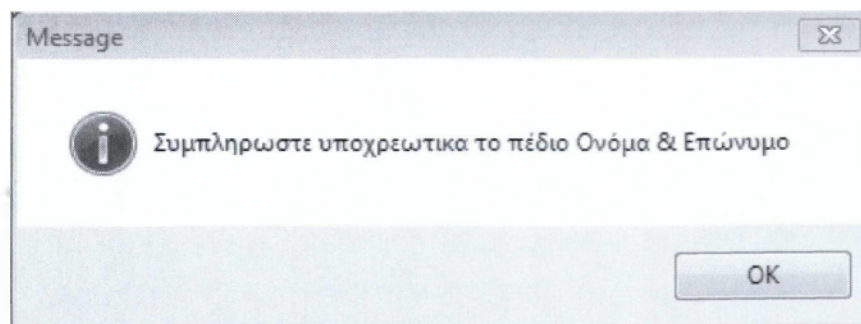


Εικόνα 51: Edit Contact

Επιλέγοντας ο χρήστης να προσθέσει ή να επεξεργαστεί μια επαφή θα του εμφανιστεί η αντίστοιχη φόρμα. Με το άνοιγμα της φόρμας υπάρχει η δυνατότητα πρόσθεσης ή τροποποίησης προσωπικών στοιχείων, καθώς και επιλογής μιας

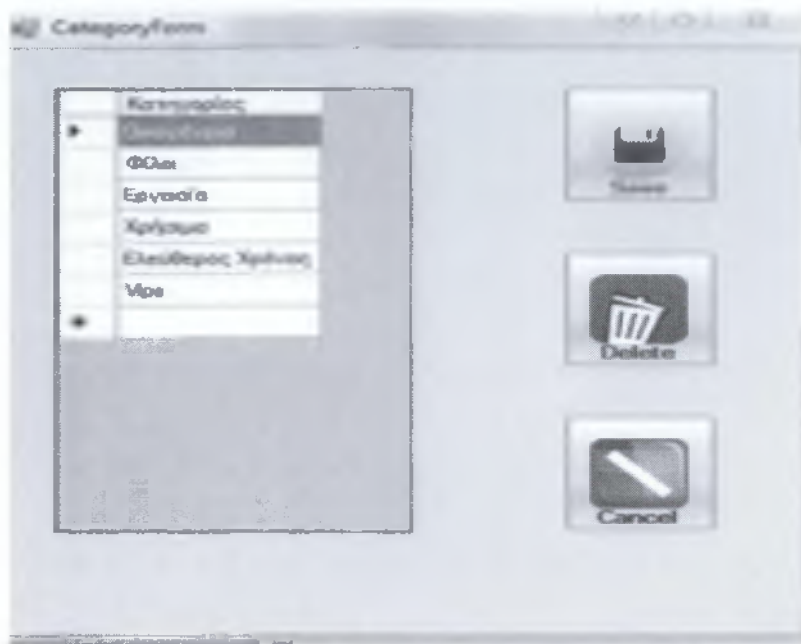
φωτογραφίας . Εφόσον, συμπληρωθούν/διαφοροποιηθούν τα στοιχεία

πατώντας το κουμπί αποθήκευσης  γίνεται και ο έλεγχος εάν έχουν καταχωρηθεί τα απαραίτητα πεδία (όνομα, επίθετο) και επιστρέφει στην αρχική του φόρμα. Στην περίπτωση που έχει αφήσει κενά τα απαραίτητα πεδία εμφανίζεται ένα Message box που παροτρύνει τον χρήστη να συμπληρώσει τα ανάλογα στοιχεία.



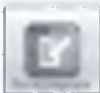
Εικόνα 52: Message box


Όταν επιθυμούμε να κλείσουμε τη φόρμα χωρίς κάποια αλλαγή πατάμε το κουμπί





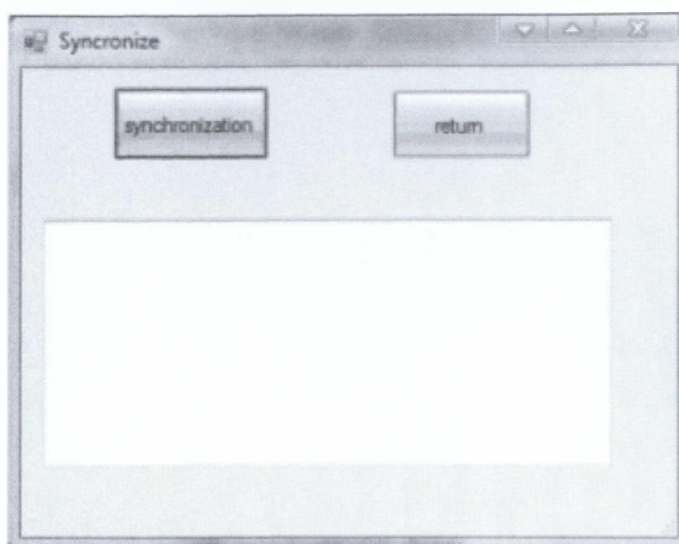
Εικόνα 53: Φόρμα Κατηγοριών

Εκτός από την καταχώρηση μιας επαφής ο χρήστης έχει την δυνατότητα να

προσθέσει επιπλέον και κατηγορίες πατώντας στο κουμπί 'κατηγορίες'  που βρίσκεται στην αρχική μας φόρμα (Εικόνα 49). Στη φόρμα 'Κατηγορίες' (Εικόνα 52) που εμφανίζεται έχουμε την δυνατότητα πρόσθεσης μια καινούργιας κατηγορίας και


έπειτα να την αποθηκεύσουμε πατώντας το κουμπί save . Με το κουμπί delete

 διαγράφουμε μια κατηγορία και με το cancel  γίνεται ακύρωση.

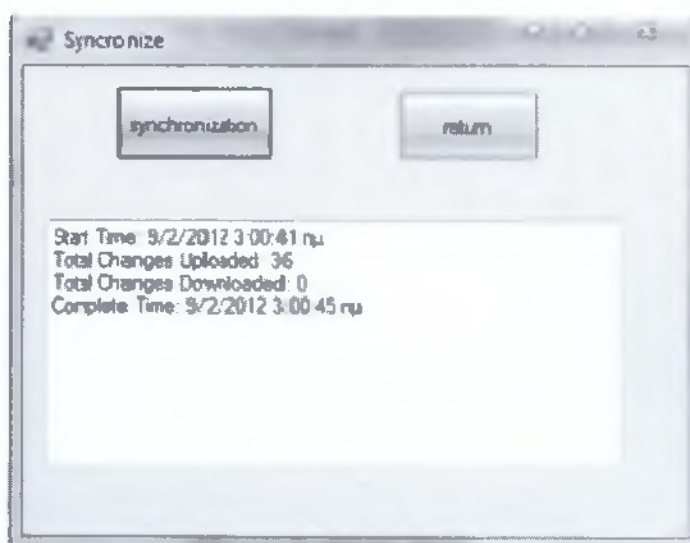


Εικόνα 54: Φόρμα Συγχρονισμού

Στην κεντρική φόρμα (Εικόνα 49) υπάρχει και η βασική λειτουργία του προγράμματος ο συγχρονισμός (Εικόνα 53). Ο χρήστης επιλέγοντας το αντίστοιχο

κουμπί  συγχρονίζει τα δεδομένα που βρίσκονται στον υπολογιστή του με οποιοδήποτε υπολογιστή έχει ορίσει. Για παράδειγμα εργάζεται και κάνει αλλαγές στον υπολογιστή που έχει σπίτι του και συγχρονίζει τις αλλαγές με τον υπολογιστή που έχει στο γραφείο. Στη φόρμα συγχρονισμού πατώντας το κουμπί synchronization

 εκτελείται ο συγχρονισμός των βάσεων (Εικόνα 54).



Εικόνα 55: Εκτελεσμένη Φόρμα Συγχρονισμού

Όταν θέλουμε να κλείσουμε τη φόρμα πατάμε return



4.5 Δημιουργία Query's

Αφού έχουμε προσθέσει την βάση δεδομένων και έχουμε δημιουργήσει την πρώτη WinForm στο project, ήρθε η ώρα να φτιάξουμε το Query που θα χρειαστούμε στον πίνακα Erafes για να φαίνεται στην ατζέντα μας το Ονοματεπώνυμο των επαφών μας. Όταν προσθέσαμε την βάση δεδομένων στο project αυτόματα στο Solution Explorer προστέθηκε το 'katalogosDataSet .xsd' όπου αυτό το αρχείο μας δείχνει τους πίνακες (tables) και τις συνδέσεις (Relations) αυτών. Κάνοντας δεξί κλικ επιλέγουμε 'Open' και ανοίγει το αρχείο 'katalogosDataSet.xsd' όπου εμφανίζονται σε γραφικό περιβάλλον (εικόνα 22) οι πίνακες και τα relations αυτών. Κάνουμε δεξί κλικ στο table adapter του πίνακα που επιθυμούμε επιλέγουμε Add → Query → Use SQL statements → next → Select which return rows και στο περιθώριο που μας δίνετε γράφουμε το Query που εμείς επιθυμούμε. Ο βασικός τύπος Query που χρησιμοποιούμε σε αυτό το project είναι ο ακόλουθος:

```
SELECT id, Onoma, Eponimo, Im_genisis, Foto, Onoma + ' ' + Eponimo
```

```
AS FullName FROM erafes
```

Στο παραπάνω Query χρησιμοποιούμε δυο βασικές εντολές SELECT και FROM. Μετά το SELECT δηλώνουμε τα πεδία από τα οποία θα πάρουμε εγγραφές και δημιουργούμε στην εφαρμογή μας το πεδίο FullName. Στο FROM δηλώνουμε το όνομα του πίνακα που βρίσκονται τα πεδία αυτά. Με λίγα λόγια το παραπάνω Query λέει να πάρουμε τις εγγραφές των πεδίων id, Onoma, Eponimo, Im_genisis, Foto (από τον πίνακα Erafes) και με το Onoma + ' ' + Eponimo AS FullName δημιουργούμε το πεδίο FullName δηλώνοντας ότι θα προέρχεται από το όνομα κενό και το επίθετο που υπάρχει στην βάση μας.

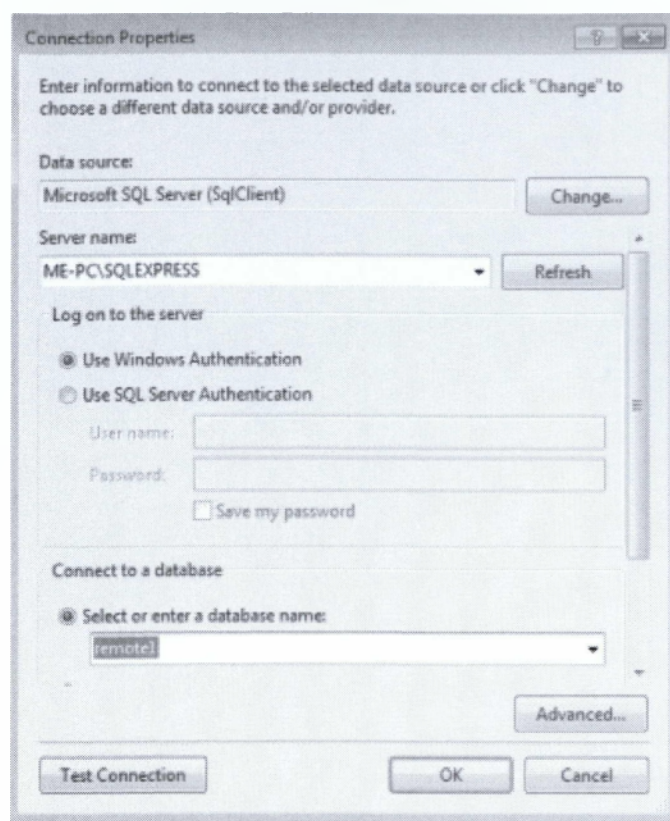
4.6 Ορισμός της απομακρυσμένης / εκσυγχρονιζόμενης βάσης.

Τελειώνοντας με όλα τα παραπάνω αυτό που μας έμεινε είναι να καθορίσουμε στην εφαρμογή μας ποια θα είναι και που θα βρίσκεται η δεύτερη βάση δεδομένων προκειμένου να μπορούν να συγχρονιστούν τα δεδομένα των δυο πινάκων. Ανοίγοντας το project μας στον Solution Explorer (Εικόνα 31) επιλέγουμε το properties πηγαίνουμε στην καρτέλα settings και εμφανίζεται το ακόλουθο παράθυρο (Εικόνα 55).

Name	Type	Scope	Value
Project	(Connectio...	Application	Data Source=MS-PC\SQLEXPRESS;Initial Catalog=Project1;Integrated Security=True
Remote	(Connectio...	Application	Data Source=MS-PC\SQLEXPRESS;Initial Catalog=remote;Integrated Security=True

Εικόνα 56:Application Settings

Στη συνέχεια επιλέγουμε τη δεύτερη επιλογή, κάνουμε κλικ στις λεπτομέρειες εμφανίζεται η εικόνα 56 και διαλέγουμε ποια βάση θέλουμε (αρκεί να έχει ίδια μορφή ή να είναι κενή και αυτόματα θα δημιουργηθεί το πρότυπο το οποίο θα προσθέσει ανάλογους πίνακες, δεδομένα κτλ).



Εικόνα 57: Connection Properties

Την προηγούμενη διαδικασία έχουμε την δυνατότητα να την πραγματοποιήσουμε και από τον κώδικά μας. Στο αρχείο app.config υπάρχουν οι ανάλογες εντολές που εμφανίζονται στη εικόνα 55. Πατώντας στο app.config μας βγάζει τον κώδικα . Οι εντολές που μας ενδιαφέρουν και πρέπει να αλλάξουμε είναι οι παρακάτω:

1. `<add name="Contacts.Properties.Settings.Project" connectionString="Data Source=ME-PC\SQLEXPRESS;Initial Catalog=Project1;Integrated Security=True" providerName="System.Data.SqlClient" />`
2. `<add name="Contacts.Properties.Settings.Remote" connectionString="Data Source=ME-PC\SQLEXPRESS;Initial Catalog=remote1;Integrated Security=True" providerName="System.Data.SqlClient" />`

όπου στο Data Source καθορίζεται το όνομα του sqlserver και στο Initial Catalog το όνομα της βάσης που χρησιμοποιούμε. Η εντολή επαναλαμβάνεται γιατί η πρώτη

ρύθμιση αφορά στην τοπική βάση, ενώ η δεύτερη στην απομακρυσμένη με την οποία γίνεται ο συγχρονισμός. Στο παράδειγμα παραπάνω το Data Source είναι ίδιο αν και κάτι τέτοιο δεν είναι απαραίτητο αφού η απομακρυσμένη βάση μπορεί να βρίσκεται οπουδήποτε.

Το `app.config` είναι ένα αρχείο ρυθμίσεων που συνοδεύει την εφαρμογή. Βασικό πλεονέκτημα είναι η δυνατότητα παραμετροποίησης της βάσης που χρησιμοποιούμε (καθώς και πολλών άλλων ρυθμίσεων) χωρίς να χρειάζεται να μεταγλωττίσουμε (`compile`) ξανά το πρόγραμμα.

Παράρτημα Α: Κώδικας Εφαρμογής

Παρακάτω παρατίθεται ο κώδικας που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής. Δεν περιλαμβάνεται ο κώδικας που δημιουργείται αυτόματα μέσω του γραφικού περιβάλλοντος (φόρμες, στοιχεία ελέγχου, κ.λπ.).

Κώδικας Main

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

namespace Contacts
{
    public partial class MainForm : Form
    {
        private string _categoryFilter = String.Empty; //Το φίλτρο
        //βάση τις επιλεγμένες κατηγορίες.
        private string _nameFilter = String.Empty; // Το φίλτρο βάση
        //τα στοιχεία που βάζουμε στην αναζήτησης του ονόματος.

        public MainForm()
        {
            InitializeComponent();

            // η συνάρτηση main load , περιέχει 3 συναρτήσεις που
            //θέλουμε να τρέχουν κάθε φορά που ανοίγει το προγράμμα μας
            private void Main_Load(object sender, EventArgs e)
            {
                BoundCategoryListBox(); // γεμίζει το Listbox κατηγοριών
                LoadTableAddapters(); // γεμίζω-φορτώνω τους πίνακες μου
                ListBoxSelectAllCategories(); // τσεκάρει όλες τις
                //κατηγορίες μου
            }

            //στην ουσία "γεμίζει" το categories listbox με τις
            //κατηγορίες απο την βάση
            private void BoundCategoryListBox()
            {
                this.categoriesListBox.DataSource =
                this.allCategoriesBindingSource;
                this.categoriesListBox.DisplayMember = "onoma_kat";
                this.categoriesListBox.ValueMember = "id_kat";
            }
            // φορτώνει τα δεδομένα
        }
    }
}
```

```

private void LoadTableAddapters()
{
this.epafesTableAdapter.Fill(this.katalogosDataSet.epafes);
this.tilefonaTableAdapter.Fill(this.katalogosDataSet.tilefona);
this.kathgoriaTableAdapter.Fill(this.katalogosDataSet.kathgoria);
this.katepfTableAdapter.Fill(this.katalogosDataSet.katepf);
}
// τσεκάρει όλες μου τις κατηγορίες
private void ListBoxSelectAllCategories()
{
    for (int i = 0; i < categoriesListBox.Items.Count; i++)
        categoriesListBox.SetItemChecked(i, true);
}

#region Business Logic

// κάνει την αναζήτηση σύμφωνα με το επώνυμο (στα δεδομένα
που έχω στο επαφες )
private void FilterContactsByName()
{
    _nameFilter = "Eponimo like " + searchTextBox.Text +
    epafesBindingSource.Filter = _nameFilter;
}

// φιλτράρει τις επαφες βάσει των κατηγοριών
private void FilterContactsByCategory()
{
}

this.epafesTableAdapter.FillByCategory(this.katalogosDataSet.epafes,
_categoryFilter);
}

//δίνει στο _categoryFilter , όλες τις κατηγορίες που έχω
επιλέξει χωρισμένες με κόμμα
private void UpdateCategoryFilter()
{
}

//δημιουργεί μια λίστα και την γεμίζει από την συνάρτηση
dimiourgi getselectedcategories
List<int> selectedCategories = GetSelectedCategories();
//ελέγχουμε εαν έχει γεμίσει ,εαν δεν έχει γίνει κάποια
επιλογή κατηγοριών το φίλτρο είναι άδειο
if (selectedCategories.Count == 0)
{
    _categoryFilter = String.Empty;
}
else
{
    //δημιουργεί το φίλτρο με τις κατηγορίες που θέλουμε
StringBuilder fb = new StringBuilder("");
// γεμίζει το fb με το πρώτο Item της κατηγορίας (τα
Id με κομμάτια)
fb.Append(String.Format("{0}",
selectedCategories[0]));
// There exists at least one category. Add it.
for (int i = 1; i < selectedCategories.Count; i++)

```



```

        {
            fb.Append(String.Format(", {0}",
selectedCategories[i]));
        }
        _categoryFilter = fb.ToString();
    }
}

//Δημιουργεί μια λίστα με τα id των κατηγοριών που είναι
επιλεγμένες
private List<int> GetSelectedCategories()
{
    List<int> selectedCategories = new List<int>();
    //Δημιουργεί την var checkedCategories κ κρατάει όλα τα
τεσκαρισμένα
    var checkedCategories = categoriesListBox.CheckedItems;
    //για κάθε τεσκαρισμένη κατηγορία προσθέτει στην λίστα
selectedCategories το id kat
    foreach (DataRowView rv in checkedCategories)
    {
        if (rv.Row.RowState != DataRowState.Detached)
            selectedCategories.Add((int)rv["id_kat"]);
    }
    // κ τέλος επιστρέφει το αποτέλεσμα που θα είναι μια
λίστα με 5 αριθμούς(πχ)
    return selectedCategories;
}
//εφαρμόζει το φιλτράρισμα των κατηγοριών. εάν όλες οι
//κατηγορίες είναι επιλεγμένες φέρνει όλες τις επαφές
// εάν κάποιες αποεπιλεγμένες καλεί τα φίλτρα της κατηγορίας

private void DoCategoryFilter()
{
    int checkedCategories =
categoriesListBox.CheckedItems.Count;
    int AllCategories = categoriesListBox.Items.Count;
    if (AllCategories == checkedCategories)

this.epafesTableAdapter.Fill(this.katalogosDataSet.epafes);
    else
    {
        UpdateCategoryFilter();
        FilterContactsByCategory();
    }
}

#endregion

#region Form Control Events

// δημιουργεί μια καινούργια επαφή και φορτώνει στην φόρμα
add/edit για
//να την επεξεργαστούμε
private void BtnAdd_Click(object sender, EventArgs e)
{
    int current = epafesBindingSource.Position;// κρατάει μια
θέση/ επαφή έχω επιλεγμένη πχ. το Κατερίνα σαν δείκτη

```

```

Object newContact = epafesBindingSource.AddNew();//
προετοιμάζει την βάση για να προσθεση μια επαφή(φτιάχνει μια νέα
επαφή)
    AddEditContact addForm = new
AddEditContact(newContact);// δημιουργώ την φόρμα add και έχω κάνει
ένα νέο αντικείμενο για να προσθεσω
    addForm.myResult += new
EventHandler(ExecuteActionFromAddEditContact);//this will be trigger
when child form closes
    addForm.ShowDialog(this);
}
// μας μεταφέρει στην φόρμα add/edit και εμφανίζει τα
στοιχεία της επαφης που διαλέξαμε
private void btnEdit_Click(object sender, EventArgs e)
{
    Object currContact = epafesBindingSource.Current;
//αντιστοιχει τον δεικτη Current με τον αντικειμενο currContact
    AddEditContact editForm = new
AddEditContact(currContact);// παίρνει τον αριθμο currContact (το
αντικειμενο ) και τον περνάει στην φόρμα για να μας εμφανίσει αυτο
που επιλέξαμε
    editForm.myResult += new
EventHandler(ExecuteActionFromAddEditContact);//this will be trigger
when child form closes
    editForm.ShowDialog(this);
}
// κάθε φορά που κλείνω το πρόγραμμα μου τρέχει αυτη η
συνάρτηση.
private void MainForm_FormClosing(object sender,
FormClosingEventArgs e)
{
    // ξετσεκάρω τις κατηγορίες και αποσυνδέω το event
katepfDataGridView.DataSource = null;
categoriesListBox.MouseUp -= categoriesListBox_MouseUp;
}
// αυτη η συνάρτηση διαγράφει την επαφη απο την εφαρμογη
και την βάση
private void btnDelete_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Είσαι σίγουρος οτι
θέλεις να διαγραφει", "Διαγραφή Επαφης", MessageBoxButtons.OKCancel);

    if (result == DialogResult.OK)
    {
        epafesBindingSource.RemoveCurrent();//προετοιμάζουμε
την πηγη/βάση για την διαγραφη της επιλεγμένης επαφής
        this.Validate();
        this.epafesBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.katalogosDataSet);
        this.epafesBindingSource.MoveFirst(); //Μας πάει στην
πρώτη επαφή αφού η άλλη έχει σβήσει( δεν μπορούμε κα την
//επεξεργαστουν)πρέπει αν έχει ένα σημαιο να
επιστρέψει
    }
}
//Τερματίζει την εφαρμογή μου
private void btnClose_Click(object sender, EventArgs e)
{
    Application.Exit();
}

```

```

}
// Μας μεταφέρει στην φόρμα category
private void buttonCategory_Click(object sender, EventArgs e)
{
    CategoryForm EditCategory = new CategoryForm();
    EditCategory.myResult += new
EventHandler(ExecuteActionFromCategory);
    EditCategory.ShowDialog(this);
}
//τρέχει κάθε φορά που αλλάζει κάτι στο πεδίο αναζήτησης
private void searchTextBox_TextChanged(object sender,
EventArgs e)
{
    FilterContactsByName();
}
// τρέχει κάθε φορά που κάνω αλλαγες στα τικ της κατηγορίας

#endregion

```

#region Event Handlers from Child Forms

```

//αυτή η συνάρτηση καλείται κάθε φορά που κλείνουμε την
add/edit
private void ExecuteActionFromAddEditContact(object sender,
EventArgs e)
{
    UserAction result = (UserAction)e;// cast the eventargs
to read the message send from child form
    if (result.WantToSaveInfo)
    {
        // ο χρήστης πάτησε το save
        this.Validate();
        this.epafesBindingSource.EndEdit();
        this.tilefonaBindingSource.EndEdit();
        // First update katepfs and then all the rest
        this.epafesTableAdapter.Update(this.katalogosDataSet);
        this.katepfTableAdapter.Update(this.katalogosDataSet);

this.tilefonaTableAdapter.Update(this.katalogosDataSet);

//this.tableAdapterManager.UpdateAll(this.katalogosDataSet);
        DoCategoryFilter();
    }
    else
    {
        // ο χρήστης πάτησε το cancel
        katalogosDataSet.RejectChanges();
        DataRowView current =
(DataRowView)epafesBindingSource.Current;
        if (current.IsNew)
        {
            current.Delete();
        }
    }
}

```

----- ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ -----

```
//εκτελείται κάθε φορά που κλείνω την categoryForm
private void ExecuteActionFromCategory(object sender,
EventArgs e)
{
    //Need to reload all info, otherwise if you deleted a
category and try to edit
    //a contact with that category this program fails...

    //Detach event handler so avoid unnecessary filters on
contacts by Category
categoriesListBox.MouseUp -= categoriesListBox_MouseUp;
//Reload All Info
LoadTableAddapters();
//set category filter to all existing categories
ListBoxSelectAllCategories();
//Attach event handler on filter contacts by Category
categoriesListBox.MouseUp += categoriesListBox_MouseUp;
}

#endregion

// Μας μεταφέρει στην φόρμα συγχρονισμού
private void btnSync_Click(object sender, EventArgs e)
{
    Sync _Sync = new Sync();
    _Sync.GoBackToParent += new EventHandler(ToDoAfterSync);
    _Sync.ShowDialog(this);
}

//This function is trigger when the child form send the
message (Event Handler)
//εκτελείται κάθε φορά που επιστρέφουμε από την φόρμα
συγχρονισμού

private void ToDoAfterSync(object sender, EventArgs e)
{
    //here you add the actions you want afetr child is done

    // Detached the event handler of the Category listbox
categoriesListBox.MouseUp -= categoriesListBox_MouseUp;

    //load again all tables
LoadTableAddapters();

    //Select All Categories
ListBoxSelectAllCategories();

    //Re-attach the event handler of the category listbox
categoriesListBox.MouseUp += categoriesListBox_MouseUp;
}
```

----- ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ -----

```
        //καλείται όταν ο χρήστης επιλέγει η αποεπιλέγει μια
κατηγορία
private void categoriesListBox_MouseUp(object sender,
MouseEventArgs e)
    {
        DoCategoryFilter();
    }
}
}
```

Κώδικας add/editform

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Contacts
{
    public partial class AddEditContact : Form
    {
        public EventHandler myResult;
        private UserAction wantToSave = new UserAction();
        //Constructor
        public AddEditContact()
        {
            InitializeComponent();

            //Constructor με παράμετρο (το αντικείμενο που θα
            επεξεργαστώ)
            public AddEditContact(Object source)
            {
                InitializeComponent();

                this.kathgoriaTableAdapter.Fill(this.katalogosDataSet.kathgoria);
                epafesBindingSource.DataSource = source;
                SetupCategories();
            }

            // αποθηκεύει τις αλλαγές και επιστρέφει
            private void btnSave_Click(object sender, EventArgs e)
            {
                // ελέγχει ένα έχω συμπληρώσει τα πεδία και πετάει το μήνυμα
                if (onomaTextBox.Text == "" || eponimoTextBox.Text == "" )
                {
                    MessageBox.Show("Συμπληρώστε υποχρεωτικά το πεδίο Ονόμα &
                    Επώνυμο", "Message",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
                    onomaTextBox.Focus();
                    return;
                }

                wantToSave.WantToSaveInfo = true;
                this.Close();
            }
            //ακυρώνει τις όποιες ενεργειές
            private void btnCancel_Click(object sender, EventArgs e)
            {
                wantToSave.WantToSaveInfo = false;
                this.Close();
            }
        }
    }
}

```

```
// ολοκληρώνει την επεξεργασία της επαφής , κλείνει την φόρμα και  
ειδοποιεί ένα ο χρήστη θέλει να σώσει τις αλλαγές ή όχι
```

```
private void EditContact_FormClosed(object sender,  
FormClosedEventArgs e)  
{  
    this.Validate();  
    onomaTextBox.Focus();  
    this.epafesBindingSource.EndEdit();  
    this.epafesBindingSource.MoveFirst();  
    this.tilefonaBindingSource.EndEdit();  
    this.Hide();  
    if (wantToSave != null)  
    {  
  
this.tableAdapterManager.UpdateAll(this.katalogosDataSet);  
        myResult(this, wantToSave); //this is the way to  
communicate with Parent form  
    }  
}
```

```
//ανοίγει το παραθύρο για να επιλεξουμε την φωτογραφία
```

```
private void uploadbutton_Click(object sender, EventArgs e)  
{  
  
    if (openFileDialog1.ShowDialog() == DialogResult.OK)  
    {  
        string imageLocation = openFileDialog1.FileName;  
        fotoPictureBox.Image = new  
Bitmap(openFileDialog1.FileName);  
  
    }  
}
```

```
// καλεί της συναρτήσεις για να τσεκάρει και γεμίζει τις κατηγορίες  
private void SetUpCategories()
```

```
{  
    BoundCategoryListBox();  
    SelectCategories();  
    categoriesListBox.ItemCheck += new  
ItemCheckedEventHandler(categoriesListBox_ItemCheck);  
}
```

```
//στην ουσία "γεμίζει" το categories listbox με τις  
κατηγορίες απο την βάση
```

```
private void BoundCategoryListBox()  
{  
    this.categoriesListBox.DataSource =  
this.categoriesBindingSource;  
    this.categoriesListBox.DisplayMember = "onoma_kat";  
    this.categoriesListBox.ValueMember = "id_kat";  
}
```

----- ΠΙΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ -----

```
// τσεκάρει τις κατηγορίες που ανήκουν στην επαφή
private void SelectCategories()
{
    var rowView = epafesBindingSource.Current as DataRowView;
// current contact

    // if current row is not null, find out selected
categories
    if (rowView != null)
    {
        // Get the katepf list.
        var katepfs =
((katalogosDataSet.epafesRow)rowView.Row).GetkatepfRows();
        int index;
        // for each katepf check the corresponding category.
        foreach (var katepf in katepfs)
        {
            index = GetIndexOfCategory(katepf.id_kat);
            if (index != -1)
                categoriesListBox.SetItemChecked(index,
true);
        }
    }

    // Gets the index in the categoriesListBox of the category
with the specified ID.
    private int GetIndexOfCategory(int categoryID)
    {
        int index = 0;
        bool keepSearching = (index <
categoriesListBox.Items.Count);
        katalogosDataSet.kathgoriaRow category;

        // While we have not found the id and we have not run out
of items.
        while (keepSearching)
        {
            // Get the category from the list box and compare its
id with categoryID.
            category = (katalogosDataSet.kathgoriaRow)
((DataRowView)categoriesListBox.Items[index]).Row;
            keepSearching = (category.id_kat != categoryID);

            // If ids are equal, we found it and need not search
anymore.
            // Otherwise, move to the next item.
            if (keepSearching)
            {
                index++;
                keepSearching = (index <
categoriesListBox.Items.Count);
            }
        }

        // If index >= count we ran out of elements (e.g. the
item was not found).
        if (index >= categoriesListBox.Items.Count)
```



```
        index = -1;
        return index;
    }

    // Called when an item's check state is about to be changed.
    void categoriesListBox_ItemCheck(object sender,
ItemCheckEventArgs e)
    {
        CheckedListBox clb = (CheckedListBox)sender;
        bool checkValue = (e.NewValue == CheckState.Checked);

        if (checkValue)
        {
            // A new category has been checked. Add a new katepf
entry.
            DataRowView rowView =
(DataRowView)katepfBindingSource.AddNew();
            var katepf = (katalogosDataSet.katepfRow)rowView.Row;
            katepf.id_kat = (int)clb.SelectedValue;
        }
        else
        {
            // A category has been unchecked. Remove the
corresponding katepf entry.
            int index = katepfBindingSource.Find("id_kat",
clb.SelectedValue);
            if (index != -1)
                katepfBindingSource.RemoveAt(index);
        }

        katepfBindingSource.EndEdit();
    }
}
}
```

Κώδικας Category Form

```

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

namespace Contacts
{
    public partial class CategoryForm : Form
    {
        public EventHandler myResult;

        public CategoryForm()
        {
            InitializeComponent();

            // Αποθηκεύει τις αλλαγές που έχουμε κάνει στην φόρμα μας
            private void kathgoriaBindingNavigatorSaveItem_Click(object
sender, EventArgs e)
            {
                this.Validate();
                this.kathgoriaBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.katalogosDataSet);

            }

// φορτώνει τα δεδομένα
            private void CategoryForm_Load(object sender, EventArgs e)
            {

                this.kathgoriaTableAdapter.Fill(this.katalogosDataSet.kathgoria
);

            }

//Κάνει αποθηκεύση των αλλαγών και ενημερώνω και το dataset μου
            private void btnSave_Click(object sender, EventArgs e)
            {
                this.Validate();
                this.kathgoriaBindingSource.EndEdit();
                this.kathgoriaTableAdapter.Update
                (this.katalogosDataSet.kathgoria);
                this.Close();
            }

// διαγράφει την επαφή που θέλω
            private void btnDelete_Click(object sender, EventArgs e)
            {
                DialogResult result = MessageBox.Show("Είσαι σίγουρος ότι
θέλεις να διαγράψεις", "Διαγραφή Επαφής",
                MessageBoxButtons.OKCancel);

                if (result == DialogResult.OK)

```

```
        {
            kathgoriaBindingSource.RemoveCurrent();
            this.Validate();
            this.kathgoriaBindingSource.EndEdit();
        }
        this.kathgoriaTableAdapter.Update(this.katalogosDataSet.kathgoria);
    }

    //ακυρώνει τις οποίες εργασίες έχω κάνει
    private void btnCancel_Click(object sender, EventArgs e)
    {
        myResult = null;
        this.Close();
    }

    //κλείνει την φόρμα
    private void CategoryForm_FormClosed(object sender,
    FormClosedEventArgs e)
    {
        {
            if (myResult != null)
            {
                myResult(this, null);
            }
        }
    }
}
```

Κώδικας φόρμας Sync

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.Synchronization;
using Microsoft.Synchronization.Data;
using Microsoft.Synchronization.Data.SqlServer;

namespace Contacts
{
    public partial class Sync : Form
    {
        //define event handler. It manages communication between
        forms Parent <-->Child

        public EventHandler GoBackToParent;
        // parent form will activate this handler when initializing
        the child
        //it is like telling the child, "when you finish let me know
        so i can do 'something'

        public Sync()
        {
            InitializeComponent();
        }

        private void returnButton_Click(object sender, EventArgs e)
        {
            //when you close the form you want parent form to update..
            so
            //in the close form event to trigger the event handler
            this.Hide();
            this.Close();
        }

        private void Sync_FormClosed(object sender,
        FormClosedEventArgs e)
        {
            //makes sure the handler is not null
            if (GoBackToParent != null)
            {
                GoBackToParent(this, null);
            }
        }

        // Οποιαδήποτε αλλαγή στην σύνδεση γίνεται (δλδ άλλον server το
        αλλάζουμε στο
    
```

```

// app.config η τις ιδιότητες του Project...in section
settings(αυτο το κάναμε για να μην πρέπει
//να αλλάζουμε και εδώ το όνομα του εκάστοτε υπολογιστή ή της
δεύτερης βάσης )
private string stgProjectConn =
Contacts.Properties.Settings.Default.Project;
private string stgRemoteConn =
Contacts.Properties.Settings.Default.Remote;

//Εκείνα τον συγχρονισμό των βάσεων

private void syncButton_Click(object sender, EventArgs e)
{
// Σύνδεση με parent server είναι το όνομα του server's
(διακομιστή)
SqlConnection serverConn = new
SqlConnection(stgProjectConn);

// Scope the parent server (prot;ypo perigrafes pinakon
..to sxhma toy)
DbSyncScopeDescription scopeDesc = new
DbSyncScopeDescription("SyncScope");

DbSyncTableDescription tableDesc_1 =
SqlSyncDescriptionBuilder.GetDescriptionForTable("epafes",
serverConn);
scopeDesc.Tables.Add(tableDesc_1);

DbSyncTableDescription tableDesc_2 =
SqlSyncDescriptionBuilder.GetDescriptionForTable("katepf",
serverConn);
scopeDesc.Tables.Add(tableDesc_2);

DbSyncTableDescription tableDesc_3 =
SqlSyncDescriptionBuilder.GetDescriptionForTable("kathgoria",
serverConn);
scopeDesc.Tables.Add(tableDesc_3);

DbSyncTableDescription tableDesc_4 =
SqlSyncDescriptionBuilder.GetDescriptionForTable("tilefona",
serverConn);
scopeDesc.Tables.Add(tableDesc_4);

// Provision the parent server;
SqlSyncScopeProvisioning serverProvision = new
SqlSyncScopeProvisioning(serverConn, scopeDesc);

serverProvision.SetCreateTableDefault(DbSyncCreationOption.Skip);

if (!serverProvision.ScopeExists("SyncScope"))
{
serverProvision.Apply();
}

// Σύνδεση με τον άλλον server μου ( την άλλη συσκευή που
βρίσκεται η βάση) εφαρμογή scope και provision

```

```

        SqlConnection clientConn = new
SqlConnection(stgRemoteConn);
        DbSyncScopeDescription clientScopeDesc =
SqlSyncDescriptionBuilder.GetDescriptionForScope("SyncScope",
serverConn);
        SqlSyncScopeProvisioning clientProvision = new
SqlSyncScopeProvisioning(clientConn, clientScopeDesc);
        if (!clientProvision.ScopeExists("SyncScope"))
        {
            clientProvision.Apply();
        }

        // προετοιμασία συγχρονισμού και καταγραφή για το τι
αλλαγές θα γίνουν (περιληπτικά)
        SyncOrchestrator syncOrchestrator = new
SyncOrchestrator();
        syncOrchestrator.LocalProvider = new
SqlSyncProvider("SyncScope", serverConn);
        syncOrchestrator.RemoteProvider = new
SqlSyncProvider("SyncScope", clientConn);
        // μια ποια σειρά θα γίνει ο sin/ronismo(poia timi tha
krathsei ama ginei allagh kai stis 2 bashs sto idio stoixeio
        syncOrchestrator.Direction = SyncDirectionOrder.UploadAndDownload;

// συγχρονισμός και καταγραφή στατιστικών
        SyncOperationStatistics syncStats =
syncOrchestrator.Synchronize();

// εμφάνιση στατιστικών στο textbox
        textBox1.Text = "Start Time: " + syncStats.SyncStartTime
+ Environment.NewLine;
        textBox1.Text += "Total Changes Uploaded: " +
syncStats.UploadChangesTotal + Environment.NewLine;
        textBox1.Text += "Total Changes Downloaded: " +
syncStats.DownloadChangesTotal + Environment.NewLine;
        textBox1.Text += "Complete Time: " +
syncStats.SyncEndTime + Environment.NewLine;
    }
}
}

```

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Wrox Press, Professional C#, 3rd Edition
- “Microsoft Visual C# 2008”, Βήμα βήμα του John Sharp
- Οοδηγός της C# 3.0 του Herbert Schildt
- <http://www.dmst.aueb.gr/louridas/lectures/dais/architecture/ar01s06.html>
- <http://envygismo.blogspot.com/2009/07/client-server-3-3-tier.html>
- http://en.wikipedia.org/wiki/Client-server_model
- http://en.wikipedia.org/wiki/Multitier_architecture
- <http://en.wikipedia.org/wiki/Peer-to-peer>
- http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
- <http://msdn.microsoft.com/en-us/sync/bb736753>
- [http://msdn.microsoft.com/en-us/library/bb726031\(SQL.90\).aspx](http://msdn.microsoft.com/en-us/library/bb726031(SQL.90).aspx)
- <http://msdn.microsoft.com/en-us/sync/cc470041>
- <http://msdn.microsoft.com/en-us/sync/bb887608>
- <http://msdn.microsoft.com/en-us/sync/bb821992>
- <http://msdn.microsoft.com/en-us/sync/bb980926.aspx>
- [http://msdn.microsoft.com/library/dd937565\(SQL.105\).aspx](http://msdn.microsoft.com/library/dd937565(SQL.105).aspx)
- http://en.wikipedia.org/wiki/.NET_Framework
- http://en.wikipedia.org/wiki/.NET_Framework_version_history
- <http://microsoft-net-framework4.software.informer.com/wiki/>
- <http://en.wikipedia.org/wiki/C%2B%2B>
- [http://msdn.microsoft.com/en-us/library/kx37x362\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/kx37x362(v=VS.90).aspx)
- http://en.wikipedia.org/wiki/Microsoft_SQL_Server