Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία:

"Ανάπτυξη Εμπορικής Εφαρμογής σε Περιβάλλον .NET"

Σπουδαστής: Γιαλαμπούκης Κωνσταντίνος *(ΑΜ:664)*

Επιβλέπων Καθηγητής: Αθανάσιος Μαλάμος

Νοέμβριος 2011

Περιεχόμενα

1.Πρόλογος

1.1 Σκοπός – Περιγραφή της πτυχιακής

Σκοπός της πτυχιακής εργασίας ήταν η δημιουργία μέρους μιας εμπορικής εφαρμογής και ειδικότερα τη διαχείριση αποθήκης. Η διαχείριση αποθήκης είναι από τα βασικότερα τμήματα μια εμπορικής διαχείρισης και συναντάται σε όλες τις εφαρμογές που έχει να παρουσιάσει η ελληνική αγορά ανεξαρτήτως εταιρείας ανάπτυξης λογισμικού και γλώσσας προγραμματισμού.

Η εφαρμογή που αναπτύχθηκε δεν μπορεί να συγκριθεί με καμία ήδη υπάρχουσα διαχείριση αποθήκης εμπορικής εφαρμογής γιατί δε δημιουργήθηκε για κάποιο τέτοιο σκοπό. Ο ρόλος που έχει η εφαρμογή είναι περισσότερο βοηθητικός ή και συμπληρωματικός καθώς πολλές φορές καλώς ή κακώς ο τελικός χρήστης για να μπορέσει να παρέμβει απευθείας στο είδος της αποθήκης του θα πρέπει να χρησιμοποιήσει κάποιο συνδυασμό κινήσεων αποθήκης για να τροποποιήσει ένα ή περισσότερα στοιχεία του χωρίς όμως να επηρεάσει κάτι άλλο στις υπόλοιπες ενότητες του προγράμματος, όπως η λογιστική, η κοστολόγηση, η παραγωγή κ.α.

1.2 Τι είναι Εμπορική Εφαρμογή

Από τότε που μπήκε στις επιχειρήσεις η μηχανοργάνωση θεωρείτε αυτονόητο ότι για να έχεις μια σωστή διαχείριση της εταιρείας σου, έστω και αν είναι μια ατομική επιχείρηση που δουλεύεις ο ίδιος και ο τζίρος σου δε ξεπερνάει τις μερικές δεκάδες χιλιάδες ευρώ το χρόνο, θα πρέπει να ρίξεις πάρα πολύ βάρος στην εμπορική εφαρμογή που θα αγοράσεις έτσι ώστε να σου δίνει εκείνα τα εργαλεία για να μπορείς να διαχειριστής την επιχείρηση σου και να έχεις το καλύτερο δυνατό αποτέλεσμα.

Μια τυπική εμπορική εφαρμογή διαθέτει διαχείριση αποθήκης, πελατών, προμηθευτών, πωλήσεων, αγορών και αξιόγραφων. Αυτές είναι οι ελάχιστες ενότητες που πρέπει να διαθέτουν για να έχεις μια πλήρη εικόνα έστω και της πιο μικρής επιχείρησης, από εκεί και πέρα όμως υπάρχουν και άλλες ενότητες, όπως η παραγωγή, η κοστολόγηση, οι πωλητές, ο προϋπολογισμός κ.α. που κυρίως ανταποκρίνονται σε μεγαλύτερες επιχειρήσεις με περισσότερη διαχείριση πληροφορίας. Κάθε ενότητα συνήθως περιέχει την διαχείριση των σταθερών στοιχείων και των οικονομικών στοιχείων αλλά και τις εκτυπώσεις που μπορούμε να πάρουμε με όλη αυτή τη πληροφορία.

2.Εισαγωγή

2.1 Τι είναι ο αντικειμενοστραφής προγραμματισμός

Στην επιστήμη υπολογιστών αντικειμενοστραφή προγραμματισμό (object-oriented programming), ή ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων ονόματι κλάση. Η κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους· αυτή υπήρξε η πρωταρχική καινοτομία του ΑΠ.

Έτσι μπορεί να οριστεί μία προδιαγραφή δομής αποθήκευσης (π.χ. μία κλάση "τηλεόραση") η οποία να περιέχει τόσο ιδιότητες (π.χ. μία μεταβλητή "τρέχον κανάλι") όσο και πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία "άνοιγμα της τηλεόρασης"). Στο εν λόγω παράδειγμα κάθε υλική τηλεόραση (κάθε αντικείμενο αποθηκευμένο πραγματικά στη μνήμη) αναπαρίσταται ως ξεχωριστό, "φυσικό" στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Επομένως μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς "καλούπια". Οι αιτίες που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), όμως τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας

2.2 Βασικές έννοιες αντικειμενοστραφούς προγραμματισμού

Κεντρική ιδέα στον αντικειμενοστραφή προγραμματισμό είναι η κλάση (class), μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού. Πρακτικώς είναι ένας τύπος δεδομένων, ή αλλιώς το προσχέδιο μίας δομής δεδομένων με δικά της περιεχόμενα, τόσο μεταβλητές όσο και διαδικασίες. Τα περιεχόμενα αυτά δηλώνονται είτε ως δημόσια (public) είτε ως ιδιωτικά (private), με τα ιδιωτικά να μην είναι προσπελάσιμα από κώδικα εκτός της κλάσης. Οι διαδικασίες των κλάσεων συνήθως καλούνται μέθοδοι (methods) και οι μεταβλητές τους γνωρίσματα (attributes) ή πεδία (fields). Μία κλάση πρέπει ιδανικά να είναι εννοιολογικά αυτοτελής, να περιέχει δηλαδή μόνο πεδία τα οποία περιγράφουν μία κατηγορία αντικειμένων και δημόσιες μεθόδους οι οποίες επενεργούν σε αυτά όταν καλούνται από το εξωτερικό πρόγραμμα, χωρίς να εξαρτώνται από άλλα δεδομένα ή κώδικα εκτός της κλάσης, και

επαναχρησιμοποιήσιμη, να αποτελεί δηλαδή μαύρο κουτί δυνάμενο να λειτουργήσει χωρίς τροποποιήσεις ως τμήμα διαφορετικών προγραμμάτων.

Αντικείμενο (object) είναι το στιγμιότυπο μίας κλάσης, δηλαδή αυτή καθαυτή η δομή δεδομένων (με αποκλειστικά δεσμευμένο χώρο στη μνήμη) βασισμένη στο «καλούπι» που προσφέρει η κλάση. Παραδείγματος χάρη, σε μία αντικειμενοστρεφή γλώσσα προγραμματισμού θα μπορούσαμε να ορίσουμε κάποια κλάση ονόματι BankAccount, η οποία αναπαριστά έναν τραπεζικό λογαριασμό, και να δηλώσουμε ένα αντικείμενο της με όνομα MyAccount. Το αντικείμενο αυτό θα έχει δεσμεύσει χώρο στη μνήμη με βάση τις μεταβλητές και τις μεθόδους που περιγράψαμε όταν δηλώσαμε την κλάση. Έτσι, στο αντικείμενο θα μπορούσε να περιέχεται ένα γνώρισμα Balance (=υπόλοιπο) και μία μέθοδος GetBalance (=επέστρεψε το υπόλοιπο). Ακολούθως θα μπορούσαμε να δημιουργήσουμε ακόμα ένα ή περισσότερα αντικείμενα της ίδιας κλάσης τα οποία θα είναι διαφορετικές δομές δεδομένων (διαφορετικοί τραπεζικοί λογαριασμοί στο παράδειγμα). Ας σημειωθεί εδώ πως τα αντικείμενα μίας κλάσης μπορούν να προσπελάσουν τα ιδιωτικά περιεχόμενα άλλων αντικειμένων της ίδιας κλάσης.

Ενθυλάκωση δεδομένων (data encapsulation) καλείται η ιδιότητα που προσφέρουν οι κλάσεις να «κρύβουν» τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμμα και να εξασφαλίζουν πως μόνο μέσω των δημόσιων μεθόδων τους θα μπορούν αυτά να προσπελαστούν. Αυτή η τακτική παρουσιάζει μόνο οφέλη καθώς εξαναγκάζει κάθε εξωτερικό πρόγραμμα να φιλτράρει το χειρισμό που επιθυμεί να κάνει στα πεδία μίας κλάσης μέσω των ελέγχων που μπορούν να περιέχονται στις δημόσιες μεθόδους της κλάσης.

Αφαίρεση δεδομένων καλείται η ιδιότητα των κλάσεων να αναπαριστούν αφαιρετικά πολύπλοκες οντότητες στο προγραμματιστικό περιβάλλον. Μία κλάση αποτελεί ένα αφαιρετικό μοντέλο κάποιας κατηγορίας αντικειμένων. Επίσης οι κλάσεις προσφέρουν και αφαίρεση ως προς τον υπολογιστή, εφόσον η καθεμία μπορεί να θεωρηθεί ένας μικρός και αυτάρκης υπολογιστής (με δική του κατάσταση, μεθόδους και μεταβλητές).

Κληρονομικότητα ονομάζεται η ιδιότητα των κλάσεων να επεκτείνονται σε νέες κλάσεις, ρητά δηλωμένες ως κληρονόμους (υποκλάσεις ή 'θυγατρικές κλάσεις'), οι οποίες μπορούν να επαναχρησιμοποιήσουν τις μεταβιβάσιμες μεθόδους και ιδιότητες της γονικής τους κλάσης αλλά και να προσθέσουν δικές τους. Στιγμιότυπα των θυγατρικών κλάσεων μπορούν να χρησιμοποιηθούν όπου απαιτούνται στιγμιότυπα των γονικών (εφόσον η θυγατρική είναι κατά κάποιον τρόπο μία πιο εξειδικευμένη εκδοχή της γονικής), αλλά το αντίστροφο δεν ισχύει. Παράδειγμα κληρονομικότητας είναι μία γονική κλάση Vehicle (=Όχημα) και οι δύο πιο εξειδικευμένες υποκλάσεις της Car (=Αυτοκίνητο) και Bicycle (=Ποδήλατο), οι οποίες λέμε ότι "κληρονομούν" από αυτήν. Πολλαπλή κληρονομικότητα είναι η δυνατότητα που προσφέρουν ορισμένες γλώσσες προγραμματισμού μία κλάση να κληρονομεί ταυτόχρονα από περισσότερες από μία γονικές. Από μία υποκλάση μπορούν να προκύψουν νέες υποκλάσεις που κληρονομούν από αυτήν, με αποτέλεσμα μία ιεραρχία κλάσεων που συνδέονται μεταξύ τους "ανά γενιά" με σχέσεις κληρονομικότητας.

Υπερφόρτωση μεθόδου (method overloading) είναι η κατάσταση κατά την οποία υπάρχουν, στην ίδια ή σε διαφορετικές κλάσεις, μέθοδοι με το ίδιο όνομα και

πιθανώς διαφορετικά ορίσματα. Αν πρόκειται για μεθόδους της ίδιας κλάσης διαφοροποιούνται μόνο από τις διαφορές τους στα ορίσματα και στον τύπο επιστροφής.

Υποσκέλιση μεθόδου (method overriding) είναι η κατάσταση κατά την οποία μία θυγατρική κλάση και η γονική της έχουν μία μέθοδο ομώνυμη και με τα ίδια ορίσματα. Χάρη στη δυνατότητα του πολυμορφισμού ο μεταγλωττιστής «ξέρει» πότε να καλέσει ποια μέθοδο, βασισμένος στον τύπο του τρέχοντος αντικειμένου. Δηλαδή πολυμορφισμός είναι η δυνατότητα των αντικειμενοστρεφών μεταγλωττιστών να αποφασίζουν δυναμικά ποια είναι η κατάλληλη να κληθεί μέθοδος σε συνθήκες υποσκέλισης.

Αφηρημένη κλάση (abstract class) είναι μία κλάση που ορίζεται μόνο για να κληρονομηθεί σε θυγατρικές υποκλάσεις και δεν υπάρχουν δικά της στιγμιότυπα (αντικείμενα). Η αφηρημένη κλάση ορίζει απλώς ένα "συμβόλαιο" το οποίο θα πρέπει να ακολουθούν οι υποκλάσεις της όσον αφορά τις υπογραφές των μεθόδων τους (όπου ως υπογραφή ορίζεται το όνομα, τα ορίσματα και η τιμή επιστροφής μίας διαδικασίας). Μία αφηρημένη κλάση μπορεί να έχει και μη αφηρημένες μεθόδους οι οποίες υλοποιούνται στην ίδια την κλάση (αν και φυσικά μπορούν να υποσκελίζονται σε υποκλάσεις). Αντιθέτως οι αφηρημένες μέθοδοί της είναι απλώς ένας ορισμός της υπογραφής τους και εναπόκειται στις υποκλάσεις να τις υλοποιήσουν. Μία αφηρημένη κλάση που δεν έχει γνωρίσματα και όλες οι μέθοδοί της είναι αφηρημένες και δημόσιες καλείται διασύνδεση (interface). Οι κλάσεις που κληρονομούν από μία διασύνδεση λέγεται ότι την "υλοποιούν".

Παρακάτω παραθέτω ένα παράδειγμα στο οποίο φαίνεται η σύνταξη και δημιουργία αντικειμένου καθώς επίσης και το κάλεσμα των μεθόδων της κλάσης.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Mathematix

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine(“Pi is ” + MathTest.GetPi());
```

```
int x = MathTest.GetSquareOf(5);
Console.WriteLine(“Square of 5 is: ” + x);
MathTest math = new MathTest();
math.value = 30;
Console.WriteLine(“Value field of match variable contains: ” + math.value);
Console.WriteLine(“Square of 30 is: ” + math.GetSquare());
Console.ReadLine();
}
class MathTest
{
public int value;
public int GetSquare()
{
return value * value;
}
public static int GetSquareOf(int x)
{
return x * x;
}
public static double GetPi()
{
return 3.14159;
}
}
}
```

}

Αν το τρέξουμε το κομμάτι του κώδικα αυτού είναι το παρακάτω:

Καθώς μπορούμε να δούμε στον κώδικα, η κλάση MathTest περιέχει ένα πεδίο το οποίο είναι αριθμός και υπάρχει μία μέθοδος που βρίσκει το τετράγωνο της μεθόδου αυτής. Επίσης περιέχει δύο static μεθόδους οι οποίες επιστρέφουν η μία το Pi και η άλλη το τετράγωνο του ορίσματος μέσα στης παρενθέσεις.

2.3.Τι είναι το .NET Framework

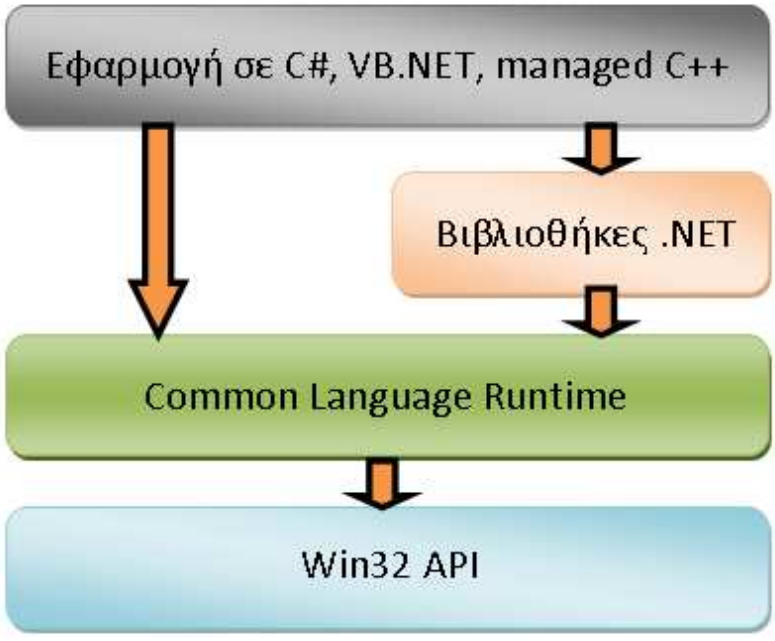
Το .NET Framework είναι ένα λογισμικό που εκτελείται κυρίως σε Microsoft Windows. Περιλαμβάνει μια μεγάλη βιβλιοθήκη και υποστηρίζει πολλές γλώσσες προγραμματισμού που επιτρέπει την «διαλειτουργικότητα» (κάθε γλώσσα μπορεί να χρησιμοποιήσει κώδικα που γράφεται σε άλλες γλώσσες). Προγράμματα που έχουν γραφτεί για το.NET Framework εκτελούνται σε ένα περιβάλλον λογισμικού (σε αντιδιαστολή με περιβάλλον υλικού), γνωστό ως Common Laguage Runtime (CLR), μια εφαρμογή εικονικής μηχανής που παρέχει σημαντικές υπηρεσίες, όπως η ασφάλεια, διαχείριση μνήμης και χειρισμό εξαίρεσης. Η βιβλιοθήκη και το CLR μαζί συνιστούν το .NET Framework.

Η .NET Framework βιβλιοθήκη παρέχει περιβάλλον εργασίας χρήστη, πρόσβαση στα δεδομένα, συνδεσιμότητα βάσεων δεδομένων, κρυπτογράφηση, ανάπτυξη εφαρμογών web , αριθμητικών αλγορίθμων και επικοινωνίας μέσω δικτύου. Προγραμματιστές παράγουν λογισμικό, συνδυάζοντας τον δικό τους πηγαίο κώδικα με το .NET Framework και άλλες βιβλιοθήκες. Η.NET Framework προορίζεται να χρησιμοποιείται από τις περισσότερες νέες εφαρμογές που έχουν δημιουργηθεί στην πλατφόρμα των Windows.

Οι γλώσσες προγραμματισμού συνήθως αποτελούνται από έναν compiler και ένα runtime περιβάλλον. Ο compiler μεταφράζει τον κώδικα σε εκτελέσιμο αρχείο που μπορεί να εκτελεστεί από τους χρήστες. Το runtime περιβάλλον παρέχει ένα σύνολο υπηρεσιών του λειτουργικού συστήματος, στον εκτελέσιμο κώδικα. Οι υπηρεσίες αυτές είναι ενσωματωμένες σε ένα επίπεδο runtime (Runtime Layer) που επιτρέπει στον κώδικα να μην ασχολείται με λεπτομέρειες χαμηλού επιπέδου του λειτουργικού συστήματος. Τέτοιες λειτουργίες μπορεί να είναι η διαχείριση μνήμης, εγγραφή και ανάγνωση αρχείων κλπ. Πριν το .NET Framework, κάθε γλώσσα είχε και το δικό της runtime περιβάλλον. Η Visual Basic ερχόταν με το MSVBVM60.DLL, ενώ η Visual C++ με το MSVCRT.DLL. Το περιβάλλον ενσωματωνόταν με τον εκτελέσιμο κώδικα και έπρεπε να εγκατασταθεί στο μηχάνημα του χρήστη.Το βασικό πρόβλημα με τα περιβάλλοντα αυτά, βρίσκεται στο ότι ήταν σχεδιασμένα για χρήση με μόνο μία γλώσσα. Δεν μπορούσαν να χρησιμοποιηθούν λειτουργίες από το περιβάλλον μιας γλώσσας, σε μία άλλη. Έτσι,

ένας από τους βασικούς στόχους του .NET Framework ήταν να ενοποιήσει τα runtime περιβάλλοντα ώστε οι προγραμματιστές να μπορούν να χρησιμοποιούν μόνο ένα περιβάλλον. Έτσι η λύση που δόθηκε ήταν η Common Language Runtime (CLR). ToCLR παρέχει δυνατότητες όπως διαχείριση μνήμης, ασφάλεια, διαχείριση λαθών κλπ, και όλα αυτά για κάθε γλώσσα που δουλεύει με το .NET Framework.
Το CLR επίσης επιτρέπει στις γλώσσες να συνεργάζονται μεταξύ τους. Μπορεί για παράδειγμα να δεσμευτεί ένα κομμάτι μνήμης με κώδικα γραμμένο στην Visual Basic .NET και το ίδιο κομμάτι να ελευθερωθεί με κώδικα γραμμένο σε άλλη γλώσσα όπως η C#.

Οι προγραμματιστές αρέσκονται στο να δουλεύουν με κώδικα που ήδη έχει δοκιμαστεί και φαίνεται να λειτουργεί, όπως για παράδειγμα το Win32 API και οι βιβλιοθήκες MFC. Η επαναχρησιμοποίηση κώδικα ήταν στόχος της προγραμματιστικής κοινότητας, από πολύ παλιά. Πολλές γλώσσες είχαν πρόσβαση σε κομμάτια κώδικα δοκιμασμένα, έτοιμα για εκτέλεση. Οι προγραμματιστές που χρησιμοποιούσαν την Visual C++ είχαν επωφεληθεί από βιβλιοθήκες όπως η Microsoft Foundation Classes (MFC) που τους επέτρεπαν να δημιουργήσουν εφαρμογές Windows εύκολα και γρήγορα. Ωστόσο, το ότι οι βιβλιοθήκες αυτές ήταν προορισμένες για μία μόνο γλώσσα σήμαινε ότι δεν μπορούσαν να χρησιμοποιηθούν με καμία άλλη γλώσσα.



Το .NET Framework παρέχει πολλές κλάσεις για να βοηθήσει τους προγραμματιστές στην επαναχρησιμοποίηση κώδικα. Οι βιβλιοθήκες .NET Class Libraries περιέχουν κώδικα για προγραμματιστικά θέματα όπως νήματα, εγγραφή/ανάγνωση αρχείων, υποστήριξη βάσεων δεδομένων, μετατροπή σε XML, δομές δεδομένων όπως στοίβες

και ουρές κλπ. Το καλύτερο σημείο βέβαια είναι το ότι η βιβλιοθήκη είναι διαθέσιμη σε κάθε γλώσσα που λειτουργεί με το .NET Framework.

Το .NET Framework παρέχει ένα σύνολο εργαλείων για να βοηθήσει στην κατασκευή κώδικα που λειτουργεί με αυτό. Η Microsoft παρέχει ένα σύνολο γλωσσών που είναι ήδη συμβατές με το .NET. Η C# είναι μία από αυτές. Επίσης δημιουργήθηκαν νέες εκδόσεις της Visual Basic και της Visual C++ όπως και μία νέα έκδοση της Jscript.NET. Ένα πολύ σημαντικό στοιχείο είναι ότι οι συμβατές γλώσσες με το .NET δεν είναι αποκλειστικά της Microsoft, αφού η εταιρία έχει δημοσιεύσει πλήρης τεκμηρίωση που δείχνει το πώς οι κατασκευαστές γλωσσών μπορούν να κάνουν τις γλώσσες τους συμβατές με το .NET, και διάφοροι κατασκευαστές το επιχείρησαν όπως η COBOL και η Perl. Υπάρχουν αυτή τη στιγμή πάνω από 20 γλώσσες τρίτων κατασκευαστών που μπορούν και λειτουργούν στο περιβάλλον .NET Framework.

Πλεονεκτήματα .NET

Το .NET έχει πολλά πλεονεκτήματα για την ανάπτυξη εφαρμογών:

- Είναι εγγενώς αντικειμενοστραφής πλατφόρμα.
- 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.
- Η χρήση βιβλιοθηκών (assemblies) κάνει πολύ εύκολη την επαναχρησιμοποίηση κώδικα.
- 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.
- Παρέχει πληθώρα έτοιμων λειτουργιών που κάνουν την ανάπτυξη κώδικα πολύ εύκολη.
- Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

Μειονεκτήματα .NET (για ανάπτυξη παιχνιδιών)

Το .NET έχει 2 μειονεκτήματα που αφορούν ειδικά την ανάπτυξη βιντεοπαιχνιδιών και όχι την γενική ανάπτυξη εφαρμογών:

- Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.
- Το CLR εισάγει μια (μικρή ίσως) καθυστέρηση στην εκτέλεση της εφαρμογής.

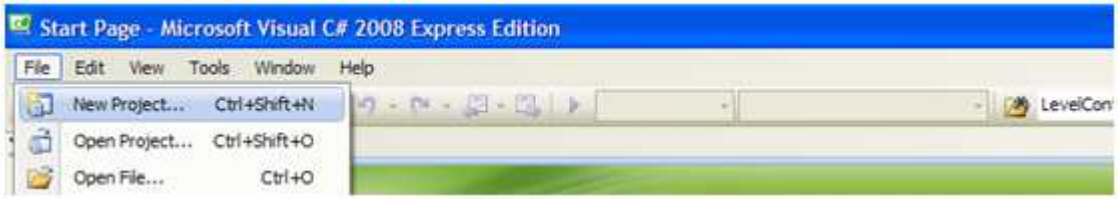
Το πρώτο, αν και είναι πολύ χρήσιμο χαρακτηριστικό για γενικό προγραμματισμό, σε ένα βιντεοπαιχνίδι μπορεί να δημιουργήσει πρόβλημα. Η απελευθέρωση μνήμης που

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.

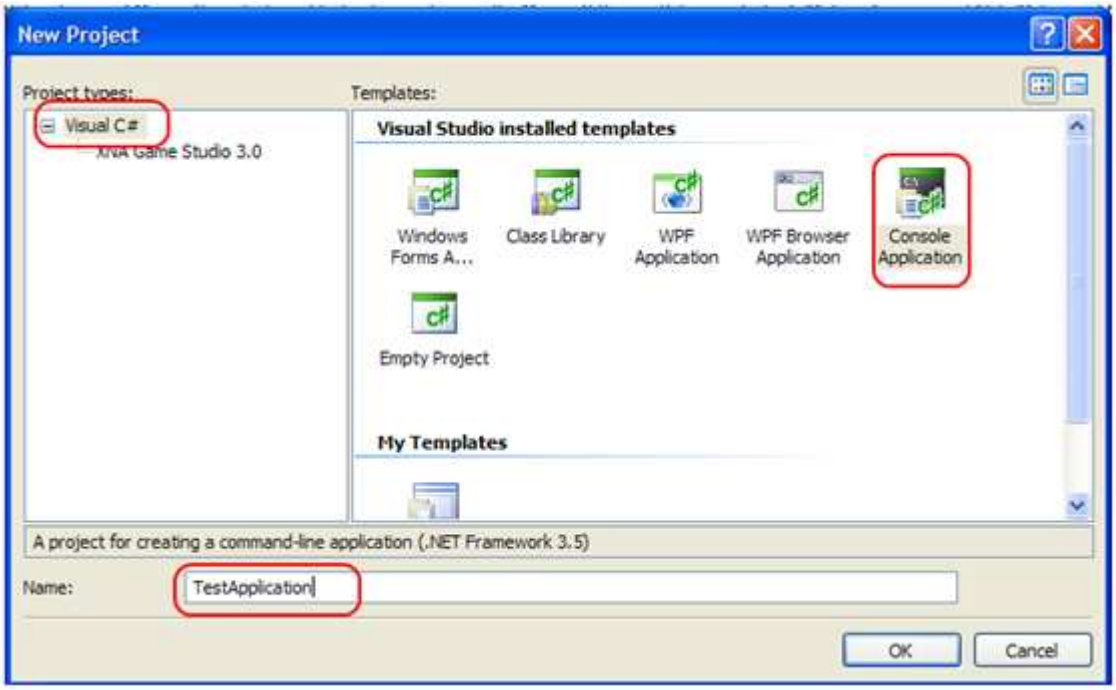
2.4 Τι είναι η C#

Η C# είναι μια σχετικά νέα αντικειμενοστραφής γλώσσα προγραμματισμού η οποία δημιουργήθηκε από την Microsoft. Δανείζεται πολλά στοιχεία, και έχει παρόμοια σύνταξη, με την C++ και την Java, κάνοντας την εκμάθηση της σχετικά εύκολη. Είναι γλώσσα ειδικά σχεδιασμένη για να υποστηρίζει το .NET framework της ίδιας εταιρείας. Βασικό χαρακτηριστικό της είναι ότι δεν παράγει απευθείας κώδικα μηχανής όπως η C++, άλλα ένα ενδιάμεσο κώδικα που στοχεύει το .NET.

Για να δούμε πρακτικά πως είναι ο κώδικας μιας C# εφαρμογής τρέχουμε το **Visual C# 2008 Express Edition**, και δημιουργούμε ένα νέο project.

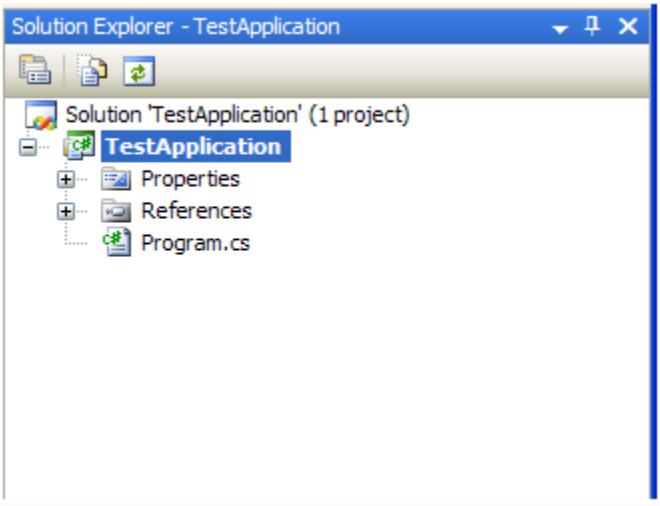


Στην συνέχεια επιλέγουμε σαν Project Type το **Visual C#**, στην συνέχεια ως template το **Console Application** και ονομάζουμε την εφαρμογή μας **TestApplication**, και πατάμε το OK.



Ουσιαστικά με τον τρόπο αυτό καθοδηγήσαμε το Visual Studio να φτιάξει το σκελετό μιας C# εφαρμογής που θα τρέχει σε DOS παράθυρο (console), με απλό κείμενο χωρίς γραφικά.

Το Visual Studio θα δημιουργήσει ένα Solution με το όνομα TestApplication και ένα Project με το ίδιο όνομα.



Κάνοντας διπλό κλικ στο αρχείο Program.c εμφανίζεται ο σκελετός του προγράμματος:

```
01 using System;
02 using System.Collections.Generic;
03 using System.Linq;
```

```
04 using System.Text;
05
06 namespace TestApplication
07 {
08     class Program
09     {
10         static void Main(string[] args)
11         {
12         }
13     }
14 }
```

Πολλά από τα χαρακτηριστικά ενός C# προγράμματος εμφανίζονται ήδη. Οι δηλώσεις **using** στην αρχή του προγράμματος καθορίζουν ποιες βιβλιοθήκες θα χρησιμοποιήσουμε στην εφαρμογή. Το **namespace** ορίζει ένα «χώρο» μέσα στο οποίο θα δημιουργούνται οι μεταβλητές στο πρόγραμμα. Είναι για καθαρά ονοματολογικούς σκοπούς και επιτρέπει το διαχωρισμό 2 μεταβλητών με το ίδιο όνομα που κάθε μια ανήκει σε διαφορετικές βιβλιοθήκες. Μπορεί να το δει κανείς σαν το όνομα ενός καταλόγου (folder) στο δίσκο. Δύο αρχεία με το ίδιο όνομα μπορούν να υπάρχουν σε δυο διαφορετικούς καταλόγους. Για να κάνω αναφορά σε κάποιο από τα δύο χρησιμοποιώ και το όνομα του καταλόγου. Το ίδιο ακριβώς συμβαίνει και με το namespace.

Το βασικό πρόγραμμα είναι μια κλάση (**class**), με το όνομα **Program**. Στην C# όλος ο κώδικας της εφαρμογής πρέπει να είναι μέρος μιας κλάσης, εν αντιθέσει με την C++ που επιτρέπει δημιουργία κώδικα και εκτός κλάσης.

Τέλος η συνάρτηση-μέλος της κλάσης Program, η **Main**, είναι το σημείο εισόδου της εφαρμογής, ανάλογα με την main() της C++. Τα ονόματα της κλάσης του προγράμματος (Program) και της συνάρτησης εισόδου (Main) είναι προκαθορισμένα και δεν μπορούν να αλλάξουν.

Πατώντας το πλήκτρο F5, η εφαρμογή μεταγλωττίζεται και τρέχει. Πολύ γρήγορα θα δούμε ένα παράθυρο κονσόλας να ανοίγει και να κλείνει. Αυτό γίνεται γιατί ουσιαστικά το πρόγραμμα δεν εκτελεί καμία ενέργεια (η Main είναι άδεια).

Αλλάζουμε το κώδικα της Main λίγο ώστε να βγάλει κάποια έξοδο στο παράθυρο της κονσόλας, το παραδοσιακό “Hello World!”:

```
01 using System;
02 using System.Collections.Generic;
03 using System.Linq;
```

```
04 using System.Text;
05
06 namespace TestApplication
07 {
08     class Program
09     {
10         static void Main(string[] args)
11         {
12             Console.WriteLine("Hello World!");
13             Console.ReadLine();
14             return;
15         }
16     }
17 }
```

Εδώ κάνουμε χρήση μιας βιβλιοθήκης του .NET, της **System**, χρησιμοποιώντας το αντικείμενο της **Console**. Το αντικείμενο αυτό διαχειρίζεται είσοδο-έξοδο στο παράθυρο της κονσόλας. Υποστηρίζονται πολλά τέτοια αντικείμενα στο .NET, που υλοποιούν ένα πλήθος λειτουργιών. Χρησιμοποιούμε τη συνάρτηση **WriteLine**() για να απεικονίσουμε μια γραμμή κειμένου και την **ReadLine**() για να διαβάσουμε μια γραμμή κειμένου – στην περίπτωση μας το πλήκτρο ENTER. Στο τέλος της συνάρτησης, η εντολή **return** σηματοδοτεί την λήξη της. Στην συγκεκριμένη περίπτωση είναι περιτή η χρήση της μιας και η Main δεν επιστρέφει τίποτα.

Παρατηρούμε επίσης ότι κάθε εντολή στην C#, όπως και στην C++ πρέπει να τερματίζεται με το ελληνικό ερωτηματικό.

Πατώντας πάλι το F5 ανοίγει το παράθυρο, και απεικονίζεται η γραμμή κειμένου «Hello World!». Για να κλείσει το παράθυρο πρέπει να πατήσουμε Enter.

Ενημερωτικά, ο Intermediate Language κώδικας που παράγεται για την συνάρτηση Main είναι ο εξής:

```
01 .method private hidebysig static void Main(string[] args) cil
   managed
02 {
03   .entrypoint
04   // Code size       19 (0x13)
05   .maxstack  8
06   IL_0000:  nop
07   IL_0001:  ldstr      "Hello World!"
08   IL_0006:
   call       void [mscorlib]System.Console::WriteLine(string)
09   IL_000b:  nop
```

```
10   IL_000c:  call       string [mscorlib]System.Console::ReadLine()
11   IL_0011:  pop
12   IL_0012:  ret
13 } // end of method Program::Main
```

Είναι σχετικά εύκολο να καταλάβει κανείς την λειτουργία του, συγκρίνοντας τον με τον αρχικό κώδικα της συνάρτησης Main (C#). Είναι για παράδειγμα εμφανές ότι φορτώνει το κείμενο «Hello World!» σε μια προσωρινή μεταβλητή (εντολή IL_001) και στην συνέχεια καλεί την συνάρτηση WriteLine για να την τυπώσει την οθόνη. Επίσης γίνεται φανερό ότι η βιβλιοθήκη System με το αντικείμενο της Console ανήκουν στο .NET και δεν αποτελούν μέρος της C#.

Σε γενικές γραμμές η C#, όσον αφορά τις μεταβλητές και τους τύπους δεδομένων της, δεν διαφέρει δραματικά από γλώσσες όπως η Java και η C++. Κάποιος με εμπειρία σε άλλες (σχετικά σύγχρονες) γλώσσες θα βρεθεί σε ένα οικείο περιβάλλον κατά την ανάπτυξη ενός προγράμματος σε C#. Ξεκινάμε με μια γρήγορη ανασκόπηση των τύπων δεδομένων που υποστηρίζει η C#.

Μεταβλητές

Η C# υποστηρίζει τους παραδοσιακούς τύπους δεδομένων όπως **int** για ακέραιους αριθμούς, **float** για αριθμούς κινητής υποδιαστολής, **string** για κείμενο, **char** για χαρακτήρες. Επιπλέον υποστηρίζει του τύπους **class** και **struct** οι οποίοι επιτρέπουν στο χρήστη να ορίσει δικά του αντικείμενα.

Ο παρακάτω πίνακας περιλαμβάνει τους πιο κοινούς τύπους δεδομένων που χρησιμοποιούμε κατά το προγραμματισμό με C#

Τύπος	**Δεδομένα**	**Παράδειγμα χρήσης**
bool	true ή false	bool isRaining = false;
byte	Ακέραιος αριθμός από 0 μέχρι 255	byte counter=4;
short	Ακέραιος αριθμός από –32,768 μέχρι 32,767	short myNumber = -3442;
int	Ακέραιος αριθμός 32 bit	int largeNumber = 1234223;
double	Αριθμός κινητής υποδιαστολής μεγάλης ακρίβειας	double stockValue = 2342.23332;
float	Αριθμός κινητής υποδιαστολής	float exchangeRate = 233.23f;
char	Χαρακτήρας ASCII 8 bit	char myChar = ‘a’;
string	Κείμενο	string username = “Kostas”;

Μερικές παρατηρήσεις πάνω στους τύπους δεδομένων: Οι τύποι short και int έχουν παρόμοια χρήση, μόνο που ο int έχει μεγαλύτερο εύρος, δηλαδή μπορεί να

αποθηκεύσει μεγαλύτερο εύρος (περισσότερους) αριθμούς από ότι ο short. Αυτό συμβαίνει γιατί μια μεταβλητής τύπου **short** χρησιμοποιεί 16 bit στην μνήμη ενώ μια **int** 32 bit. Το ίδιο ισχύει για τους τύπους **float** (32bit) και **double** (64bit). Ο χαρακτήρας f στο τέλος του αριθμού στο παράδειγμα για το τύπο float σηματοδοτεί ότι ο αριθμός αυτός είναι πράγματι τύπου float (και συνεπώς απαιτεί λιγότερο χώρο για να αποθηκευτεί). Τέλος ένας χαρακτήρας (char) ορίζεται με μονά εισαγωγικά και ένα κείμενο (string) με διπλά.

Class και αντικείμενα

Πέρα από αυτούς τους βασικούς τύπους δεδομένων, η C# υποστηρίζει και το τύπο **class**. Η **class** μας επιτρέπει να ορίσουμε δικά μας αντικείμενα. Εν συντομία για όσους δεν γνωρίζουν τι είναι, ένα αντικείμενο σε μια αντικειμενοστραφή γλώσσα προγραμματισμού (όπως η C#, C++, Java κλπ), είναι μια οντότητα που χαρακτηρίζεται από δεδομένα και συμπεριφορά. Τα αντικείμενα αυτά έχουν άμεση αντιστοίχιση με αντικείμενα του πραγματικού κόσμου. Για παράδειγμα ένα αυτοκίνητο σε ένα παιχνίδι αγώνων ράλι χαρακτηρίζεται από το χρώμα του, το πόσες πόρτες έχει, την ταχύτητα του, την κατεύθυνση του, αν έχει ταχύτητες ή είναι αυτόματο, το ποσοστό ζημίας από τρακαρίσματα που έχει υποστεί. Αυτά είναι τα δεδομένα που περιγράφουν το αντικείμενο του αυτοκινήτου. Επιπλέον μπορούμε να ορίσουμε και κάποιες «λειτουργίες» που επιδρούν και διαχειρίζονται τα δεδομένα αυτά. Μια λειτουργία θα μπορούσε να είναι η «αλλαγή ταχύτητας». Μια άλλη «πάτημα γκαζιού». Οι λειτουργίες αυτές αλλάζουν τις τιμές των δεδομένων του αντικειμένου.

Στην C# το αντικείμενο του αυτοκινήτου μεταφέρεται ως

```
01 class Car
02 {
03     private float speed;
04     private int gear;
05 
06     public Car()
07     {
08         speed = 0;
09         gear = 0;
10     }
11 
12     public void IncreaseSpeed(float amount)
13     {
14         speed = speed + amount;
```

```
15     }
16
17     public void DecreaseSpeed(float amount)
18     {
19         speed = speed - amount;
20     }
21
22     public void ChangeGear(int newGear)
23     {
24         gear = newGear;
25     }
26 }
```

Με το τρόπο αυτό ορίζουμε ένα νέο τύπο δεδομένων, το αντικείμενο Car το οποίο περιγράφει με απλοϊκό τρόπο μια οντότητα αυτοκινήτου και διάφορες λειτουργίες πάνω σε αυτό (με επίσης απλοϊκό τρόπο). Το αντικείμενο περιλαμβάνει 2 μεταβλητές speed και gear περιέχουν την τρέχουσα ταχύτητα και ταχύτητα (δεν είναι φοβερά τα νέο-ελληνικά όσον αφορά τους τεχνολογικούς όρους;) του αυτοκινήτου, συναρτήσεις (μεθόδους) για αύξηση και μείωση της ταχύτητας και μια ειδική μέθοδο (Car()) τη οποίας την ύπαρξη θα εξηγήσω σε λίγο.

Για να ορίσουμε ένα αυτοκίνητο με βάση το πρότυπο αυτό το δηλώνουμε ως εξής:

```
1 Car audiQuattro;
```

Η παραπάνω δήλωση δημιουργεί μια μεταβλητή τύπου Car, αλλά δεν δεσμεύει μνήμη για το αντικείμενο αυτό καθ’αυτό. Αυτό πρέπει να γίνει ως εξής:

```
1 audiQuattro = new Car();
```

Ο τελεστής new δεσμεύει όση ακριβώς μνήμη χρειάζεται για να αποθηκευτεί ένα αντικείμενο τύπου Car. Το audiQuattro είναι τώρα ένα νέο αντικείμενο τύπου Car. Για να αλλάξω τις ιδιότητες (μεταβλητές) του νέου αυτοκινήτου καλώ απλά τις μεθόδους του:

```
1 audiQuattro.ChangeGear(1);
2 audiQuattro.IncreaseSpeed(20);
```

Παρατηρούμε ότι οι μεταβλητές δηλώνονται ως private και οι μέθοδοι ως public. Σε ένα αντικείμενο, οτιδήποτε δηλώνεται ως private δεν είναι ορατό και προσπελάσιμο εκτός του αντικειμένου. Αντίθετα, οτιδήποτε δηλώνουμε ως public είναι. Για παράδειγμα η παρακάτω χρήση δεν είναι επιτρεπτή:

```
1 audiQuattro.speed = 15.0f;
```

Αυτό συμβαίνει γιατί η speed έχει δηλωθεί ως private. Ο μεταγλωττιστής της C# δεν θα επιτρέψει την ολοκλήρωση της δημιουργίας του εκτελέσιμου κώδικα και θα επισημάνει το λάθος. Ο χαρακτηρισμός private/public δεν ανήκει αποκλειστικά σε μεταβλητές/μέθοδους αλλά σε οτιδήποτε περιέχει ένα αντικείμενο. Θα μπορούσε για παράδειγμα η μεταβλητή speed να είναι public.

Τέλος θέλω να αναφερθώ στη μέθοδο **Car()**. Η μέθοδος αυτή, που έχει πάντα το ίδιο όνομα με το **class** που τη περιέχει ονομάζεται **constructor** και καλείται ***πάντα*** όταν δεσμεύουμε μνήμη για ένα αντικείμενο του αντίστοιχου τύπου με τη χρήση του τελεστή **new**. Σκοπός της είναι η αρχικοποίηση του αντικειμένου δίνοντας για παράδειγμα κάποια αρχική τιμή στις μεταβλητές του. Στο συγκεκριμένο παράδειγμα τη χρησιμοποιούμε για να θέσουμε αρχικές τιμές 0 στις μεταβλητές speed και gear (κάτι που δεν είναι απολύτως απαραίτητο, καθώς γίνεται αυτόματα).

Εκτός της **class**, η C# υποστηρίζει τη δημιουργία νέων τύπων δεδομένων με τη χρήση της **struct**. Η **struct** είναι παρόμοια σε χρήση με τη **class**, διαφέρει όμως σε μερικά σημεία τα οποία θα αναλύσουμε σε επόμενο άρθρο. Ανακεφαλαιώνοντας, η C# υποστηρίζει βασικούς τύπους δεδομένων (primitive types), όπως ο **int**, **short**, **char** κλπ, και πιο σύνθετους όπως η **class** (reference types). Μια βασική διαφορά τους είναι ότι ένα βασικό τύπο μπορώ να τον ορίσω απευθείας:

```
1 int value = 22;
```

ενώ ένα πιο σύνθετο τύπου (class) πρέπει να χρησιμοποιήσω το τελεστή new για να δεσμεύσω μνήμη για αυτό:

```
1 Car fiat = new Car();
```

Αυτή η διαφορά έχει μεγάλη επίπτωση στη χρήση τους και σε τι είδους μνήμη αποθηκεύονται, κάτι που έχει με την σειρά του επίπτωση στην ανάπτυξη βιντεοπαιχνιδιών με το XNA Game Studio.

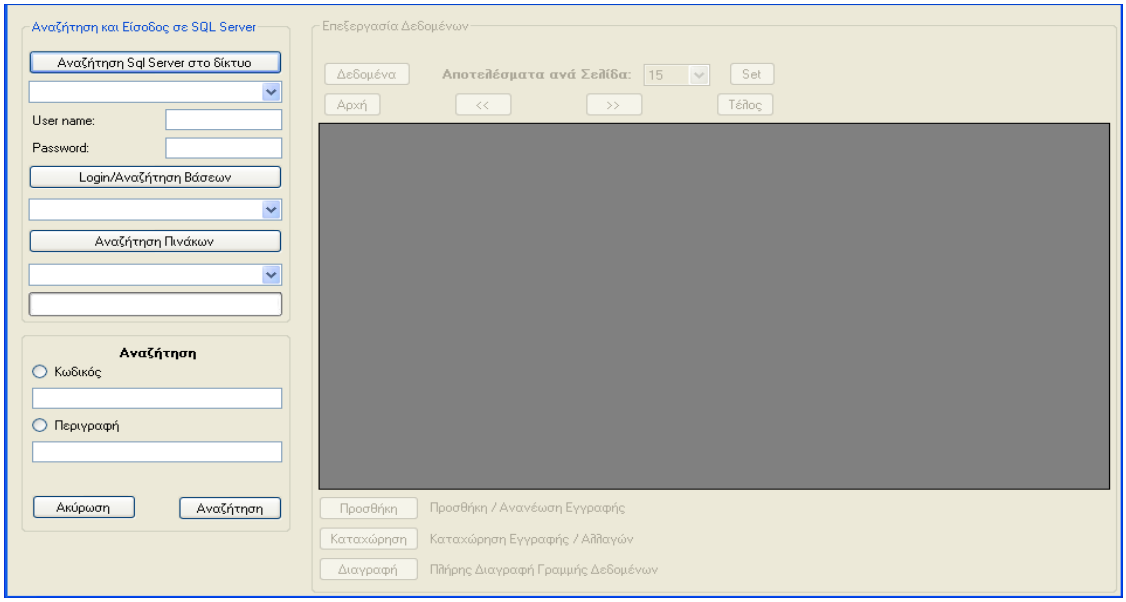
3.Ανάπτυξη της εφαρμογής

3.1.Γενικά

Η ιδέα για την υλοποίηση αυτής της εφαρμογής προήλθε εξ' ολοκλήρου από κάποιους τελικούς χρήστες του προγράμματος της ALTEC, Atlantis E.R.P. όπου είχαν σοβαρή δυσκολία να τροποποιήσουν τα ποσοτικά υπόλοιπα κάποιων ειδών τους στην αποθήκη. Η δυσκολία υπάρχει γιατί κανένα εμπορικό πρόγραμμα δε σε "αφήνει" απ' ευθείας να τροποποιήσεις αυτό το αποτέλεσμα, θα πρέπει δηλαδή να δικαιολογηθεί με κάποια κίνηση αποθήκης, όπως κίνηση απογραφής, εσωτερική διακίνηση από αποθηκευτικό χώρο σε αποθηκευτικό χώρο κ.α. Δε θα αναφερθώ στους λόγους όπου μπορεί ένας χρήστης να έχει έρθει σε τέτοια κατάσταση, ούτε στην επεξήγηση των κινήσεων που λύνουν αυτό το πρόβλημα αλλά πρέπει να δικαιολογηθούν λογιστικά ή εμπορικά καθώς δεν αφορά τον αναγνώστη.

3.2.Ανάλυση εφαρμογής

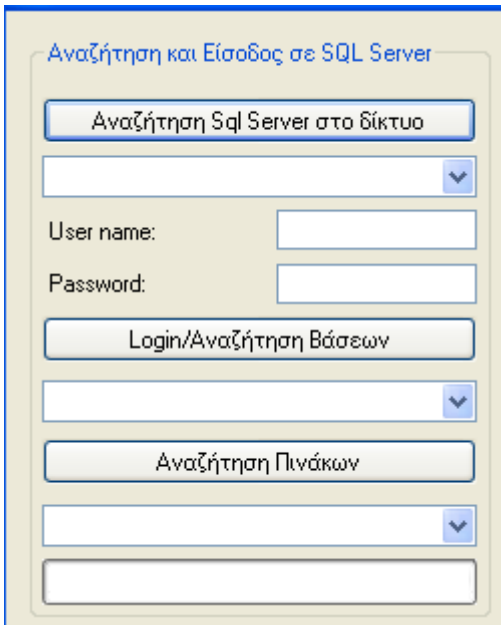
Το πάνελ της εφαρμογής αποτελείται από τρείς κεντρικές φόρμες. Οι δύο φόρμες αποτελούν τον διάλογο της εφαρμογής όπου με πολύ απλά βήματα καθοδηγούν τον χρήστη για να συνδεθεί με τη βάση δεδομένων και τον πίνακα που τον ενδιαφέρει, αλλά και να κάνει αναζήτηση στον πίνακα των εγγραφών που χρειάζεται. Η τρίτη φόρμα φορτώνει τον πίνακα και με τρία κουμπιά διαχείρισης μπορείς να διαμορφώσεις τις εγγραφές σου. Παρακάτω θα αναλύσουμε τη κάθε φόρμα ξεχωριστά αλλά και τον κώδικα.



Εικόνα 1 Κεντρικό Πάνελ Εφαρμογής

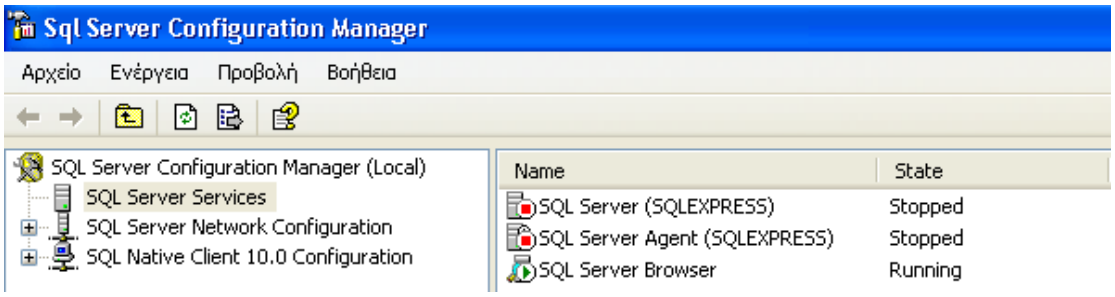
3.3.Ανάλυση πρώτης φόρμας διαλόγου

Στη πρώτη φόρμα διαλόγου, ο χρήστης με το κουμπί *Αναζήτηση SQL Server στο δίκτυο* αναζητά εγκατεστημένους τοπικά SQL Servers αλλά και SQL Servers που είναι εγκατεστημένοι σε άλλους υπολογιστές, όπως ο Server που είναι πολύ πιθανό να υπάρχουν και εκεί εγκατεστημένοι. Σε λίγα δευτερόλεπτα ο χρήστης έχει μπροστά του μια λίστα με όλους τους διαθέσιμους servers. Στα δύο πεδία που υπάρχουν κάτω από τη λίστα ο χρήστης καλείται να βάλει το username και το password της σύνδεσης με τον SQL Server που έχει επιλέξει. Αυτά τα δύο πεδία ορίζονται κατά την εγκατάσταση του SQL Server Management Studio και αρμόδιος να τα γνωρίζει είναι είτε ο διαχειριστής του δικτύου, αν αναφερόμαστε σε μία εταιρεία, είτε ο ίδιος ο χρήστης αν έχουμε μια τοπική εγκατάσταση που την έχει κάνει ο ίδιος.



Εικόνα 2 Φόρμα διαλόγου σύνδεσης σε βάση δεδομένων

Παρακάτω αναλύεται ο κώδικας που χρησιμοποιήθηκε για την ανεύρεση ενεργών συνδέσεων SQL Server στο δίκτυο και την σύνδεση με αυτόν. Απαραίτητη προϋπόθεση για να επιστραφούν σωστά αποτελέσματα είναι η υπηρεσία του SQL Server από το SQL Server Configuration Manager να είναι ενεργοποιημένη όπως φαίνεται στην εικόνα πιο κάτω.



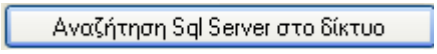
Εικόνα 3 Sql Server Configuration Manager

```
public class SQLInfoEnumerator
      {
            /// <summary>
            /// A string to hold the selected SQL Server
            /// </summary>
            string m_SQLServer;
            /// <summary>
            /// A string to hold the username
            /// </summary>
            string m_Username;
            /// <summary>
            /// A string to hold the password
            /// </summary>
            string m_Password;
            /// <summary>
            /// Property to set the SQL Server instance
            /// </summary>
            public string SQLServer
            {
                  set{m_SQLServer=value;}
            }
            /// <summary>
            /// Property to set the Username
            /// </summary>
            public string Username
            {
                  set{m_Username=value;}
            }
            /// <summary>
            /// Property to set the Password
            /// </summary>
            public string Password
            {
                  set{m_Password=value;}
            }

            /// <summary>
            /// Enumerate the SQL Servers returning a list (if any
exist)
            /// </summary>
            /// <returns></returns>
            public string[] EnumerateSQLServers()
            {
                  return RetrieveInformation(SQL_DRIVER_STR);
            }
            /// <summary>
            /// Enumerate the specified SQL server returning a list
of databases (if any exist)
            /// </summary>
            /// <returns></returns>
            public string[] EnumerateSQLServersDatabases()
            {
                  return
RetrieveInformation(SQL_DRIVER_STR+";SERVER="+ m_SQLServer+";UID=" +
m_Username +";PWD=" +m_Password);
            }

            /// <summary>
            /// Enumerate for SQLServer/Databases based on the passed
information it the string
```

```
            /// The more information provided to SQLBrowseConnect the
more granular it gets so
            /// if only DRIVER=SQL SERVER passed then a list of all
SQL Servers is returned
            /// If DRIVER=SQL SERVER;Server=ServerName is passed then
a list of all Databases on the
            /// servers is returned etc
            /// </summary>
            /// <param name="InputParam">A valid string to query
for</param>
            /// <returns></returns>
            private string[] RetrieveInformation(string InputParam)
            {
                  IntPtr m_environmentHandle=IntPtr.Zero;
                  IntPtr m_connectionHandle = IntPtr.Zero;
                  StringBuilder inConnection = new
StringBuilder(InputParam);
                  short stringLength= (short)inConnection.Length;
                  StringBuilder outConnection = new
StringBuilder(DEFAULT_RESULT_SIZE);
                  short stringLength2Ptr= 0;

                  try
                  {
                        if (SQL_SUCCESS ==
SQLAllocHandle(SQL_HANDLE_ENV, m_environmentHandle, out
m_environmentHandle))
                        {
                              if (SQL_SUCCESS ==
SQLSetEnvAttr(m_environmentHandle,SQL_ATTR_ODBC_VERSION,(IntPtr)SQL_O
V_ODBC3,0))
                              {
                                    if (SQL_SUCCESS ==
SQLAllocHandle(SQL_HANDLE_DBC, m_environmentHandle, out
m_connectionHandle))
                                    {
                                          if (SQL_NEED_DATA ==
SQLBrowseConnect(m_connectionHandle, inConnection, stringLength,
outConnection, DEFAULT_RESULT_SIZE, out stringLength2Ptr))
                                          {
                                                if (SQL_NEED_DATA !=
SQLBrowseConnect(m_connectionHandle, inConnection, stringLength,
outConnection, DEFAULT_RESULT_SIZE, out stringLength2Ptr))
                                                {
                                                      throw new
ApplicationException("No Data Returned.");
                                                }
                                          }
                                    }
                              }
                        }
                  }

                  catch (Exception ex)
                  {
                        throw new ApplicationException("Cannot Locate
SQL Server.");
                  }
                  finally
                  {
                        FreeConnection(m_connectionHandle);
```

```
                        FreeConnection(m_environmentHandle);
                    }
                    if (outConnection.ToString()!="")
                    {return
ParseSQLOutConnection(outConnection.ToString());}
                    else{return null;}


            }
            /// <summary>
            /// Parse an outConnection string returned from
SQLBrowseConnect
            /// </summary>
            /// <param name="outConnection">string to parse</param>
            /// <returns></returns>
            private string[] ParseSQLOutConnection(string
outConnection)
            {
                  int m_Start = outConnection.IndexOf(START_STR) + 1;
                  int m_lenString = outConnection.IndexOf(END_STR) -
m_Start;
            if ((m_Start > 0) && (m_lenString > 0))
            {
                outConnection = outConnection.Substring(m_Start,
m_lenString);
            }
            else
            {
                // outConnection = string.Empty;
                outConnection = outConnection.Substring(m_Start);
            }
                  return outConnection.Split(",".ToCharArray());
            }
            private void FreeConnection(IntPtr handleToFree)
            {
                        if(handleToFree!= IntPtr.Zero)

      SQLFreeHandle(SQL_HANDLE_DBC,handleToFree);
            }
```

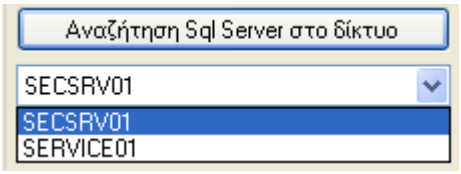
Ανάλυση φόρμας 1



Εικόνα 4 Κουμπί αναζήτησης SQL server

```
private void btnLoadSqlServers_Click(object sender, EventArgs e)
        {
            ticker.Start();
            btnLoadSqlServers.Enabled = false;
            this.Cursor = Cursors.WaitCursor;
            cmbSqlServers.Items.Clear();
            called = CallFor.SqlServerList;
            intlDelg.BeginInvoke(new AsyncCallback(CallBackMethod),
intlDelg);
        }
```

Πατώντας το κουμπί της εικόνας 4 τρέχει ο πιο πάνω κώδικας με αποτέλεσμα να φορτώνεται η λίστα με τους διαθέσιμους SQL Servers όπως φαίνεται παρακάτω

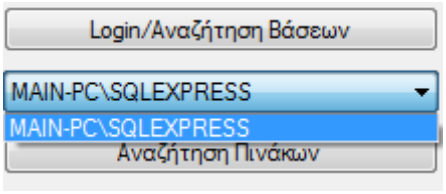


Εικόνα 5 Combobox με αποτελέσματα

Μετά την επιλογή του διαθέσιμου SQL Server ο χρήστης πρέπει να βάλει στα κατάλληλα πεδία το username και το password της σύνδεσης ώστε να συνδεθεί στη βάση.

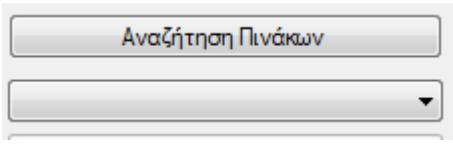
Εικόνα 6 Πεδία σύνδεσης SQL Server

Στη συνέχεια ο χρήστης αν έχει τοποθετήσει σωστά τα στοιχεία στα πεδία πατώντας το κουμπί *Login/Αναζήτηση Βάσεων* θα του επιστρέψει όλες τις διαθέσιμες βάσεις όπως φαίνεται στη παρακάτω εικόνα.



Εικόνα 7 Αναζήτηση Βάσεων και Επιλογή Βάσης

Τέλος, το μόνο που έχει να κάνει ο χρήστης είναι να επιλέξει τον πίνακα που θέλει να εμφανίσει και πατώντας το πλήκτρο *δεδομένα* θα του φέρει τις πρώτες εγγραφές του πίνακα.



Εικόνα 8 Αναζήτηση και Επιλογή Πίνακα

3.3.Κώδικας διαλόγου φόρμας 1

Παρακάτω παρατίθεται αναλυτικά ο κώδικας της πρώτης φόρμας που αναφέραμε στη προηγούμενη ενότητα. Επίσης κατά τη διάρκεια των ερωτοαπαντήσεων υπάρχει μια μπάρα προόδου που "μετράει" τον χρόνο αναζήτησης. (Εικόνα 9)

Εικόνα 9 Μπάρα προόδου

```
  private void btnLoadSqlServers_Click(object sender, EventArgs e)
        {
            ticker.Start();
            btnLoadSqlServers.Enabled = false;
            this.Cursor = Cursors.WaitCursor;
            cmbSqlServers.Items.Clear();
            called = CallFor.SqlServerList;
            intlDelg.BeginInvoke(new AsyncCallback(CallBackMethod),
intlDelg);
        }

        void ticker_Elapsed(object sender, ElapsedEventArgs e)
        {
            if (this.InvokeRequired)
            {
                this.Invoke(new TimerDelegate(ticker_Elapsed),
sender, e);
            }
            else
            {
                if (prgProgress.Value == prgProgress.Maximum)
                {
                    prgProgress.Value = 0;
                }
                else
                {
                    prgProgress.Value += 20;
                }
            }
        }




        private void btnGetAllDataBases_Click(object sender,
EventArgs e)
        {
            if (cmbSqlServers.Items.Count > 0 &&
                txtPassword.Text.Trim().CompareTo(string.Empty) != 0
&&
```

```
                txtUserName.Text.Trim().CompareTo(string.Empty) != 0)
            {
                ticker.Start();
                btnGetAllDataBases.Enabled = false;
                this.Cursor = Cursors.WaitCursor;
                cmbAllDataBases.Items.Clear();
                sqlInfo.SQLServer = cmbSqlServers.Text.Trim();
                sqlInfo.Username = txtUserName.Text.Trim();
                sqlInfo.Password = txtPassword.Text.Trim();
                MessageBox.Show("You may get the list sql servers if
user name and password are not correct.", "Information",
                    MessageBoxButtons.OK, MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1, 0, false);
                intlDelg = new
InternalDelegate(sqlInfo.EnumerateSQLServersDatabases);
                called = CallFor.SqlDataBases;
                intlDelg.BeginInvoke(new
AsyncCallback(CallBackMethod), intlDelg);
            }
            else
            {
                MessageBox.Show("Δεν υπάρχει εγκατεστημένος SQL
Server ή τα πεδία Username, Password είναι κενά!", "Σφάλμα",
MessageBoxButtons.OK, MessageBoxIcon.Warning,
MessageBoxDefaultButton.Button1, 0, false);
            }
        }
```

```
        private void btnGetAllTables_Click(object sender, EventArgs
e)
        {
            if (cmbAllDataBases.Text.Trim().CompareTo(string.Empty)
!= 0)
            {
                btnGetAllTables.Enabled = false;
                this.Cursor = Cursors.WaitCursor;
                StringBuilder connStr = new StringBuilder();
                connStr.Append("Database=");
                connStr.Append(cmbAllDataBases.Text);
                connStr.Append(";Server=");
                connStr.Append(cmbSqlServers.Text);
                connStr.Append(";User=");
                connStr.Append(txtUserName.Text.Trim());
                connStr.Append(";Password=");
                connStr.Append(txtPassword.Text.Trim());
                connStr.Append(";Enlist=false; Asynchronous
Processing=true");
                connectionString = connStr.ToString();
                sqlQuery = getTablesFromDataBase;
                SetDataObjects();
                try
                {
```

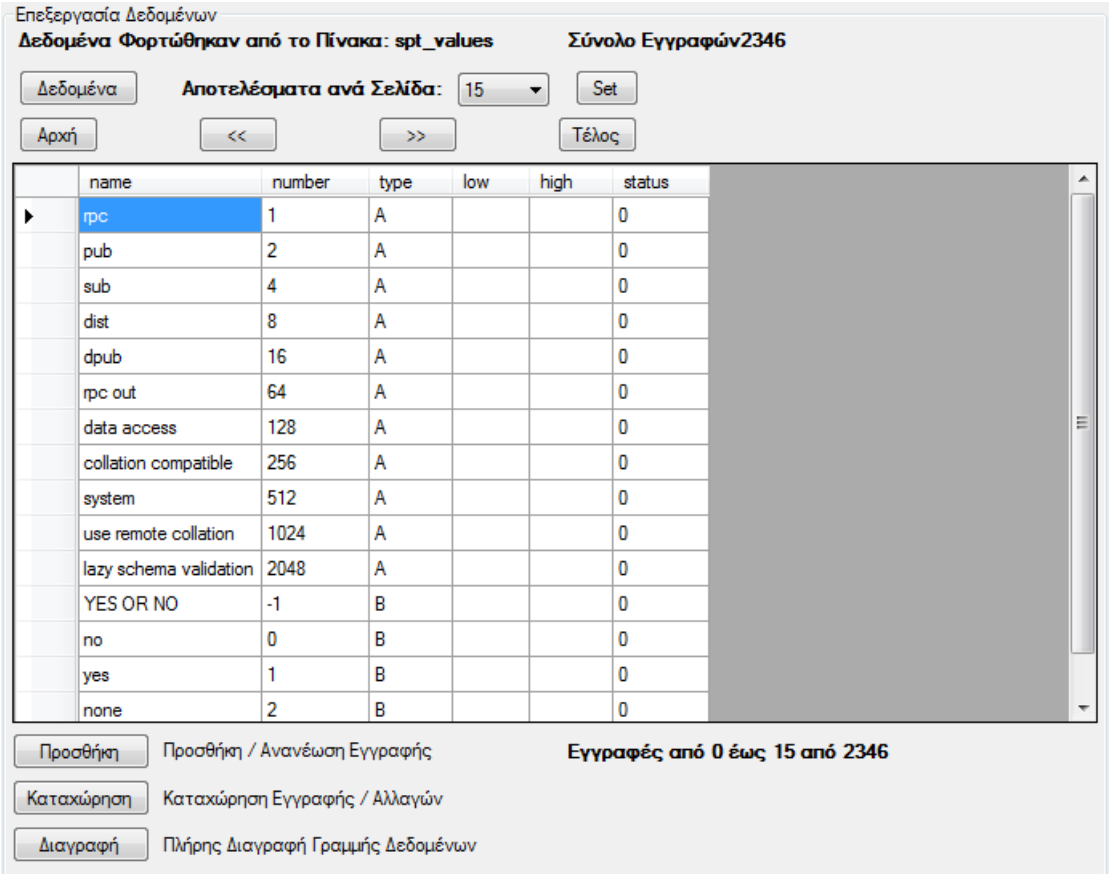
```
                    ticker.Start();
                    connection.Open();
                    command.BeginExecuteReader(new
AsyncCallback(CallBackMethod),
                        command, CommandBehavior.CloseConnection);
                    called = CallFor.SqlTables;
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                    ticker.Stop();
                }
                finally
                {
                    btnGetAllTables.Enabled = true;
                    this.Cursor = Cursors.Default;
                }
            }
            else
            {
                MessageBox.Show("Διαλέξτε Βάση Δεδομένων",
                    "Βάση Δεδομένων Κενή", MessageBoxButtons.OK,
MessageBoxIcon.Warning,
                    MessageBoxDefaultButton.Button1, 0, false);
            }
        }
```

```
        private void CallBackMethod(IAsyncResult result)
        {
            if (this.InvokeRequired)
            {
                this.Invoke(new AsyncDelegate(CallBackMethod),
result);
            }
            else
            {
                try
                {
                    prgProgress.Value = prgProgress.Maximum;
                    switch (called)
                    {
                        case CallFor.SqlServerList:
                            string[] sqlServers =
intlDelg.EndInvoke(result);
                            cmbSqlServers.Items.AddRange(sqlServers);
                            if (cmbSqlServers.Items.Count > 0)
                            {
                                cmbSqlServers.Sorted = true;
                                cmbSqlServers.SelectedIndex = 0;
                            }
```

```
                            this.Cursor = Cursors.Default;
                            btnLoadSqlServers.Enabled = true;
                            txtUserName.Select();
                            txtUserName.Focus();
                            break;
                        case CallFor.SqlDataBases:
                            string[] sqlDatabases =
intlDelg.EndInvoke(result);

cmbAllDataBases.Items.AddRange(sqlDatabases);
                            if (cmbAllDataBases.Items.Count > 0)
                            {
                                cmbAllDataBases.Sorted = true;
                                cmbAllDataBases.SelectedIndex = 0;
                            }
                            this.Cursor = Cursors.Default;
                            btnGetAllDataBases.Enabled = true;
                            break;
                        case CallFor.SqlTables:
                            reader =
command.EndExecuteReader(result);
                            cmbTables.Items.Clear();
                            while (reader.Read())
                            {

cmbTables.Items.Add(reader[0].ToString());
                            }
                            if (cmbTables.Items.Count > 0)
                            {
                                cmbTables.Sorted = true;
                                cmbTables.SelectedIndex = 0;
                                grpDataManipulate.Enabled = true;
                            }
                            else
                            {
                                grpDataManipulate.Enabled = false;
                            }
                            break;
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
                finally
                {
                    if (called == CallFor.SqlTables)
                    {
                        btnGetAllTables.Enabled = true;
                        this.Cursor = Cursors.Default;
                    }
                    prgProgress.Value = 0;
                    prgProgress.Refresh();
                    ticker.Stop();
                }
            }
        }
```

3.4. Κώδικας διαλόγου φόρμας 2

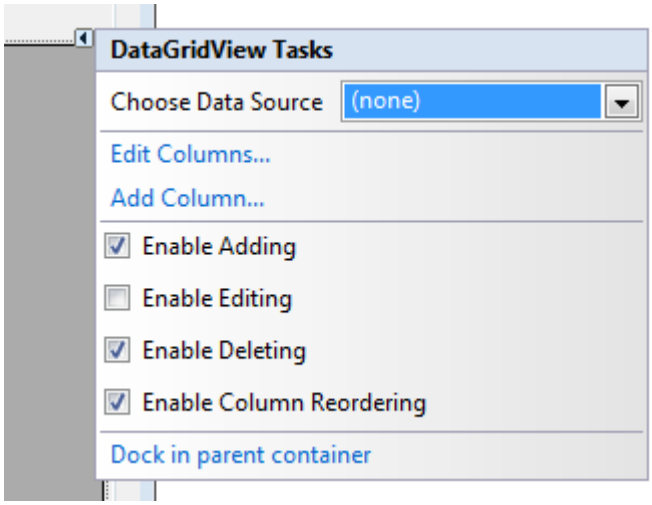
Στη δεύτερη φόρμα του προγράμματος φορτώνεται ο πίνακας που έχει επιλέξει ο χρήστης να δει. Τα δεδομένα παρουσιάζονται σε πίνακα δομής DataGridView με τον χρήστη να έχει τη δυνατότητα να επιλέξει πόσες εγγραφές θα βλέπει σε μία σελίδα μέχρι να περάσει στη επόμενη.



Εικόνα 10 Φόρμα 2 Παρουσίασης Δεδομένων

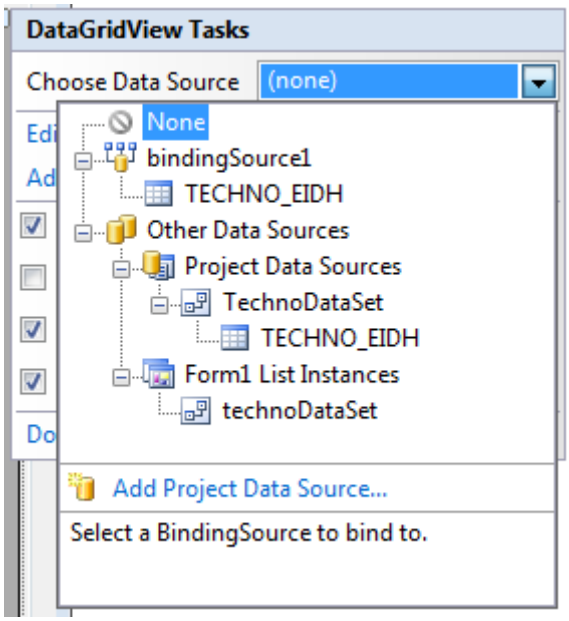
Εδώ θα αναλύσουμε την δομή και την λειτουργία του DataGridView. Το DataGridView είναι ένα στοιχείο ελέγχου που έχει σχεδιαστεί ώστε να είναι βέβαιο ότι οι πίνακες της βάσης δεδομένων θα φορτωθούν. Θα βρείτε το DataGridView στην ομάδα Στοιχεία για Toolbox σας. Προτιμάτε να χρησιμοποιείται διότι θυμίζει η μορφή του Excel και έτσι φαίνεται οικείο στον χρήστη.

Στον προγραμματιστή η χρήση του δίνει κάποιες συγκεκριμένες ευκολίες, όπως ο ορισμός των στηλών που θα εμφανίσει ο πίνακας αν δε θέλουμε να εμφανίσουμε όλο το πίνακα, δίνει απευθείας ταξινόμηση στη στήλη αν το επιθυμεί ο χρήστης κ.α. Η αρχική παραμετροποίηση του γίνεται από ένα βελάκι που εμφανίζεται μετά την επιλογή του, όπως φαίνεται στη παρακάτω εικόνα.



Εικόνα 11 Αρχική Παραμετροποίηση DataGridView

Όπως φαίνεται και στην εικόνα αμέσως με ορισμένα "κλικ" μπορούμε να ορίσουμε αν θα επιτρέπουμε στον πίνακα μας να προστεθούν δεδομένα, να επεξεργαστούν δεδομένα, να διαγραφούν και να ταξινομηθούν. Επίσης από το πεδίο *Choose Data Source* μπορεί να οριστεί απ' ευθείας η βάση που θα "τραβάει" τα δεδομένα. Εδώ να αναφέρουμε ότι στο συγκεκριμένα project έχει χρησιμοποιηθεί query.



Εικόνα 12 Επιλογή Data Source

Παρακάτω υπάρχουν κάποια στιγμιότυπα κώδικα που φτιάχνει μόνο του το DataGridView μόνο επιλέγοντας το και επιλέγοντας τη βασική παραμετροποίηση.

```
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected
TechnoDataSet(global::System.Runtime.Serialization.SerializationInfo
info, global::System.Runtime.Serialization.StreamingContext context)
:
                base(info, context, false) {
            if ((this.IsBinarySerialized(info, context) == true)) {
                this.InitVars(false);

global::System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler1 = new
global::System.ComponentModel.CollectionChangeEventHandler(this.Schem
aChanged);
                this.Tables.CollectionChanged +=
schemaChangedHandler1;
                this.Relations.CollectionChanged +=
schemaChangedHandler1;
                return;
            }
            string strSchema = ((string)(info.GetValue("XmlSchema",
typeof(string))));
            if ((this.DetermineSchemaSerializationMode(info, context)
== global::System.Data.SchemaSerializationMode.IncludeSchema)) {
                global::System.Data.DataSet ds = new
global::System.Data.DataSet();
                ds.ReadXmlSchema(new
global::System.Xml.XmlTextReader(new
global::System.IO.StringReader(strSchema)));
                if ((ds.Tables["TECHNO_EIDH"] != null)) {
                    base.Tables.Add(new
TECHNO_EIDHDataTable(ds.Tables["TECHNO_EIDH"]));
                }
                this.DataSetName = ds.DataSetName;
                this.Prefix = ds.Prefix;
                this.Namespace = ds.Namespace;
                this.Locale = ds.Locale;
                this.CaseSensitive = ds.CaseSensitive;
                this.EnforceConstraints = ds.EnforceConstraints;
                this.Merge(ds, false,
global::System.Data.MissingSchemaAction.Add);
                this.InitVars();
            }
            else {
                this.ReadXmlSchema(new
global::System.Xml.XmlTextReader(new
global::System.IO.StringReader(strSchema)));
            }
            this.GetSerializationData(info, context);

global::System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler = new
global::System.ComponentModel.CollectionChangeEventHandler(this.Schem
aChanged);
            base.Tables.CollectionChanged += schemaChangedHandler;
            this.Relations.CollectionChanged += schemaChangedHandler;
```

```
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected override void
ReadXmlSerializable(global::System.Xml.XmlReader reader) {
            if ((this.DetermineSchemaSerializationMode(reader) ==
global::System.Data.SchemaSerializationMode.IncludeSchema)) {
                this.Reset();
                global::System.Data.DataSet ds = new
global::System.Data.DataSet();
                ds.ReadXml(reader);
                if ((ds.Tables["TECHNO_EIDH"] != null)) {
                    base.Tables.Add(new
TECHNO_EIDHDataTable(ds.Tables["TECHNO_EIDH"]));
                }
                this.DataSetName = ds.DataSetName;
                this.Prefix = ds.Prefix;
                this.Namespace = ds.Namespace;
                this.Locale = ds.Locale;
                this.CaseSensitive = ds.CaseSensitive;
                this.EnforceConstraints = ds.EnforceConstraints;
                this.Merge(ds, false,
global::System.Data.MissingSchemaAction.Add);
                this.InitVars();
            }
            else {
                this.ReadXml(reader);
                this.InitVars();
            }
           }
```

```
public TECHNO_EIDHRow AddTECHNO_EIDHRow(string CODE, string DESCR,
double PRIMARYQTY, double COUNTQTY, System.DateTime APOGDATE, int
STOID) {
                TECHNO_EIDHRow rowTECHNO_EIDHRow =
((TECHNO_EIDHRow)(this.NewRow()));
                object[] columnValuesArray = new object[] {
                        null,
                        CODE,
                        DESCR,
                        PRIMARYQTY,
                        COUNTQTY,
                        APOGDATE,
                        STOID};
                rowTECHNO_EIDHRow.ItemArray = columnValuesArray;
                this.Rows.Add(rowTECHNO_EIDHRow);
                return rowTECHNO_EIDHRow;
            }


[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
            internal void InitVars() {
                this.columnID = base.Columns["ID"];
                this.columnCODE = base.Columns["CODE"];
                this.columnDESCR = base.Columns["DESCR"];
                this.columnPRIMARYQTY = base.Columns["PRIMARYQTY"];
                this.columnCOUNTQTY = base.Columns["COUNTQTY"];
                this.columnAPOGDATE = base.Columns["APOGDATE"];
                this.columnSTOID = base.Columns["STOID"];
            }


[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
            private void InitClass() {
                this.columnID = new
global::System.Data.DataColumn("ID", typeof(int), null,
global::System.Data.MappingType.Element);
                base.Columns.Add(this.columnID);
                this.columnCODE = new
global::System.Data.DataColumn("CODE", typeof(string), null,
global::System.Data.MappingType.Element);
                base.Columns.Add(this.columnCODE);
                this.columnDESCR = new
global::System.Data.DataColumn("DESCR", typeof(string), null,
global::System.Data.MappingType.Element);
                base.Columns.Add(this.columnDESCR);
                this.columnPRIMARYQTY = new
global::System.Data.DataColumn("PRIMARYQTY", typeof(double), null,
global::System.Data.MappingType.Element);
                base.Columns.Add(this.columnPRIMARYQTY);
                this.columnCOUNTQTY = new
global::System.Data.DataColumn("COUNTQTY", typeof(double), null,
global::System.Data.MappingType.Element);
                base.Columns.Add(this.columnCOUNTQTY);
                this.columnAPOGDATE = new
global::System.Data.DataColumn("APOGDATE",
typeof(global::System.DateTime), null,
global::System.Data.MappingType.Element);
                base.Columns.Add(this.columnAPOGDATE);
                this.columnSTOID = new
global::System.Data.DataColumn("STOID", typeof(int), null,
global::System.Data.MappingType.Element);
```

```
                base.Columns.Add(this.columnSTOID);
                this.Constraints.Add(new
global::System.Data.UniqueConstraint("Constraint1", new
global::System.Data.DataColumn[] {
                                this.columnID}, true));
                this.columnID.AutoIncrement = true;
                this.columnID.AutoIncrementSeed = -1;
                this.columnID.AutoIncrementStep = -1;
                this.columnID.AllowDBNull = false;
                this.columnID.ReadOnly = true;
                this.columnID.Unique = true;
                this.columnCODE.AllowDBNull = false;
                this.columnCODE.MaxLength = 25;
                this.columnDESCR.AllowDBNull = false;
                this.columnDESCR.MaxLength = 50;
               }


private void InitAdapter() {
            this._adapter = new
global::System.Data.SqlClient.SqlDataAdapter();
            global::System.Data.Common.DataTableMapping tableMapping
= new global::System.Data.Common.DataTableMapping();
            tableMapping.SourceTable = "Table";
            tableMapping.DataSetTable = "TECHNO_EIDH";
            tableMapping.ColumnMappings.Add("ID", "ID");
            tableMapping.ColumnMappings.Add("CODE", "CODE");
            tableMapping.ColumnMappings.Add("DESCR", "DESCR");
            tableMapping.ColumnMappings.Add("PRIMARYQTY",
"PRIMARYQTY");
            tableMapping.ColumnMappings.Add("COUNTQTY", "COUNTQTY");
            tableMapping.ColumnMappings.Add("APOGDATE", "APOGDATE");
            tableMapping.ColumnMappings.Add("STOID", "STOID");
            this._adapter.TableMappings.Add(tableMapping);
            this._adapter.DeleteCommand = new
global::System.Data.SqlClient.SqlCommand();
            this._adapter.DeleteCommand.Connection = this.Connection;
            this._adapter.DeleteCommand.CommandText = @"DELETE FROM
[dbo].[TECHNO_EIDH] WHERE (([ID] = @Original_ID) AND ([CODE] =
@Original_CODE) AND ([DESCR] = @Original_DESCR) AND
((@IsNull_PRIMARYQTY = 1 AND [PRIMARYQTY] IS NULL) OR ([PRIMARYQTY] =
@Original_PRIMARYQTY)) AND ((@IsNull_COUNTQTY = 1 AND [COUNTQTY] IS
NULL) OR ([COUNTQTY] = @Original_COUNTQTY)) AND ((@IsNull_APOGDATE =
1 AND [APOGDATE] IS NULL) OR ([APOGDATE] = @Original_APOGDATE)) AND
((@IsNull_STOID = 1 AND [STOID] IS NULL) OR ([STOID] =
@Original_STOID)))";
            this._adapter.DeleteCommand.CommandType =
global::System.Data.CommandType.Text;
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_ID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "ID",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_CODE",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "CODE",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
```

```
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_DESCR",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "DESCR",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_PRIMARYQTY",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "PRIMARYQTY",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_PRIMARYQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "PRIMARYQTY",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_COUNTQTY",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "COUNTQTY",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_COUNTQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "COUNTQTY",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_APOGDATE",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "APOGDATE",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_APOGDATE",
global::System.Data.SqlDbType.DateTime, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "APOGDATE",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_STOID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "STOID",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.DeleteCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_STOID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "STOID",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.InsertCommand = new
global::System.Data.SqlClient.SqlCommand();
            this._adapter.InsertCommand.Connection = this.Connection;
            this._adapter.InsertCommand.CommandText = @"INSERT INTO
[dbo].[TECHNO_EIDH] ([CODE], [DESCR], [PRIMARYQTY], [COUNTQTY],
[APOGDATE], [STOID]) VALUES (@CODE, @DESCR, @PRIMARYQTY, @COUNTQTY,
@APOGDATE, @STOID);
```

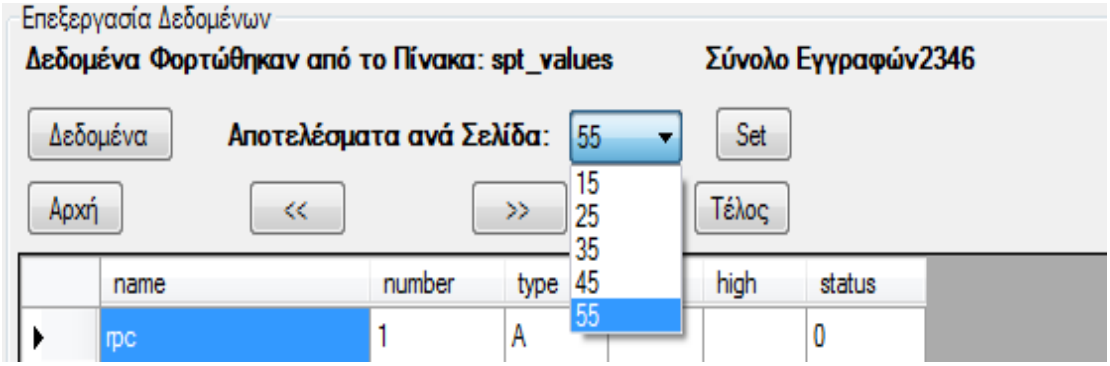
```
SELECT ID, CODE, DESCR, PRIMARYQTY, COUNTQTY, APOGDATE, STOID FROM
TECHNO_EIDH WHERE (ID = SCOPE_IDENTITY())";
            this._adapter.InsertCommand.CommandType =
global::System.Data.CommandType.Text;
            this._adapter.InsertCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@CODE",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "CODE",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.InsertCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@DESCR",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "DESCR",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.InsertCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@PRIMARYQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "PRIMARYQTY",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.InsertCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@COUNTQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "COUNTQTY",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.InsertCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@APOGDATE",
global::System.Data.SqlDbType.DateTime, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "APOGDATE",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.InsertCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@STOID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "STOID",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand = new
global::System.Data.SqlClient.SqlCommand();
            this._adapter.UpdateCommand.Connection = this.Connection;
            this._adapter.UpdateCommand.CommandText = @"UPDATE
[dbo].[TECHNO_EIDH] SET [CODE] = @CODE, [DESCR] = @DESCR,
[PRIMARYQTY] = @PRIMARYQTY, [COUNTQTY] = @COUNTQTY, [APOGDATE] =
@APOGDATE, [STOID] = @STOID WHERE (([ID] = @Original_ID) AND ([CODE]
= @Original_CODE) AND ([DESCR] = @Original_DESCR) AND
((@IsNull_PRIMARYQTY = 1 AND [PRIMARYQTY] IS NULL) OR ([PRIMARYQTY] =
@Original_PRIMARYQTY)) AND ((@IsNull_COUNTQTY = 1 AND [COUNTQTY] IS
NULL) OR ([COUNTQTY] = @Original_COUNTQTY)) AND ((@IsNull_APOGDATE =
1 AND [APOGDATE] IS NULL) OR ([APOGDATE] = @Original_APOGDATE)) AND
((@IsNull_STOID = 1 AND [STOID] IS NULL) OR ([STOID] =
@Original_STOID)));
SELECT ID, CODE, DESCR, PRIMARYQTY, COUNTQTY, APOGDATE, STOID FROM
TECHNO_EIDH WHERE (ID = @ID)";
            this._adapter.UpdateCommand.CommandType =
global::System.Data.CommandType.Text;
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@CODE",
global::System.Data.SqlDbType.VarChar, 0,
```

```
global::System.Data.ParameterDirection.Input, 0, 0, "CODE",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@DESCR",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "DESCR",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@PRIMARYQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "PRIMARYQTY",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@COUNTQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "COUNTQTY",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@APOGDATE",
global::System.Data.SqlDbType.DateTime, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "APOGDATE",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@STOID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "STOID",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_ID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "ID",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_CODE",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "CODE",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_DESCR",
global::System.Data.SqlDbType.VarChar, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "DESCR",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_PRIMARYQTY",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "PRIMARYQTY",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_PRIMARYQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "PRIMARYQTY",
```

```
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_COUNTQTY",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "COUNTQTY",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_COUNTQTY",
global::System.Data.SqlDbType.Float, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "COUNTQTY",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_APOGDATE",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "APOGDATE",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_APOGDATE",
global::System.Data.SqlDbType.DateTime, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "APOGDATE",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@IsNull_STOID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "STOID",
global::System.Data.DataRowVersion.Original, true, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@Original_STOID",
global::System.Data.SqlDbType.Int, 0,
global::System.Data.ParameterDirection.Input, 0, 0, "STOID",
global::System.Data.DataRowVersion.Original, false, null, "", "",
""));
            this._adapter.UpdateCommand.Parameters.Add(new
global::System.Data.SqlClient.SqlParameter("@ID",
global::System.Data.SqlDbType.Int, 4,
global::System.Data.ParameterDirection.Input, 0, 0, "ID",
global::System.Data.DataRowVersion.Current, false, null, "", "",
""));
        }
```

Βλέποντας τον κώδικα παραπάνω κατανοεί κάποιος την τεράστια ευκολία που δίνει στον προγραμματιστή το Microsoft Visual Studio, καθώς του προσφέρει με το πάτημα ενός κουμπιού μεγάλο κέρδος σε εξοικονόμηση ώρας ανάπτυξης του προγράμματος.

Τα κουμπιά στη πάνω άκρη της φόρμας δίνουν τη δυνατότητα στο χρήστη να ορίσει τον αριθμό των αποτελεσμάτων που θα εμφανίζει ανα σελίδα και την περιήγηση μέσα στη βάση

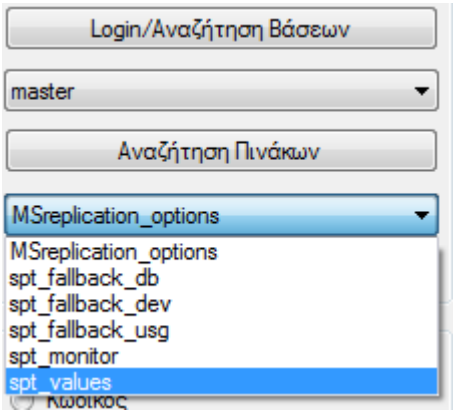


Εικόνα 13 Κουμπιά περιήγησης βάσης δεδομένων

```
private void btnLoad_Click(object sender, EventArgs e)
        {
            lblLoadedTable.Text = "Ανάγνωση Δεδομένων από" +
cmbTables.Text.Trim();
            btnLoad.Enabled = false;
            this.Cursor = Cursors.WaitCursor;
            try
            {
                if (userTable != null)
                {
                    userTable.Clear();
                }
                bindingSource1.DataSource = null;
                userDataGridView.DataSource = bindingSource1;
                userDataGridView.Rows.Clear();
                userDataGridView.Refresh();
                sqlQuery = "SELECT * FROM [" + cmbTables.Text.Trim()
+ "]";
                SetDataObjects();
                connection.Open();
                ticker.Start();
                adapter.Fill(tempDataSet);
                totalRecords = tempDataSet.Tables[0].Rows.Count;
                tempDataSet.Clear();
                tempDataSet.Dispose();
                adapter.Fill(ds, 0, 5, cmbTables.Text.Trim());
                userTable = ds.Tables[cmbTables.Text.Trim()];

                foreach (DataColumn dc in userTable.Columns)
                {
                    DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn();
                    column.DataPropertyName = dc.ColumnName;
```

```
                    column.HeaderText = dc.ColumnName;
                    column.Name = dc.ColumnName;
                    column.SortMode =
DataGridViewColumnSortMode.Automatic;
                    column.ValueType = dc.DataType;
                    userDataGridView.Columns.Add(column);
                }
                lblLoadedTable.Text = "Δεδομένα Φορτώθηκαν από το
Πίνακα: " + userTable.TableName;
                lblTotRecords.Text = "Σύνολο Εγγραφών" +
totalRecords;
                CreateTempTable(0,
int.Parse(cmbNoOfRecords.Text.Trim()));

                btnPrevious.Enabled = true;
                btnFirst.Enabled = true;
                btnPrevious.Enabled = true;
                btnNext.Enabled = true;
                btnLast.Enabled = true;
                btnAdd.Enabled = true;
                btnUpdate.Enabled = true;
                btnDelete.Enabled = true;
                cmbNoOfRecords.Enabled = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                connection.Close();
                btnLoad.Enabled = true;
                this.Cursor = Cursors.Default;
                prgProgress.Value = 0;
                prgProgress.Update();
                prgProgress.Refresh();
                ticker.Stop();
            }
        }
```

Στον κώδικα που φαίνεται πιο πάνω διακρίνουμε ότι στο πίνακα φορτώνονται τα δεδομένα που έχει επιλέξει ο χρήστης από το διάλογο της φόρμας ένα και όχι "καρφωτά" από την επιλογή πηγής του DataGridView :



Εικόνα 14 Επιλογή πίνακα

```
private void CreateTempTable(int startRecord, int noOfRecords)
        {
            if (startRecord == 0 || startRecord < 0)
            {
                btnPrevious.Enabled = false;
                startRecord = 0;
            }
            int endRecord = startRecord + noOfRecords;
            if (endRecord >= totalRecords)
            {
                btnNext.Enabled = false;
                isLastPage = true;
                endRecord = totalRecords;
            }
            currentPageStartRecord = startRecord;
            currentPageEndRecord = endRecord;
            lblPageNums.Text = "Εγγραφές από " + startRecord + " έως
"
                + endRecord + " από " + totalRecords;
            currentIndex = endRecord;

            try
            {
                userTable.Rows.Clear();
                if (connection.State == ConnectionState.Closed)
                {
                    connection.Open();
                }
                adapter.Fill(ds, startRecord, noOfRecords,
cmbTables.Text.Trim());
                userTable = ds.Tables[cmbTables.Text.Trim()];
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                connection.Close();
            }
```

```
            bindingSource1.DataSource = userTable.DefaultView;
            userDataGridView.AllowUserToResizeColumns = true;

        }
```

Ο κώδικας που αναφέρεται στις δύο πιο πάνω σελιδες φορτώνεται αυτόματα πατώντας το κουμπί δεδομένα.

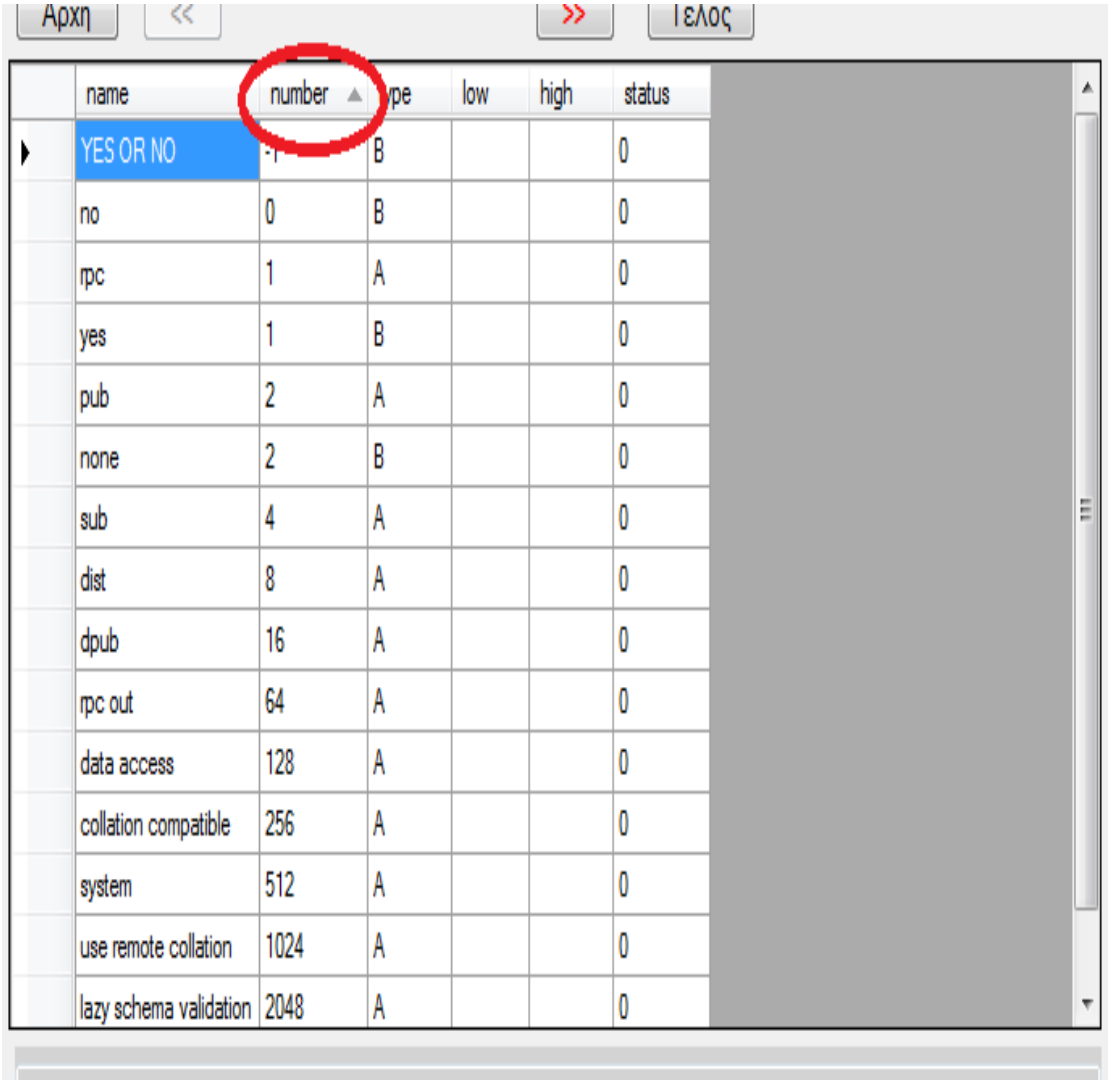
Παρακάτω δίνεται ο κωδικάς για τα κουμπιά πλοήγησης.

```
private void btnFirst_Click(object sender, EventArgs e)
        {
            CreateTempTable(0, int.Parse(cmbNoOfRecords.Text));
            btnPrevious.Enabled = false;
            btnNext.Enabled = true;
            btnLast.Enabled = true;
            isLastPage = false;
        }
```

```
private void btnPrevious_Click(object sender, EventArgs e)
        {
            if (isLastPage)
            {
                int noc = FindLastPageRecords();
                CreateTempTable((totalRecords - noc -
int.Parse(cmbNoOfRecords.Text)),
                    int.Parse(cmbNoOfRecords.Text));
            }
            else
            {
                CreateTempTable((currentIndex - 2 *
(int.Parse(cmbNoOfRecords.Text))),
                    int.Parse(cmbNoOfRecords.Text));
            }
            btnNext.Enabled = true;
            btnLast.Enabled = true;
            isLastPage = false;
        }
```

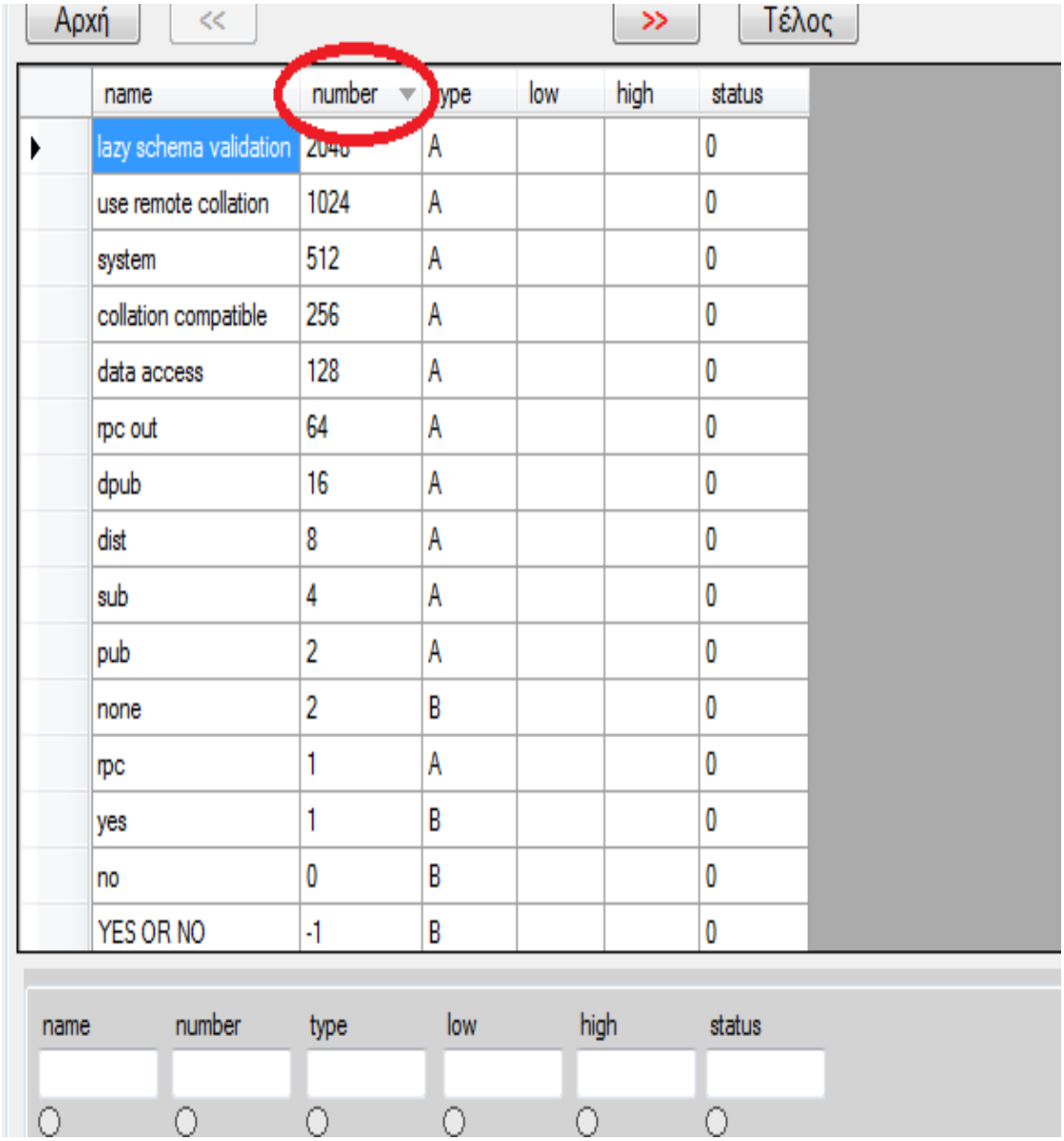
```
        private void btnNext_Click(object sender, EventArgs e)
        {
            CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
            btnPrevious.Enabled = true;
        }




        private void btnLast_Click(object sender, EventArgs e)
        {
            int totPages = totalRecords /
int.Parse(cmbNoOfRecords.Text);
            int remainingRecs = FindLastPageRecords();

            CreateTempTable(totalRecords - remainingRecs,
int.Parse(cmbNoOfRecords.Text));
            btnPrevious.Enabled = true;
            btnNext.Enabled = false;
            isLastPage = true;
        }






        private int FindLastPageRecords()
        {
            return (totalRecords % int.Parse(cmbNoOfRecords.Text));
        }

        private void btnNoOfPages_Click(object sender, EventArgs e)
        {
            CreateTempTable(0,
int.Parse(cmbNoOfRecords.Text.Trim()));
            if (int.Parse(cmbNoOfRecords.Text.Trim()) >=
totalRecords)
            {
                btnFirst.Enabled = false;
                btnPrevious.Enabled = false;
                btnNext.Enabled = false;
                btnLast.Enabled = false;
            }
            else
            {
                btnFirst.Enabled = true;
                btnPrevious.Enabled = true;
                btnNext.Enabled = true;
                btnLast.Enabled = true;
            }

        }
```

Παρακάτω δίνεται ένα παράδειγμα ταξινόμησης σε πίνακα πατώντας με το ποντίκι στη στήλη που μας ενδιαφέρει. Η ταξινόμηση γίνεται αμέσως σε όλες τις εγγραφές είτε σε φθίνουσα σειρά, είτε σε αύξουσα.



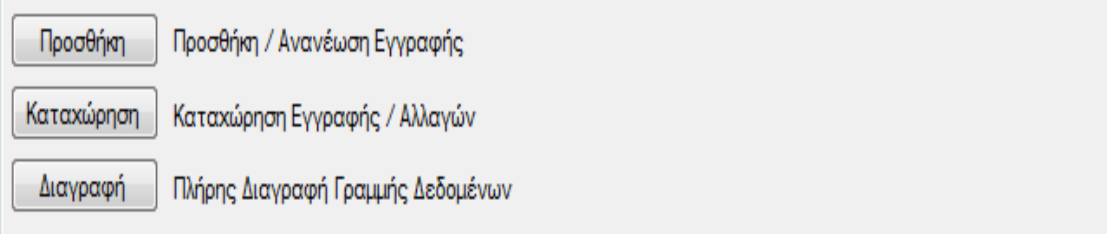
name	number	type	low	high	status
YES OR NO	-1	B			0
no	0	B			0
rpc	1	A			0
yes	1	B			0
pub	2	A			0
none	2	B			0
sub	4	A			0
dist	8	A			0
dpub	16	A			0
rpc out	64	A			0
data access	128	A			0
collation compatible	256	A			0
system	512	A			0
use remote collation	1024	A			0
lazy schema validation	2048	A			0

Εικόνα 15 Ταξινόμηση με αύξουσα σειρά



Εικόνα 16 Ταξινόμηση με φθίνουσα σειρά

Αφού ο χρήστης φορτώσει τον πίνακα που θέλει και ρυθμίσει πόσες εγγραφές θα βλέπει ανά σελίδα, στη συνέχεια μπορεί είτε να προσθέσει εγγραφές στον πίνακα, είτε να μεταβάλει τις ήδη υπάρχουσες, είτε να διαγράψει μια ολόκληρη εγγραφή.



Εικόνα 17 Κουμπιά επεξεργασίας εγγραφών

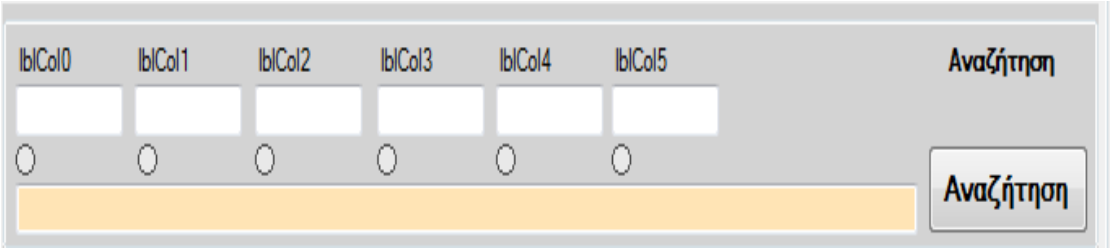
```
private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
                userDataGridView.ReadOnly = false;
                btnAdd.Enabled = false;
                btnUpdate.Enabled = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
```

```
private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                connection.Open();
                adapter.Update(userTable);
                userDataGridView.ReadOnly = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
                btnUpdate.Enabled = true;
            }
            finally
            {
                btnAdd.Enabled = true;
                btnLoad.Enabled = true;
                connection.Close();
            }
        }
```

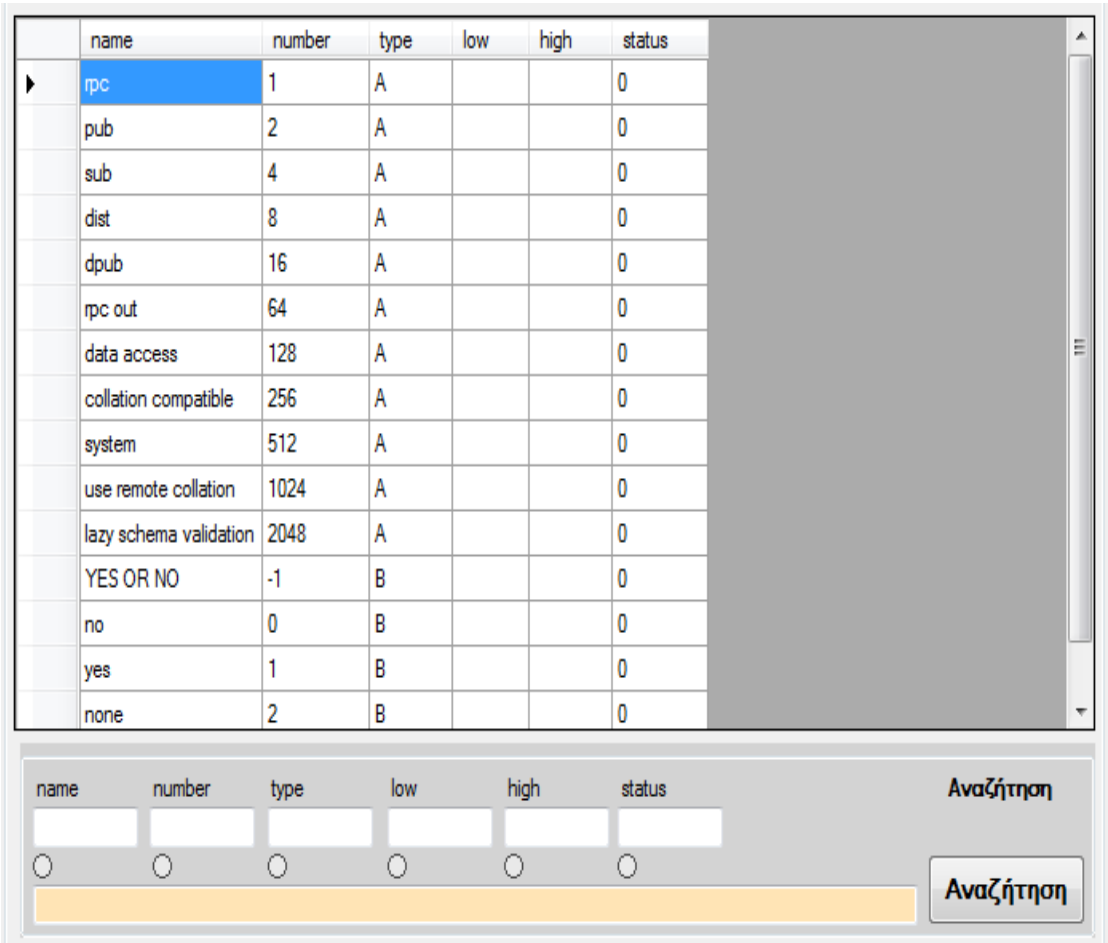
```
private void btnDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Είστε σίγουροι ότι θέλετε να
διαγράψετε τις εγγραφές;",
                "Προσοχή", MessageBoxButtons.YesNo,
                MessageBoxIcon.Warning,
MessageBoxDefaultButton.Button2, 0, false)
                == DialogResult.Yes)
            {
                try
                {
                    connection.Open();
                    int cnt = userDataGridView.SelectedRows.Count;
                    for (int i = 0; i < cnt; i++)
                    {
                        if (this.userDataGridView.SelectedRows.Count
> 0 &&

this.userDataGridView.SelectedRows[0].Index !=
                            this.userDataGridView.Rows.Count - 1)
                        {
                            this.userDataGridView.Rows.RemoveAt(

this.userDataGridView.SelectedRows[0].Index);
                        }
                    }

                    adapter.Update(userTable);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
                finally
                {
                    connection.Close();
                    btnLoad.Enabled = true;
                }
            }
           }
```

3.5. Κώδικας διαλόγου φόρμας 3

Η τρίτη φόρμα του προγράμματος περιέχει την αναζήτηση. Όταν φορτώνεται ο πίνακας διαβάζονται οι τίτλοι των στηλών και αναγράφονται με τη σειρά στη φόρμα της αναζήτησης, εκεί υπάρχει ένας περιορισμός όπου φορτώνονται μέχρι έξι στήλες. Ο χρήστης μπορεί να κάνει αναζήτηση σε μία εξ' αυτών. Το κουμπί αναζήτησης δουλεύει επίσης και σαν αναζήτηση επομένου.



Εικόνα 18 Φόρμα αναζήτησης



Εικόνα 19 Αντιστοιχία τίτλων-αναζήτηση

Παρακάτω δίνεται ο κώδικας που χρησιμοποιήθηκε για την αναζήτηση.

```
if (rcol1.Checked)
            {
                if (this.findTextBox1.Text.Length > 0)
                {
                    string col1 =
userDataGridView.Columns[0].HeaderText;

//MessageBox.Show(userDataGridView.Columns[0].ValueType.ToString());
                    String typoscol =
userDataGridView.Columns[0].ValueType.ToString();
                    //if (typoscol == "System.Int64")
                    if (typoscol != "System.String")
                    {
                     infosearch.Text = "Στον τύπο αυτό δεν
υποστηρίζεται μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";

                    int loc = bindingSource1.Find(col1,
this.findTextBox1.Text);


//MessageBox.Show(bindingSource1.Position.ToString());
                    while (loc == -1)
                     {
                   if (isLastPage == true)
                   {
                       loc = -1;
                       MessageBox.Show("Η αναζήτηση δεν ταιριάζει με
καμία εγγραφή");
                       break;

                   }
                   CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                   loc = bindingSource1.Find(col1,
this.findTextBox1.Text);
                   //MessageBox.Show(cmbNoOfRecords.Text);

                }

                bindingSource1.Position = loc;

                    }
                    else
                    {

                        int loc = bindingSource1.Find(col1,
this.findTextBox1.Text);
                        while (loc == -1)
                        {
                            DataView dv = new
DataView(this.userTable);

                            dv.RowFilter = string.Format(col1 + "
LIKE '{0}*'", this.findTextBox1.Text);
                            if (dv.Count > 0)
                            {
                                loc = bindingSource1.Find(col1,
dv[0][col1]);
                            }
```

```
                            else
                            {
                                if (isLastPage == true && loc == -1)
                                {
                                    loc = -1;
                                    MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                    break;
                                }
                                CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                                btnPrevious.Enabled = true;
                            }
                        }
                        bindingSource1.Position = loc;
                    }
                }

            }
            else if (rcol2.Checked)
            {

                if (this.findTextBox2.Text.Length > 0)
                {
                    string col1 =
userDataGridView.Columns[1].HeaderText;

//MessageBox.Show(userDataGridView.Columns[0].ValueType.ToString());
                    String typoscol =
userDataGridView.Columns[1].ValueType.ToString();

                    if (typoscol != "System.String")
                    {
                        infosearch.Text = "Στον τύπο αυτό δεν
υποστηρίζεται μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";

                        int loc = bindingSource1.Find(col1,
this.findTextBox2.Text);


//MessageBox.Show(bindingSource1.Position.ToString());
                        while (loc == -1)
                        {
                            if (isLastPage == true)
                            {
                                loc = -1;
                                MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                break;

                            }
                            CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                            loc = bindingSource1.Find(col1,
this.findTextBox2.Text);
                            //MessageBox.Show(cmbNoOfRecords.Text);

                        }

                        bindingSource1.Position = loc;
```

```
                    }
                    else
                    {
                        col1 =
userDataGridView.Columns[1].HeaderText;
                        //DataView dv0 = new
DataView(this.userTable);
                        //dv0.Sort = col1;


                        int loc = bindingSource1.Find(col1,
this.findTextBox2.Text);
                        //String typoscol =
userDataGridView.Columns[1].ValueType.ToString();
                        //infosearch.Text = typoscol;

                        while (loc == -1)
                        {
                            DataView dv = new
DataView(this.userTable);
                            dv.RowFilter = string.Format(col1 + "
LIKE '{0}*'", this.findTextBox2.Text);
                            if (dv.Count > 0)
                            {

//MessageBox.Show(dv.Count.ToString());
                                loc = bindingSource1.Find(col1,
dv[0][col1]);
                            }
                            else
                            {
                                if (isLastPage == true && loc == -1)
                                {
                                    loc = -1;
                                    MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                    break;
                                }
                                CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                                btnPrevious.Enabled = true;
                            }

                        }
                        bindingSource1.Position = loc;
                    }
                    }

            }
            else if (rcol3.Checked) {

                if (this.findTextBox3.Text.Length > 0)
                {
                    string col1 =
userDataGridView.Columns[2].HeaderText;

//MessageBox.Show(userDataGridView.Columns[0].ValueType.ToString());
                    String typoscol =
userDataGridView.Columns[2].ValueType.ToString();

                    if (typoscol != "System.String")
```

```
                    {
                        infosearch.Text = "Στον τύπο αυτό δεν
υποστηρίζεται μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";

                        int loc = bindingSource1.Find(col1,
this.findTextBox3.Text);


//MessageBox.Show(bindingSource1.Position.ToString());
                        while (loc == -1)
                        {
                            if (isLastPage == true)
                            {
                                loc = -1;
                                MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                break;

                            }
                            CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                            loc = bindingSource1.Find(col1,
this.findTextBox3.Text);
                            //MessageBox.Show(cmbNoOfRecords.Text);

                        }

                        bindingSource1.Position = loc;

                    }
                    else
                    {
                        col1 =
userDataGridView.Columns[2].HeaderText;
                        int loc = bindingSource1.Find(col1,
this.findTextBox3.Text);

                        //String typoscol =
userDataGridView.Columns[1].ValueType.ToString();
                        //infosearch.Text = typoscol;

                        while (loc == -1)
                        {
                            DataView dv = new
DataView(this.userTable);
                            dv.RowFilter = string.Format(col1 + "
LIKE '{0}*'", this.findTextBox3.Text);
                            if (dv.Count > 0)
                            {

//MessageBox.Show(dv.Count.ToString());
                                loc = bindingSource1.Find(col1,
dv[0][col1]);
                            }
                            else
                            {
                                if (isLastPage == true && loc == -1)
                                {
                                    loc = -1;
                                    MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
```

```
                                    break;
                                }
                                CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                                btnPrevious.Enabled = true;
                            }

                        }
                        bindingSource1.Position = loc;
                    }
                }


            }
            else if (rcol4.Checked) {
                if (this.findTextBox4.Text.Length > 0)
                {
                    string col1 =
userDataGridView.Columns[3].HeaderText;

//MessageBox.Show(userDataGridView.Columns[0].ValueType.ToString());
                    String typoscol =
userDataGridView.Columns[3].ValueType.ToString();

                    if (typoscol != "System.String")
                    {
                        infosearch.Text = "Στον τύπο αυτό δεν
υποστηρίζεται μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";

                        int loc = bindingSource1.Find(col1,
this.findTextBox4.Text);


//MessageBox.Show(bindingSource1.Position.ToString());
                        while (loc == -1)
                        {
                            if (isLastPage == true)
                            {
                                loc = -1;
                                MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                break;

                            }
                            CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                            loc = bindingSource1.Find(col1,
this.findTextBox4.Text);
                            //MessageBox.Show(cmbNoOfRecords.Text);

                        }

                        bindingSource1.Position = loc;

                    }
                    else
                    {
                        col1 =
userDataGridView.Columns[3].HeaderText;
                        int loc = bindingSource1.Find(col1,
this.findTextBox4.Text);
```

```
                        //String typoscol =
userDataGridView.Columns[1].ValueType.ToString();
                        //infosearch.Text = typoscol;

                        while (loc == -1)
                        {
                            DataView dv = new
DataView(this.userTable);
                            dv.RowFilter = string.Format(col1 + "
LIKE '{0}*'", this.findTextBox4.Text);
                            if (dv.Count > 0)
                            {

//MessageBox.Show(dv.Count.ToString());
                                loc = bindingSource1.Find(col1,
dv[0][col1]);
                            }
                            else
                            {
                                if (isLastPage == true && loc == -1)
                                {
                                    loc = -1;
                                    MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                    break;
                                }
                                CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                                btnPrevious.Enabled = true;
                            }

                        }
                        bindingSource1.Position = loc;
                    }
                }


            }
            else if (rcol5.Checked)
            {
                if (this.findTextBox3.Text.Length > 0)
                {
                    string col1 =
userDataGridView.Columns[4].HeaderText;

//MessageBox.Show(userDataGridView.Columns[0].ValueType.ToString());
                    String typoscol =
userDataGridView.Columns[4].ValueType.ToString();

                    if (typoscol != "System.String")
                    {
                        infosearch.Text = "Στον τύπο αυτό δεν
υποστηρίζεται μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";

                        int loc = bindingSource1.Find(col1,
this.findTextBox5.Text);


//MessageBox.Show(bindingSource1.Position.ToString());
                        while (loc == -1)
                        {
```

```
                            if (isLastPage == true)
                            {
                                loc = -1;
                                MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                break;

                            }
                            CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                            loc = bindingSource1.Find(col1,
this.findTextBox5.Text);
                            //MessageBox.Show(cmbNoOfRecords.Text);

                        }

                        bindingSource1.Position = loc;

                    }
                    else
                    {
                        col1 =
userDataGridView.Columns[4].HeaderText;
                        int loc = bindingSource1.Find(col1,
this.findTextBox5.Text);
                        //String typoscol =
userDataGridView.Columns[1].ValueType.ToString();
                        //infosearch.Text = typoscol;

                        while (loc == -1)
                        {
                            DataView dv = new
DataView(this.userTable);
                            dv.RowFilter = string.Format(col1 + "
LIKE '{0}*'", this.findTextBox5.Text);
                            if (dv.Count > 0)
                            {

//MessageBox.Show(dv.Count.ToString());
                                loc = bindingSource1.Find(col1,
dv[0][col1]);
                            }
                            else
                            {
                                if (isLastPage == true && loc == -1)
                                {
                                    loc = -1;
                                    MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                    break;
                                }
                                CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                                btnPrevious.Enabled = true;
                            }

                        }
                        bindingSource1.Position = loc;
                    }
                }
```

```
            }
            else if (rcol6.Checked)
            {
                if (this.findTextBox6.Text.Length > 0)
                {
                    string col1 =
userDataGridView.Columns[5].HeaderText;

//MessageBox.Show(userDataGridView.Columns[0].ValueType.ToString());
                    String typoscol =
userDataGridView.Columns[5].ValueType.ToString();

                    if (typoscol != "System.String")
                    {
                        infosearch.Text = "Στον τύπο αυτό δεν
υποστηρίζεται μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";

                        int loc = bindingSource1.Find(col1,
this.findTextBox6.Text);


//MessageBox.Show(bindingSource1.Position.ToString());
                        while (loc == -1)
                        {
                            if (isLastPage == true)
                            {
                                loc = -1;
                                MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                break;

                            }
                            CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                            loc = bindingSource1.Find(col1,
this.findTextBox6.Text);
                            //MessageBox.Show(cmbNoOfRecords.Text);

                        }

                        bindingSource1.Position = loc;

                    }
                    else
                    {
                        col1 =
userDataGridView.Columns[5].HeaderText;
                        int loc = bindingSource1.Find(col1,
this.findTextBox6.Text);
                        //String typoscol =
userDataGridView.Columns[1].ValueType.ToString();
                        //infosearch.Text = typoscol;

                        while (loc == -1)
                        {
                            DataView dv = new
DataView(this.userTable);
                            dv.RowFilter = string.Format(col1 + "
LIKE '{0}*'", this.findTextBox6.Text);
                            if (dv.Count > 0)
```

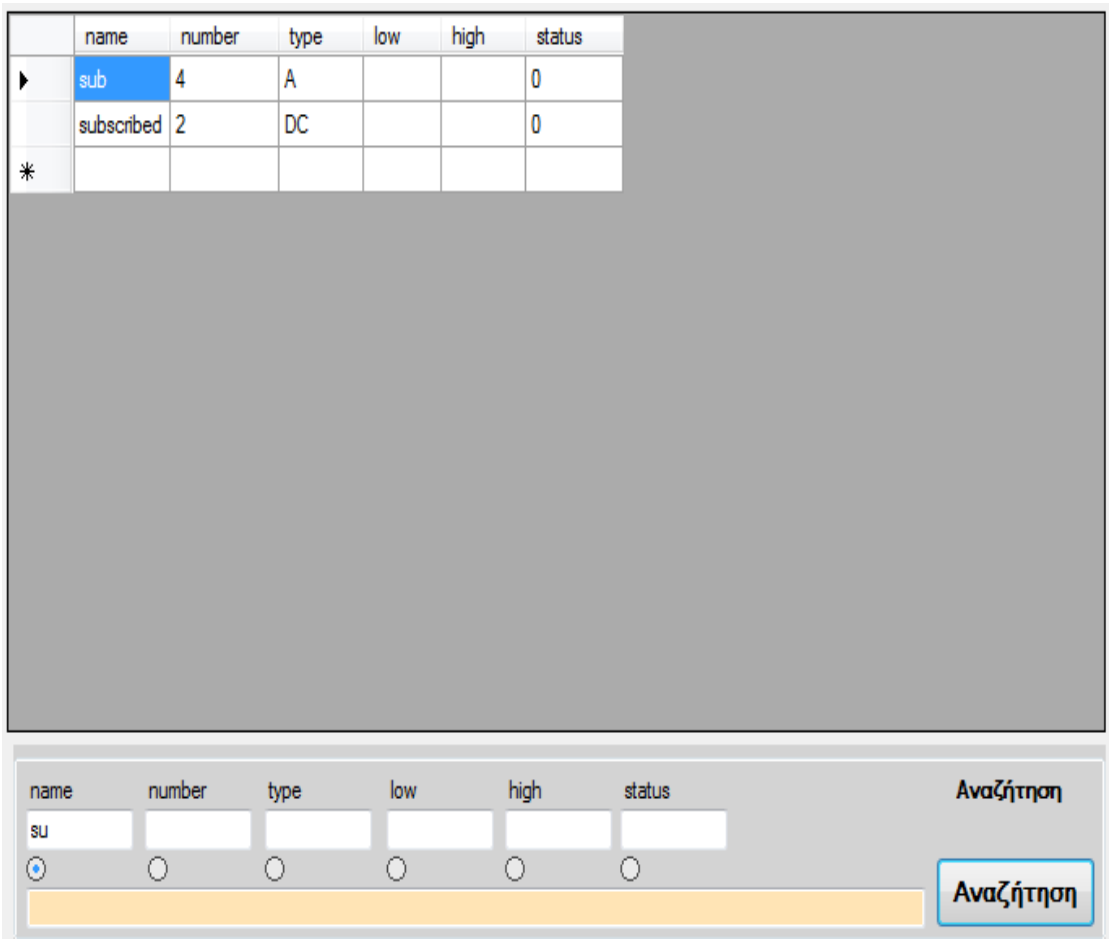
```
                            {

//MessageBox.Show(dv.Count.ToString());
                                loc = bindingSource1.Find(col1,
dv[0][col1]);
                            }
                            else
                            {
                                if (isLastPage == true && loc == -1)
                                {
                                    loc = -1;
                                    MessageBox.Show("Η αναζήτηση δεν
ταιριάζει με καμία εγγραφή");
                                    break;
                                }
                                CreateTempTable(currentIndex,
int.Parse(cmbNoOfRecords.Text));
                                btnPrevious.Enabled = true;
                            }

                        }
                        bindingSource1.Position = loc;
                    }
                }


            }
            }
             catch (Exception ex)
            {
                    MessageBox.Show("Κάποιο λάθος προκλήθηκε !!!
"+ex.ToString());
            }

    }



        private void Form1_Load(object sender, EventArgs e)
        {
            HideSearch();
        }

        void HideSearch() {

            lblCol0.Text = "";
            lblCol1.Text = "";
            lblCol2.Text = "";
            lblCol3.Text = "";
            lblCol4.Text = "";
            lblCol5.Text = "";

            lblCol0.Hide();
            lblCol1.Hide();
            lblCol2.Hide();
            lblCol3.Hide();
            lblCol4.Hide();
            lblCol5.Hide();

            findTextBox1.Hide();
```

```
            findTextBox2.Hide();
            findTextBox3.Hide();
            findTextBox4.Hide();
            findTextBox5.Hide();
            findTextBox6.Hide();

            rcol1.Hide();
            rcol2.Hide();
            rcol3.Hide();
            rcol4.Hide();
            rcol5.Hide();
            rcol6.Hide();

        }



        private void findTextBox1_TextChanged(object sender,
EventArgs e)
        {

        }

        private void findTextBox2_TextChanged(object sender,
EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void cmbNoOfRecords_SelectedIndexChanged(object
sender, EventArgs e)
        {

        }

        private void Search1_Click(object sender, EventArgs e)
        {
            try
            {

                CreateTempTable(0,totalRecords);
                ButtLoadDisable();
                infosearch.Text = "";


                if (rcol1.Checked && findTextBox1.Text.Length > 0)
                {
                    FindEggrafi(0, 1);
                }
                else if (rcol2.Checked && findTextBox2.Text.Length >
0)
                {
                    FindEggrafi(1,2);
                }
                else if (rcol3.Checked && findTextBox3.Text.Length >
0)
```

```
                {
                    FindEggrafi(2, 3);
                }
                else if (rcol4.Checked && findTextBox4.Text.Length >
0)
                {
                    FindEggrafi(3, 4);
                }
                else if (rcol5.Checked && findTextBox5.Text.Length >
0)
                {
                    FindEggrafi(4, 5);
                }
                else if (rcol6.Checked && findTextBox6.Text.Length >
0)
                {
                    FindEggrafi(5, 6);
                }
                else {
                    MessageBox.Show("Ξαναδώσε περιγραφή αναζήτησης\n
το πεδίο δεν μπορεί να είναι άδειο");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }

        }
        void FindEggrafi(int kolona, int stxt)
        {
            string col1 =
userDataGridView.Columns[kolona].HeaderText;
            String typoscol =
userDataGridView.Columns[kolona].ValueType.ToString();
            DataView dv = new DataView(this.userTable);
            string findtxt;

            if (stxt == 1)
            {
                findtxt = findTextBox1.Text;
            }
            else if (stxt == 2)
            {
                findtxt = findTextBox2.Text;
            }
            else if (stxt == 3)
            {
                findtxt = findTextBox3.Text;
            }
            else if (stxt == 4)
            {
                findtxt = findTextBox4.Text;
            }
            else if (stxt == 5)
            {
                findtxt = findTextBox5.Text;
            }
            else if (stxt == 6)
            {
                findtxt = findTextBox6.Text;
```

```
            }
            else
            {
                findtxt = "";
            }

            if (typoscol != "System.String")
            {
                infosearch.Text = "Στον τύπο αυτό δεν υποστηρίζεται
μερική αναζήτηση δώστε ακριβώς αυτό ψάχνετε";
                int loc = bindingSource1.Find(col1, findtxt);
                if (loc == -1)
                {
                        loc = -1;
                        MessageBox.Show("Η αναζήτηση δεν ταιριάζει με
καμία εγγραφή");
                }

                bindingSource1.Position = loc;

            }
            else
            {
                dv.RowFilter = string.Format(col1 + " LIKE '{0}*'",
findtxt);
                userDataGridView.DataSource = dv;

                if (dv.Count == 0) {
                    MessageBox.Show("Η αναζήτηση δεν ταιριάζει με
καμία εγγραφή");

                }

            }
        }
```

Παρακάτω δίνεται ένα παράδειγμα αναζήτησης.



4.Επίλογος

4.1 Παρατηρήσεις

- Το πρόγραμμα αν και φτιάχτηκε να δουλεύει και μέσω του Atlantis II E.R.P. της ALTEC ύστερα από τις τελευταίες αναβαθμίσεις του προγράμματος διακόπηκε αυτή η υπηρεσία. Βέβαια το πρόγραμμα είναι έτσι φτιαγμένο που να τρέχει και stand alone.

- Ο κώδικας που χρησιμοποιήθηκε για την αναζήτηση δεν προήλθε ούτε από βοήθεια του MSDN αλλά ούτε και από αναζήτηση στο διαδίκτυο. Επειδή χρησιμοποιήθηκε DataGridView οι αναφορές στην αναζήτηση είναι ελάχιστες και ελλιπείς.

4.2 Πηγές

http://en.wikipedia.org/wiki/Main_Page

http://www.java2s.com/Code/CSharp/CatalogCSharp.htm

http://www.csharp-station.com/Tutorial.aspx

http://www.c-sharpcorner.com/

http://msdn.microsoft.com/en-us/vstudio/hh388566