ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μηχανογράφηση χρηματιστηριακού γραφείου για διαχείριση ομολόγων

Του φοιτητή

Παρχαρίδη Παύλου

Αρ. Μητρώου: 991350

Επιβλέπων καθηγητής

Κωνσταντίνος Γιακουστίδης

Θεσσαλονίκη 2014

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη πτυχιακή εργασία αφορά μια εφαρμογή διαχείρισης ομολόγων, για χρήση σε χρηματιστηριακό γραφείο. Η εφαρμογή διαθέτει λειτουργίες για δημιουργία και συντήρηση πελατολογίου, δημιουργία κωδικών για διάφορες ιστοσελίδες για ασφαλή αποθήκευση και θέαση, και θέαση τιμών ομολόγων σε πραγματικό χρόνο.

Για την υλοποίηση της χρησιμοποιήθηκαν οι τεχνολογίες: C# .NET με χρήση WPF και του προτύπου προγραμματισμού MVVM,για την βάση δεδομένων χρησιμοποιήθηκε ADO.NET Entity Framework και SQL Server 2008 R2.

SUMMARY

This Bachelor thesis is a bonds management application for a finance services agency. The application has utilities for creating and maintaining customer records, creating and saving usernames and passwords for frequently used websites and also viewing live prices for bonds.

Technologies used for the application: C#.NET WPF with MVVM pattern and for the database ADO.NET Entity Framework with SQL Server 2008 R2.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή της πτυχιακής μου εργασίας κ.Κωνσταντίνο Γιακουστίδη για την βοήθεια του καθ' όλη την διάρκεια της εκπόνησης της. Επίσης, τον φίλο Ελευθέριο Αβραμίδη που τα σχόλια του συνέβαλαν στην βελτιστοποίηση της εργασίας μου. Τέλος, την οικογένεια μου για την συνεχή υποστήριξη τους και ανεκτίμητη βοήθεια τους.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1
Εισαγωγή

Τα τελευταία χρόνια παρατηρείται μια ραγδαία αύξηση της ανάπτυξης εφαρμογών για την διευκόλυνση της λειτουργίας μικρομεσαίων γραφείων και γενικά εταιρειών. Κατά τη δεκαετία του 80 ο ηλεκτρονικός υπολογιστής χρησιμοποιούνταν μόνο για βασικές ανάγκες και υπολογισμούς. Αργότερα, στη δεκαετία του 90, με την κυκλοφορία του Microsoft Office και Exchange, ολοένα και περισσότερες εταιρείες άρχισαν να χρησιμοποιούν το Microsoft Word, Outlook και PowerPoint για να διαχειρίζονται τα ηλεκτρονικά μηνύματα, τους πελάτες και τα αρχεία τους.

Στις μέρες μας, οι εταιρείες έχουν συνειδητοποιήσει ότι ακόμα και να συνδυάσουν όλες αυτές τις εφαρμογές του εμπορίου, δε θα μπορέσουν να καλύψουν τις ιδιαίτερες ανάγκες τους. Οι εφαρμογές CRM (Customer Relationship Management – Συστήματα Διαχείρισης Πελατειακών Σχέσεων) κατέχουν μεγάλο μέρος στη συγκεκριμένη αγορά , αλλά και πάλι, δεν μπορούν να ικανοποιήσουν όλες τις λειτουργίες ενός γραφείου. Τις εφαρμογές CRM τις συναντάμε συνήθως σε λογιστικά γραφεία και στο λογιστήριο των επιχειρήσεων.

Τις συγκεκριμένες αυτές ανάγκες των επιχειρήσεων, έρχονται να καλύψουν εταιρείες ανάπτυξης λογισμικού, δημιουργώντας εφαρμογές που μπορεί να ικανοποιούν από μια απλή και βασική λειτουργία, μέχρι κάτι πολύ εξειδικευμένο και πολύπλοκο. Αυτό είναι πολύ σημαντικό για μια εταιρεία, διότι έτσι όλες οι πληροφορίες και τα δεδομένα είναι διαθέσιμα ανά πάσα στιγμή, αυξάνοντας σε μεγάλο βαθμό την παραγωγικότητα της.

Οι εφαρμογές αυτές, μπορεί να είναι εγκατεστημένες τοπικά, σε έναν αλλά και σε περισσότερους ηλεκτρονικούς υπολογιστές, χρησιμοποιώντας κοινή βάση δεδομένων, ή να είναι διαδικτυακές εφαρμογές εγκατεστημένες σε μεγάλους εξυπηρετητές (Servers). Ακόμα, σε πολλές από αυτές υπάρχει η δυνατότητα σύνδεσης με κάποια άλλη εφαρμογή, ώστε να χρησιμοποιήσει κατευθείαν τα δεδομένα της, για παράδειγμα μια εφαρμογή που μπορεί να πάρει και να χρησιμοποιήσει τις αποθηκευμένες επαφές ηλεκτρονικού ταχυδρομείου από το Microsoft Outlook.

Η συγκεκριμένη πτυχιακή εργασία περιλαμβάνει την υλοποίηση μίας τέτοιας εφαρμογής, σχεδιασμένη για τις ανάγκες ενός χρηματιστηριακού γραφείου και διαθέτει τις παρακάτω λειτουργίες:

- Πελατολόγιο, όπου αποθηκεύονται και διαχειρίζονται όλοι οι πελάτες του γραφείου.
- Λειτουργία για την ασφαλή αποθήκευση και διαχείριση προσωπικών κωδικών ηλεκτρονικών ιστοσελίδων.
- Παρακολούθηση τιμών και διαφόρων άλλων στοιχείων ομολόγων, σε πραγματικό χρόνο.

1.1 Δομή της εργασίας

Στο κεφάλαιο 2 αρχικά θα γίνει μια γενική παρουσίαση των τεχνολογιών που χρησιμοποιεί η εφαρμογή οι οποίες είναι το Windows Presentation Foundation (WPF), το πρότυπο Model View ViewModel (MVVM), και το Entity Framework (EF), και στην συνέχεια θα γίνει πιο αναλυτική παρουσίαση πως χρησιμοποιήθηκαν αυτές οι τεχνολογίες στα επιμέρους κομμάτια της εφαρμογής, δηλαδή στο πελατολόγιο, στον διαχειριστή κωδικών και στην εμφάνιση τιμών ομολόγων.

Στο κεφάλαιο 3, θα γίνει παρουσίαση της διεπαφής της εφαρμογής βήμα-βήμα, δηλαδή κάθε μενού της εφαρμογής τι λειτουργίες μας παρέχει.

Στο τέλος, θα αναφέρουμε τα συμπεράσματα μας για την χρήση των εν λόγο τεχνολογιών και πώς βοηθούν στην γρηγορότερη ανάπτυξη λογισμικού και στην καλύτερη συντήρηση του, και θα γίνει και παράθεση κώδικα στο παράρτημα όπου μπορεί κάποιος να δει με λεπτομέρεια τις σχετικές υλοποιήσεις μας.

Κεφάλαιο 2
Παρουσίαση και περιγραφή της εφαρμογής

Η εφαρμογή έχει αναπτυχθεί σε γλώσσα προγραμματισμού C# χρησιμοποιώντας ως σύστημα παρουσίασης το WPF (Windows Presentation Foundation) και είναι σχεδιασμένη με την τεχνική MVVM (Model View ViewModel). Η C# και το WPF περιλαμβάνονται στο Microsoft .NET Framework. Επίσης για την βάση δεδομένων της χρησιμοποιείται ο Microsoft Sql Server 2008 R2 και η σύνδεση της βάσης δεδομένων με την εφαρμογή γίνεται με την βοήθεια του Entity Framework v4. Η συγγραφή της έγινε στο Visual Studio 2012 Express Edition της Microsoft.

Στο υποκεφάλαιο που ακολουθεί γίνεται μια εκτενέστερη περιγραφή όλων αυτών των τεχνολογιών που χρησιμοποιήθηκαν κατά την ανάπτυξη της.

2.1.1 Windows Presentation Foundation (WPF)

Το WPF αποτελεί ένα καινούργιο σύστημα παρουσίασης για την ανάπτυξη εφαρμογών Windows, που προσφέρει μία εκπληκτική οπτική εμπειρία στο χρήστη. Με το WPF μπορείς να δημιουργήσεις μια μεγάλη γκάμα τοπικών και διαδικτυακών εφαρμογών.

Ο πυρήνας του WPF είναι μία μηχανή απεικόνισης βασισμένη στην ανεξάρτητη ανάλυση και στα διανύσματα έτσι ώστε να εκμεταλλεύεται καλύτερα το μοντέρνο γραφικό εξοπλισμό (Hardware). Τα χαρακτηριστικά που περιλαμβάνει είναι η γλώσσα XAML (Extensive Application Markup Language), εργαλεία (Controls), σύνδεση δεδομένων (Data Binding), 2-D και 3-D γραφικά, γραφική απεικόνιση (Animation), τεχνοτροπίες (Styles), έγγραφα, πρότυπα (Templates), κείμενο και τυπογραφεία.

Στο WPF χρησιμοποιείτε η γλώσσα XAML για την κατασκευή της εμφάνισης της εφαρμογής ενώ παράλληλα χρησιμοποιείτε μία γλώσσα προγραμματισμού (Code-Behind) που ανήκει στο .NET Framework (C# ή Visual Basic) και είναι υπεύθυνη για την συμπεριφορά της εφαρμογής. Αυτός ο διαχωρισμός της εμφάνισης από την συμπεριφορά – λειτουργία έχει τα εξής πλεονεκτήματα:

- Η ανάπτυξη είναι πιο αποδοτική διότι οι σχεδιαστές μπορούν να δουλεύουν το κομμάτι της εμφάνισης ενώ παράλληλα οι προγραμματιστές να δημιουργούν το λειτουργικό μέρος της εφαρμογής.
- Μειώνονται τα κόστη ανάπτυξης και συντήρησης διότι ο κώδικας της εμφάνισης δεν είναι ενοποιημένος με τον κώδικα των λειτουργιών της εφαρμογής.
- Μπορούν να χρησιμοποιηθούν διάφορα σχεδιαστικά εργαλεία για την συγγραφή της γλώσσας XAML για εφαρμογές με μεγαλύτερες απαιτήσεις. Για παράδειγμα ένα τέτοιο εργαλείο είναι το Microsoft Expression Blend.
- Οι εφαρμογές WPF μπορούν πολύ εύκολα να διαμορφωθούν έτσι ώστε να χρησιμοποιούνται σε διάφορους τόπους με διαφορετικές γλώσσες(Globalization και Localization).

Η γλώσσα XAML χρησιμοποιείτε για την δημιουργία παραθύρων, περιθώρια κειμένων (Dialog Boxes), σελίδων, εργαλείων χρήστη (User Control) και είναι υπεύθυνη για να τους δίνει μορφή και γραφική απεικόνιση.

Το παράδειγμα που ακολουθεί χρησιμοποιεί XAML για να δημιουργήσει ένα παράθυρο Windows που περιέχει ένα κουμπί:

```
<Window
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   Title="Window with Button"
   Width="250" Height="100">

  <!—Προσθέτει το κουμπί στο παράθυρο -->
  <Button Name="button">Click Me!</Button>

</Window>
```

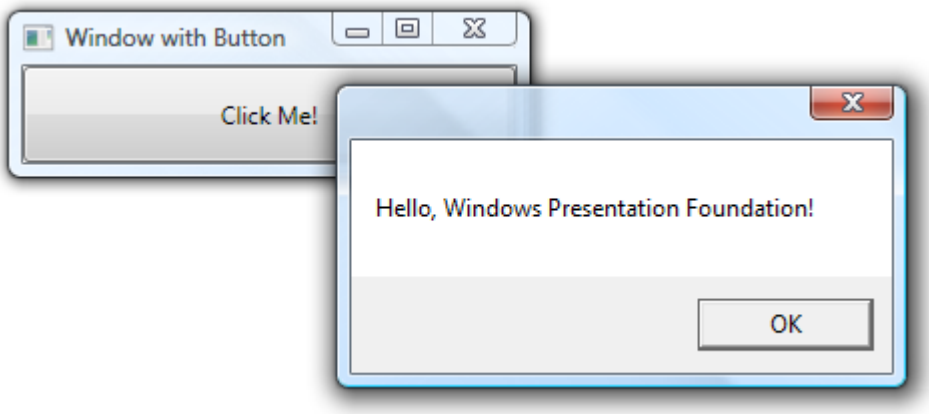
Εικόνα 2.1: Η διεπαφή του χρήστη που ορίστηκε από τον κώδικα XAML

Στη συνέχεια πρέπει να γράψουμε τον κώδικα (Code Behind) που είναι υπεύθυνος για την λειτουργία του συγκεκριμένου παραθύρου, δηλαδή τί θέλουμε να ενεργοποιείται πατώντας το κουμπί “Click Me!”. Ο κώδικας είναι γραμμένος σε γλώσσα C# και είναι ο παρακάτω:

```
using System.Windows; // Window, RoutedEventArgs, MessageBox

namespace SDKSample
{
    public partial class AWindow : Window
    {
        public AWindow()
        {
            // Αρχικοποιεί το στοιχείο (Component), καλώντας τη συνάρτηση αυτή
             // γίνεται η σύνδεση της διεπαφής με την κλάση, ορίζοντας τις ιδιότητες
           //(Properties) και τις μεθόδους (Events) της.
            InitializeComponent();
        }

        void button_Click(object sender, RoutedEventArgs e)
        {
            // Εμφάνιση πλαισίου κειμένου όταν πατάς το κουμπί
            MessageBox.Show("Hello, Windows Presentation Foundation!");
        }
    }
}
```



Εικόνα 2.2: Το αποτέλεσμα του πατήματος του κουμπιού

2.1.2 MVVM

Το πρότυπο Model View Viewmodel (Μοντέλο Όψη (Όψη Μοντέλο) - MVVM) είναι ένα αρχιτεκτονικό πρότυπο που χρησιμοποιείται στην μηχανική λογισμικού το οποίο αρχικοποιήθηκε από την Microsoft ως εξειδίκευση του σχεδιαστικού προτύπου Presentation Model του Μάρτιν Φάουλερ. Το MVVM είναι κατά ένα μεγάλο μέρος βασισμένο στο πρότυπο Model-View-Controller (MVC) και είναι μια εξειδικευμένη υλοποίηση που στοχεύει σε πλατφόρμες ανάπτυξης UI (User Interface – Διεπαφές Χρήστη) οι οποίες υποστηρίζουν προγραμματισμό με συμβάντα (Event-driven programming),πιο συγκεκριμένα δηλαδή σε Windows Presentation Foundation (WPF) και Silverlight της πλατφόρμας .NET χρησιμοποιώντας XAML και .NET γλώσσες προγραμματισμού.

Το MVVM κάνει σαφή διαχωρισμό μεταξύ της ανάπτυξης της γραφικής διεπαφής χρήστη (GUI) και της ανάπτυξης της επιχειρηματικής λογικης (business logic ή back end logic) που ονομάζεται μοντέλο (ή αλλιώς μοντέλο δεδομένων (data model) για να το διαχωρίζουμε από το μοντέλο όψης (view model)). Το view model του MVVM είναι ένας μετατροπέας αξιών (value converter),αυτό σημαίνει ότι το view model είναι υπεύθυνο για την κοινοποίηση των αντικειμένων δεδομένων του μοντέλου με τέτοιο τρόπο ώστε να χρησιμοποιούνται πιο εύκολα. Από αυτή την άποψη, το view model είναι πιο κοντά στην έννοια του μοντέλου παρά στην έννοια της όψης (view),και χειρίζεται σχεδόν όλη την λογική των views.Το view model επίσης υλοποιεί ένα πρότυπο διαμεσολαβητή (mediator pattern) ο οποίος προσφέρει πρόσβαση στην λογική του προγράμματος μέσω ενός συνόλου περιπτώσεων χρήσης (use cases) που υποστηρίζονται από το view.

Το MVVM σχεδιάστηκε ώστε να χρησιμοποιεί τις λειτουργίες σύνδεσης δεδομένων (data binding) του WPF, έτσι γίνεται καλύτερος διαχωρισμός της ανάπτυξης των όψεων (views) από το υπόλοιπο πρότυπο αφαιρώντας οποιοδήποτε κώδικα από τις όψεις. Με αυτόν τον τρόπο, οι προγραμματιστές διεπαφών χρήστη (UX developers) δεν γράφουν κώδικα για την διεπαφή χρήστη (GUI), αλλά χρησιμοποιούν την γλώσσα σήμανσης του πλαισίου (XAML) και δημιουργούν συντομεύσεις (binding) προς το view model,το οποίο είναι γραμμένο και συντηρείται από προγραμματιστές εφαρμογών. Αυτός ο διαχωρισμός ρόλων επιτρέπει στους σχεδιαστές διεπαφών χρήστη να αφοσιωθούν στις ανάγκες των

διεπαφών και όχι στον προγραμματισμό της λογικής της εφαρμογής, και έτσι τα διάφορα κομμάτια της εφαρμογής μπορούν να δουλεύονται ταυτόχρονα από διάφορα μέλη με αποτέλεσμα την αύξηση της παραγωγικότητας. Ακόμα και αν μια εφαρμογή αναπτύσσεται από έναν μόνο προγραμματιστή ο διαχωρισμός αυτός των όψεων (views) από το μοντέλο (model) είναι πιο παραγωγικό γιατί επιτρέπει αργότερα να γίνουν αλλαγές στην όψη (view) χωρίς να επηρεάζεται ο κώδικας πίσω από αυτές.

Πιο συγκεκριμένα, τα στοιχεία που ορίζουν το πρότυπο MVVM είναι τα εξής:

- **Μοντέλο (Model):** Το μοντέλο περιέχει όλη την λογική του προγράμματος είτε αυτή είναι ορισμένη με αντικείμενα (object-oriented approach) είτε με καθαρά δεδομένα (data-centric approach).
- **Όψη (View):** Όψεις είναι όλα τα γραφικά στοιχεία της εφαρμογής μας (τα πλαίσια, τα κουμπιά κτλ.)
- **Όψη Μοντέλο (View Model):** Το View Model υλοποιεί την λογική του μοντέλου (Model) έτσι ώστε να μπορεί να χρησιμοποιηθεί από τις όψεις (views). Λειτουργεί σαν γέφυρα μεταξύ μοντέλου και όψεων, μετατρέπει τις πληροφορίες του μοντέλου σε πληροφορίες για τις όψεις και επίσης μεταβιβάζει εντολές (commands) από τις όψεις στο μοντέλο.

Περαιτέρω, για την διασύνδεση του Μοντέλου με τις Όψεις και το View Model, χρησιμοποιούνται οι εξής δύο τεχνολογίες που μας προσφέρει το WPF,το Data Binding και το Commanding:

- **Data Binding:** Με το data binding μπορούμε να συνδέσουμε στοιχεία των όψεων απευθείας με πηγές δεδομένων (data sources), πχ. Να έχουμε ένα textbox να παίρνει τιμή απευθείας από μια πηγή του μοντέλου, και αυτό το ορίζουμε απλά στον κώδικα XAML του γραφικού στοιχείου χωρίς να βάλουμε κώδικα πίσω από την όψη.

Παράδειγμα κώδικα data binding:

Σε αυτό το παράδειγμα ορίζεται ότι το textbox θα παίρνει δεδομένα από μία συγκεκριμένη διεύθυνση στο διαδίκτυο:

```
<TextBox Grid.Row="1" Grid.Column="1" Text="{Binding
Path=Feed.Link.AbsoluteUri, Mode=OneWay}" />
```

- **Commanding:** Με τις εντολές που παρέχει το WPF μπορούμε να ελέγχουμε όταν αλλάζει κάτι σε μια όψη και να μεταφέρουμε αυτό το συμβάν αυτόματα στο μοντέλο.

Παράδειγμα κώδικα commanding:

Δήλωση της command στο View Model:

```
public ICommand OpenFeedCommand
{
    get;
    private set;
}
```

Δήλωσή της στην όψη, η οποία είναι ένα κουμπί στην περίπτωση μας:

```
<Button
   Grid.Row="1" Grid.Column="1" Height="23"
   HorizontalAlignment="Right" Margin="0,0,0,0"
   VerticalAlignment="Top" Width="60"
   Command="{Binding OpenFeedCommand}">
Open
</Button>
```

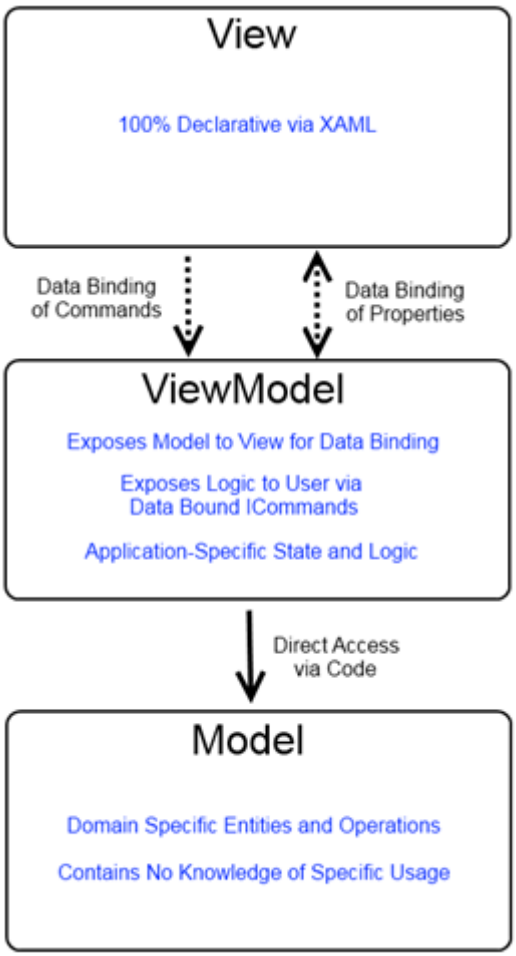
- **Templating**: Επίσης, το WPF μας προσφέρει και το templating, το οποίο έχει να κάνει εξολοκλήρου με τις όψεις και έχει σκοπό και αυτό να διαχωρίζει τον κώδικα από την όψη. Έτσι, ακολουθώντας αυτή την νοοτροπία το templating μας επιτρέπει να αλλάζουμε την εμφάνιση μιας όψης, ενός κουμπιού πχ., χωρίς να χρειάζεται να δημιουργήσουμε από την αρχή ένα custom control όπως θα κάναμε με windows forms στο

παρελθόν αλλά με το να δημιουργήσουμε ένα template για το συγκεκριμένο control, το κουμπί δηλαδή.

Παράδειγμα κώδικα templating στην όψη:

```
<DataTemplate DataType="{x:Type model:FeedItem}">
    <TextBlock Text="{Binding Path=Title}" />
</DataTemplate>
```

Το παρακάτω διάγραμμα εξηγεί τις διασυνδέσεις μεταξύ Model-View-View Model:



Εικόνα 2.3: Διάγραμμα MVVM

Ανακεφαλαιώνοντας:

Το Μοντέλο (Model):

- Δεν γνωρίζει οτιδήποτε άλλο εκτός από τον εαυτό του.
- Μπορεί να χρησιμοποιηθεί από οποιαδήποτε μορφή του προγράμματος και δεν περιορίζεται από συγκεκριμένα πλαίσια (frameworks) ή τεχνολογίες.

Η Όψη (View):

- Η όψη καθορίζεται πλήρως σε XAML.
- Η όψη το μόνο που χρειάζεται να γνωρίζει είναι με το τι πρόκειται να συνδεθεί, ονομαστικά. Δεν χρειάζεται να ξέρει τι θα γίνει αν αλλάξει κάποια ιδιότητα (property) ή αν εκτελεστεί κάποια εντολή (command).
- Η κατάσταση (state) μιας όψης βασίζεται εξολοκλήρου στα δεδομένα με τα οποία είναι συνδεμένη (data binding).

Η Όψη Μοντέλο (View Model):

- Το View Model δεν γνωρίζει τίποτα για την όψη (View).
- Το View Model επικοινωνεί άμεσα με το μοντέλο με σκοπό να το κάνει διαθέσιμο για σύνδεση δεδομένων (data binding).
- Το View Model διαχειρίζεται όλες τις πληροφορίες που έχουν να κάνουν με την εφαρμογή.

2.1.3 Entity Framework (EF)

Το Entity Framework είναι ένα εργαλείο που συνδέει τη βάση δεδομένων με τα αντικείμενα στον κώδικα που είναι υπεύθυνα για τη διαχείρισή της. Επιτρέπει στους .NET προγραμματιστές να δημιουργούν τη σύνδεση αυτή με την βάση, με έναν αυτοματοποιημένο τρόπο όπου δεν απαιτείται η συγγραφή του μεγαλύτερου μέρους του κώδικα.

Το EF έχει δημιουργηθεί από την Entity Framework Ομάδα, οι οποία είναι μέρος του Microsoft Open Tech Hub, σε συνεργασία με μια ομάδα ελεύθερων προγραμματιστών.

Μπορούμε να χρησιμοποιήσουμε το EF με δύο διαφορετικούς τρόπους. Ο πρώτος είναι να γράψουμε με κώδικα στην εφαρμογή τις κλάσεις που θα αποτελούν τους πίνακες της βάσης δεδομένων και τις μεταξύ τους σχέσεις, το λεγόμενο μοντέλο (Model). Στη συνέχεια, με μια απλή διαδικασία, που δεν περιλαμβάνει συγγραφή κώδικα ή σχεδιασμό της βάσης δεδομένων, θα δημιουργήσουμε τη βάση δεδομένων από τις κλάσεις αυτές αυτόματα, με την βοήθεια του EF.

Ο δεύτερος τρόπος είναι να σχεδιάσουμε απευθείας τη βάση δεδομένων, με τους πίνακες που περιέχει και τις μεταξύ τους σχέσεις, με ένα εργαλείο όπως για παράδειγμα το SQL Server Management Studio (Εργαλείο διαχείρισης του Microsoft SQL Server). Στη συνέχεια αφού σχεδιάσουμε τη βάση, μέσα από την εφαρμογή μας με απλά πάλι βήματα, χωρίς να χρειάζεται επί το πλείστον συγγραφή κώδικα, δημιουργούμε αυτόματα με την βοήθεια του EF τις κλάσεις που είναι υπεύθυνες για την διαχείριση των δεδομένων της βάσης, δηλαδή το μοντέλο.

Παρατηρούμαι ότι όποιον από τους δύο τρόπους και αν επιλέξουμε για την εφαρμογή μας, γλιτώνουμε τη μισή διαδικασία από το να μην χρησιμοποιούσαμε το EF. Στην πρώτη γλιτώνουμε το κομμάτι του σχεδιασμού της βάσης και στην δεύτερη το κομμάτι της συγγραφής του κώδικα. Η επιλογή ποιου τρόπου θα ακολουθήσουμε καθορίζετε από τις ανάγκες της εφαρμογής και από την προσωπική μας θέληση.

Στην εφαρμογή της πτυχιακής χρησιμοποιήθηκε ο δεύτερος τρόπος και θα γίνει λεπτομερής περιγραφή στο υποκεφάλαιο που ακολουθεί.

2.2 Περιγραφή της εφαρμογής

Εδώ θα μελετηθούν εκτενέστερα όλες οι λειτουργίες που περιλαμβάνονται στην εφαρμογή. Οι οποίες είναι:

1. Πελατολόγιο, όπου αποθηκεύονται και διαχειρίζονται όλοι οι πελάτες του γραφείου.
2. Λειτουργία για την ασφαλή αποθήκευση και διαχείριση προσωπικών κωδικών ηλεκτρονικών ιστοσελίδων.
3. Παρακολούθηση τιμών και διαφόρων άλλων στοιχείων ομολόγων, σε πραγματικό χρόνο.

2.2.1 Πελατολόγιο

Το πελατολόγιο περιέχει τρεις βασικές λειτουργίες.

- Την εμφάνιση των πελατών.
- Τη διαγραφή πελάτη.
- Την προσθήκη πελάτη.

Για να γίνει πιο ξεκάθαρος ο τρόπος υλοποίησης αυτών των λειτουργιών, παρατίθεται ακριβώς πιο κάτω το μοντέλο της κλάσης του πελάτη, στην οποία υπάρχουν και όλα τα αντίστοιχα πεδία του πίνακα του πελάτη της βάσης δεδομένων, που έχει δημιουργηθεί με το Entity Framework.

```
public partial class Customer
  {
    public int CustomerId { get; set; }
    public string LastName { get; set; }
    public string FirstName { get; set; }
    public string Email { get; set; }
    public string LocalPhoneNumb { get; set; }
    public string CellPhoneNumb { get; set; }
    public string Address { get; set; }
```

```
        public string City { get; set; }
        public string ZipCode { get; set; }
        public string Country { get; set; }
    }
```

Αρχίζοντας, θα δούμε τη συνάρτηση που υλοποιεί την εμφάνιση των πελατών. Παρακάτω βρίσκεται ο κώδικας που την αποτελεί:

```
    public void ShowCustomersList()
    {
            using (var dataContext = new Entities())                    (1)
            {
            var list = dataContext.Customers.ToList();                  (2)
            Customers = new ObservableCollection<Customer>(list);       (3)
            }
        OnPropertyChanged("Customers");                                 (4)
    }
```

Στη γραμμή κώδικα με την αρίθμηση (1) προετοιμάζεται η σύνδεση του μοντέλου της εφαρμογής με την βάση δεδομένων, έτσι ώστε στη συνέχεια να προσθέσουμε, να αλλάξουμε και να διαγράψουμε οποιαδήποτε εγγραφή επιθυμούμε.

Έτσι στην γραμμή (2) δημιουργούμε μία λίστα που περιέχει όλες τις εγγραφές των πελατών από τη βάση. Στη συνέχεια για να μπορέσουμε να τις εμφανίσουμε χρειάζεται να μετατραπεί αυτή τη λίστα σε αντικείμενα του τύπου κλάσης Customer. Τέλος στο γραμμή (4) υπάρχει μία μέθοδος (Event) έτσι ώστε μόλις υπάρξει μια αλλαγή στα στοιχεία στη βάση να μας τα εμφανίσει.

Η σύνδεση με την βάση δεδομένων γίνεται κατά αυτόν τον τρόπο όταν χρησιμοποιείτε το Entity Framework και στην ουσία αντικαθιστά τον κλασσικό, όπου χρειάζεται να δημιουργήσεις ένα Sql Connection String και στη συνέχεια να κάνεις τα Sql Queries προς τη βάση.

Για την καλύτερη κατανόηση του τρόπου επικοινωνίας και διαχείρισης της βάσης δεδομένων από το πρόγραμμα, θα δούμε τώρα τη συνάρτηση της διαγραφής, η οποία είναι:

```
public void RemoveCustomer()
    {
        using (var dataContext = new Entities())                                    (1)
        {
            int ID = SelectedCustomer.CustomerId;                                   (2)
            Customer cust =dataContext.Customers.First(i =>i.CustomerId ==ID);(3)
            dataContext.Customers.Remove(cust);                                     (4)
            dataContext.SaveChanges();                                              (5)
        } ShowCustomersList();                                                      (6)
    }
```

Αντίστοιχα και στην συνάρτηση της διαγραφής, στη γραμμή (1) γίνεται η σύνδεση με τη βάση δεδομένων. Στη (2) αποθηκεύεται σε μία ακέραια μεταβλητή με το όνομα “ID”, ο μοναδικός αριθμός που αντιπροσωπεύει τον πελάτη που έχει επιλεχθεί από την διεπαφή της εφαρμογής, την οποία θα μελετήσουμε ακριβώς μετά από την ανάλυση της διαγραφής. Οπότε στη γραμμή (3) γίνεται μία αναζήτηση στη βάση δεδομένων μας και συγκεκριμένα στον πίνακα των πελατών, του πελάτη με τον μοναδικό αριθμό που αποθηκεύτηκε νωρίτερα. Αυτός ο πελάτης σώζεται προσωρινά στη μεταβλητή με το όνομα “cust”, τύπου κλάσης Customer. Στη (4) με την βοήθεια της δεσμευμένης συνάρτησης Remove του Entity Framework, διαγράφεται ο συγκεκριμένος πελάτης από την βάση. Η γραμμή (5) αποτελεί μία απαραίτητη και βασική ενέργεια , διότι χωρίς την εντολή SaveChanges() τίποτα από τα προηγούμενα δε θα αποθηκευτούν στη βάση και θα παραμείνουν μόνο προσωρινά, έως ότου τελειώσει η εκτέλεση της συγκεκριμένης συνάρτησης. Τέλος στην έκτη γραμμή, ξανακαλούμε τη συνάρτηση ShowCustomersList() όπου θα μας εμφανίσει τους πελάτες, εκτός από αυτόν που διαγράψαμε.

Η υλοποίηση της προσθήκης πελάτη είναι πανομοιότυπη με την διαγραφή, οπότε περνάμε κατευθείαν στην παρουσίαση της διεπαφής του πελατολογίου όπου και θα αναλύσουμε τον τρόπο επικοινωνίας της με τον κώδικα που βρίσκεται από πίσω.

Κάθε μία από τις τρεις κύριες λειτουργίες της εφαρμογής μας, δηλαδή του πελατολογίου, των κωδικών και των πληροφοριών των ομολόγων, αποτελείται από ένα View και ένα ViewModel, σύμφωνα με το σχεδιαστικό πρότυπο MVVM. Οι συναρτήσεις που αναλύθηκαν ακριβώς πιο πάνω, έχουν αναπτυχθεί και αποτελούν

μέρος του ViewModel του πελατολογίου. Το πελατολόγιο διαθέτει και ένα View, το οποίο είναι υπεύθυνο για την γραφική αναπαράστασή του. Η σύνδεση μεταξύ του View και του ViewModel γίνεται με την βοήθεια των Data Bindings και των ICommands. Παρακάτω παρατίθεται μέρος από το View του πελατολογίου, όπου θα μελετήσουμε πως ακριβώς καλούνται οι συναρτήσεις της Εμφάνισης και της Διαγραφής, που μελετήθηκαν νωρίτερα. Ο κώδικας είναι γραμμένος σε γλώσσα XAML και είναι ο παρακάτω:

```
<DataGrid Name="CustDatagrid"                                    (1)
Height="Auto"  Margin="5" AutoGenerateColumns="False"            (2)

ItemsSource="{Binding Customers, Mode=TwoWay,                    (3)
UpdateSourceTrigger=PropertyChanged                              (4)

 VerticalAlignment="Top"  HorizontalAlignment="Center"  Width="Auto"          (5)
IsReadOnly="True" SelectionMode="Single"                         (6)

 SelectedItem="{Binding SelectedCustomer}">                      (7)
```

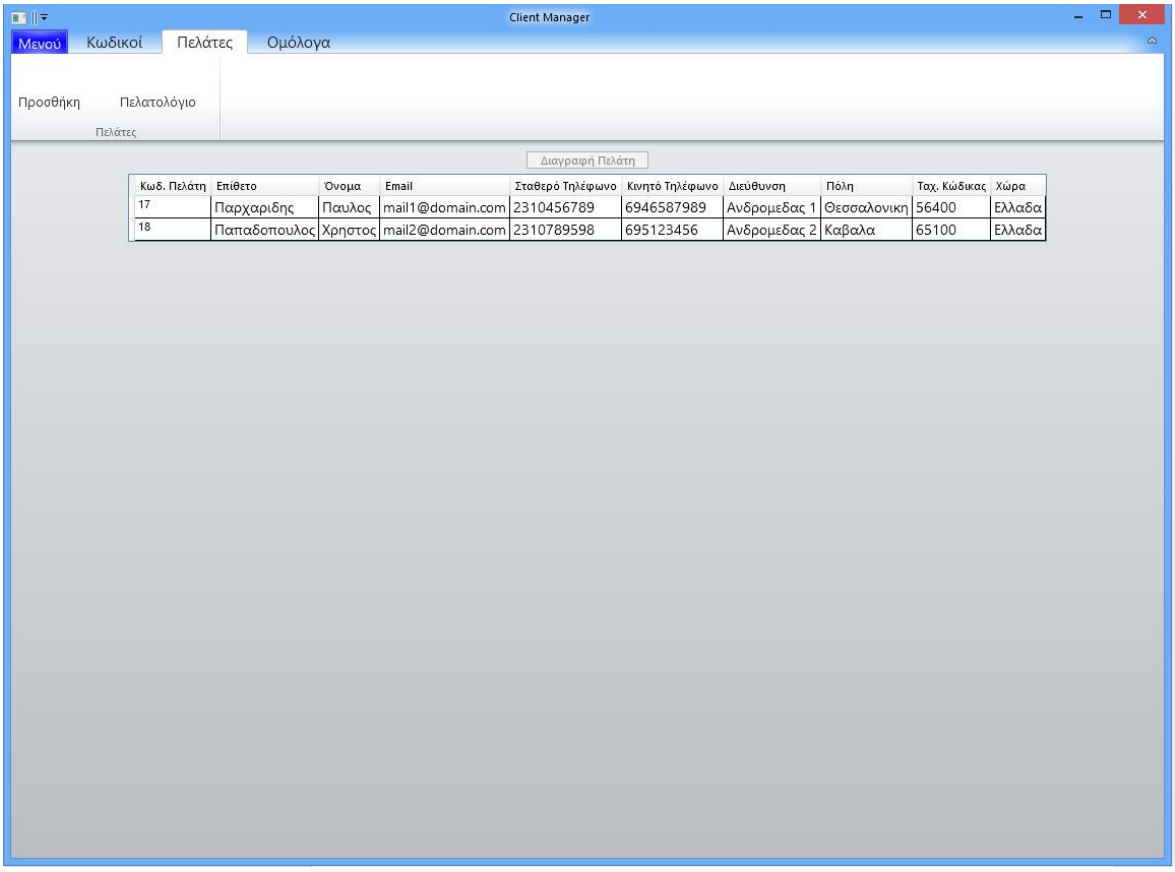
Το συγκεκριμένο κομμάτι κώδικα αποτελεί μόνο την αρχή του Datagrid Control. Σε κάθε Control (εργαλείο) του WPF ορίζεται η αρχή του και το τέλος του. Στη συγκεκριμένη περίπτωση η αρχή του είναι το <DataGrid *Διάφοροι παράμετροι και ιδιότητες*> μετά ακολουθούν άλλα Controls που περιέχονται σε αυτό και στο τέλος υπάρχει το κλείσιμο του, όπου δηλώνεται με αυτόν τον τρόπο </DataGrid>. Το συγκεκριμένο Control δημιουργεί ένα πλέγμα ώστε μέσα σε αυτό να μπορείς να εμφανίσεις και να επεξεργαστείς διάφορα δεδομένα.

Όταν δημιουργούμε ένα Control πρέπει να του δώσουμε κάποια χαρακτηριστικά, όπως το μέγεθος του, τη θέση του, το όνομα του και πολλά άλλα. Στις γραμμές (1), (2) υπάρχει η ονομασία του (Name) , πόσο ύψος θα καταλαμβάνει στην οθόνη (Height), τη θέση του (Margin) και αν θα δημιουργούνται οι στήλες αυτόματα (AutoGenerateColumns). Στις (3) και (4) βλέπουμε πως το Control συνδέεται με τον πίσω κώδικα, τα λεγόμενα Data Bindings που αναφέρθηκαν πριν

στη θεωρία. Στην ιδιότητα ItemsSource του λέμε ότι θα συνδέεται με τα δεδομένα που περιέχονται στην κλάση Customers και όποια αλλαγή γίνει πίσω στην βάση να εμφανίζεται στο Control μας με την βοήθεια του UpdateSourceTrigger. Οπότε με αυτόν τον τρόπο υπάρχει άμεση επικοινωνία μεταξύ του Control, διεπαφής, και του κώδικα που τρέχει στο παρασκήνιο.

Στις γραμμές (5) και (6) βλέπουμε κάποιες ακόμα ιδιότητες που διαμορφώνουν την γραφική απεικόνιση του. Στη γραμμή (7) ορίζουμε την λειτουργία της επιλογής κάποιας εγγραφής από το Control. Αν θυμηθούμε και το παράδειγμα τις διαγραφής πελάτη θα διαπιστώσουμε ότι με αυτόν τον τρόπο πήραμε το ID του πελάτη ώστε να τον διαγράψουμε από την βάση.

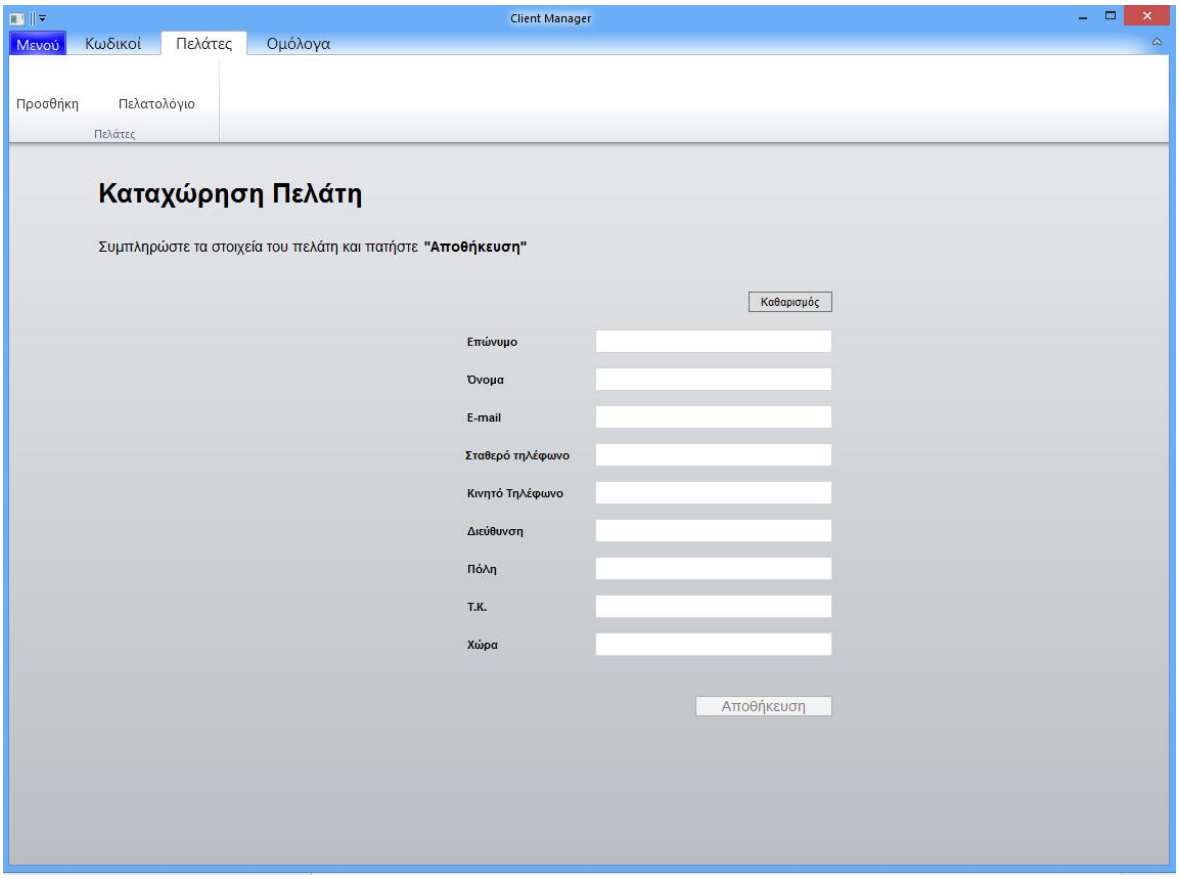
Παρακάτω παρουσιάζεται το γραφικό αποτέλεσμα της εμφάνισης του πελατολογίου.



Εικόνα 2.4: Το πελατολόγιο

Με την επιλογή κάποιου πελάτη, ενεργοποιείτε το κουμπί Διαγραφή Πελάτη έτσι ώστε αν το πατήσεις να τρέξει ο κώδικας που παρουσιάστηκε νωρίτερα.

Η προσθήκη πελάτη γίνεται με την βοήθεια της φόρμας που ακολουθεί.



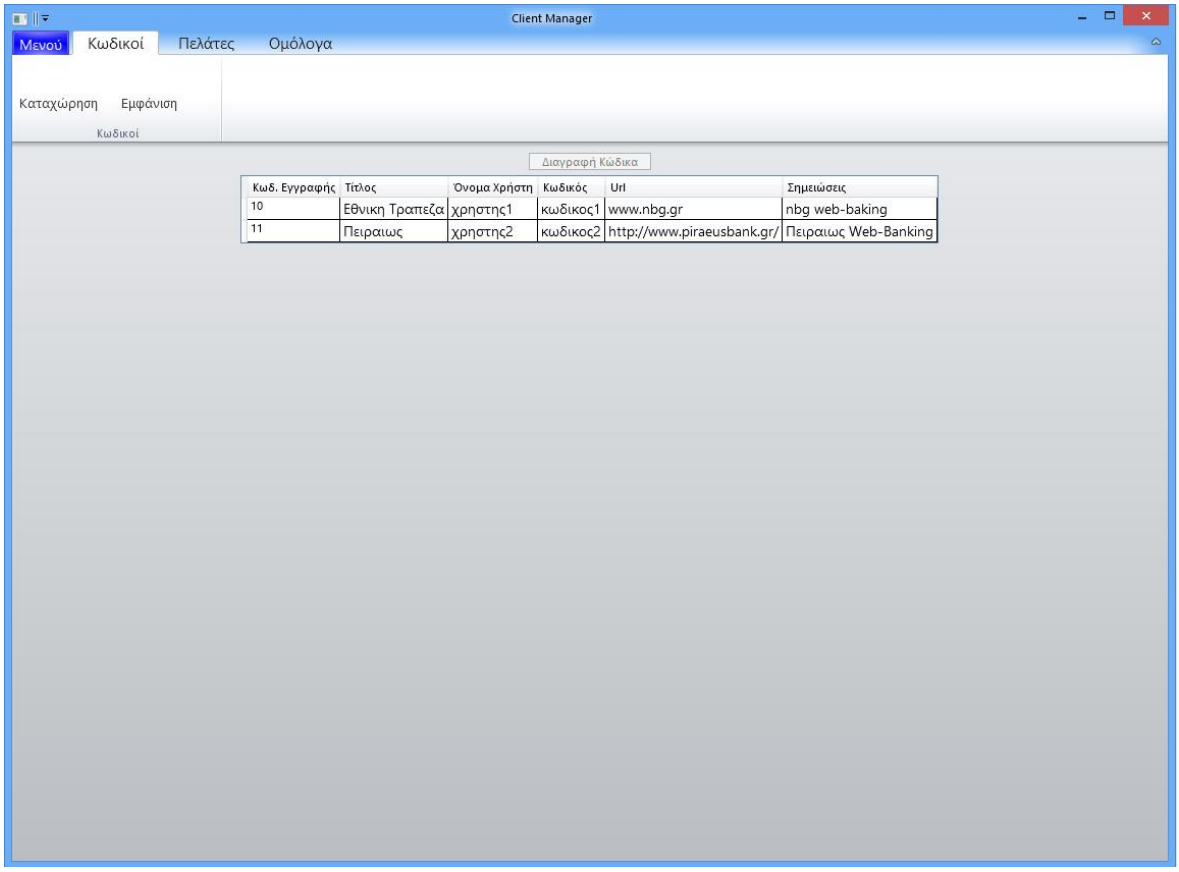
Εικόνα 2.5: Η προθήκη πελάτη

Μόλις γίνει η καταχώρηση των στοιχείων του πελάτη ενεργοποιείτε το κουμπί Αποθήκευση, όπου με το πάτημα του, ότι υπάρχει στα πεδία περνάει στις συγκεκριμένες στήλες του πίνακα Customers. Το κουμπί καθαρισμός διαγράφει όλα τα δεδομένα από την φόρμα, δείχνοντας μας την εικόνα που βλέπουμε τώρα.

2.2.2 Διαχείριση προσωπικών κωδικών

Ο σκοπός της διαχείρισης των προσωπικών κωδικών, είναι να υπάρχει ένα μέρος όπου θα μπορείς να διατηρείς προσωπικά στοιχεία διαφόρων ιστοσελίδων και όχι μόνο. Όταν κάνεις μία εγγραφή σε μια ιστοσελίδα χρειάζεσαι ένα Όνομα Χρήστη (User Name) και ένα Κωδικό (Password). Οπότε με αυτόν τον τρόπο μπορείς να αποθηκεύεις για πολλές ιστοσελίδες τα συγκεκριμένα στοιχεία ώστε να μην υπάρχει ο φόβος της απώλειας τους. Όταν καταχωρείς στοιχεία για μία ιστοσελίδα σου δίνει τη δυνατότητα να βάλεις ένα τίτλο στην εγγραφή, το όνομα χρήστη, τον κωδικό, τη διεύθυνση όπου βρίσκεται η συγκεκριμένη ιστοσελίδα και κάποια σημείωση – παρατήρηση.

Στην εικόνα που ακολουθεί βλέπουμε πως αυτά τα στοιχεία εμφανίζονται.



Εικόνα 2.6: Εμφάνιση κωδικών

Οι λειτουργίες που περιλαμβάνονται στη διαχείριση των προσωπικών κωδικών είναι οι εξής:

- Εμφάνιση
- Καταχώρηση
- Διαγραφή

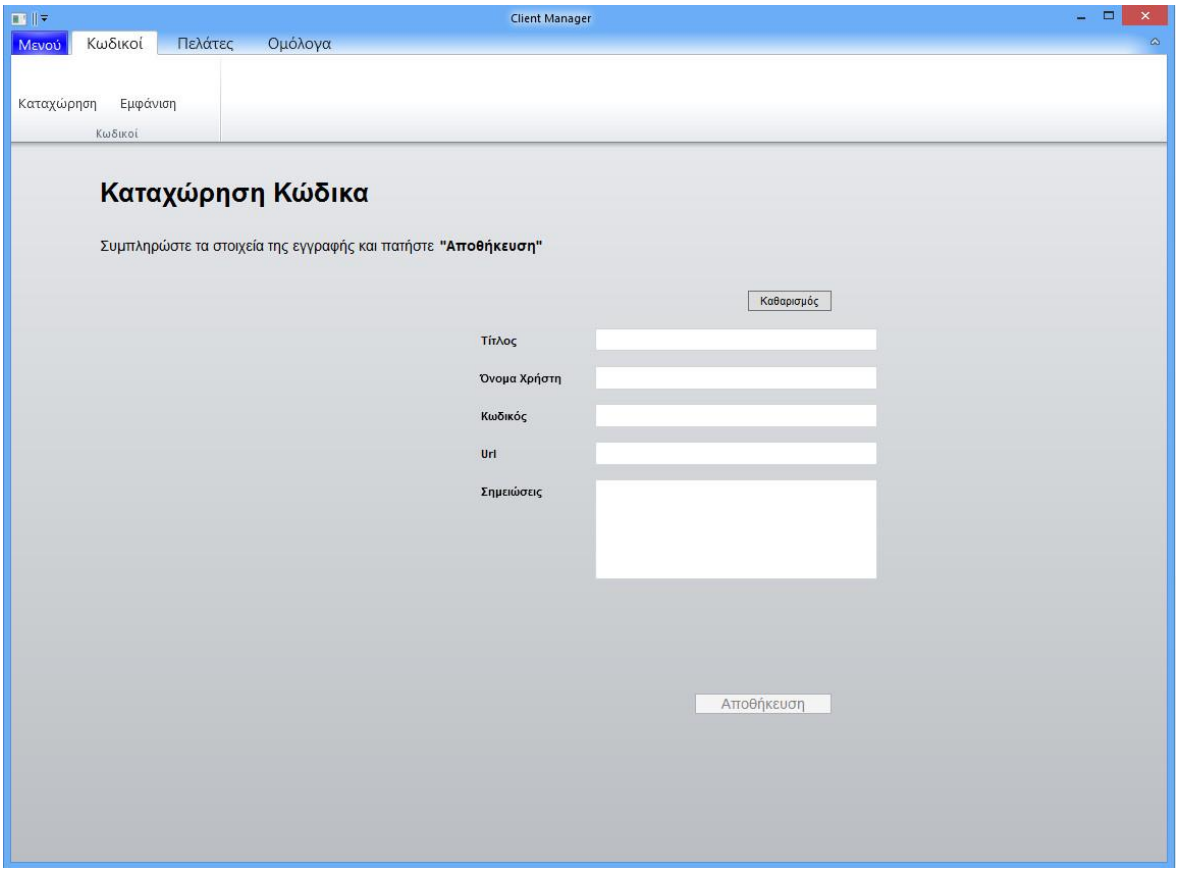
Το μοντέλο που αποτελεί τη συγκεκριμένη λειτουργία της εφαρμογής είναι το εξής:

```
public partial class Code
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string UserName { get; set; }
    public string Password { get; set; }
    public string Url { get; set; }
    public string Notes { get; set; }
}
```

Όπως στο πελατολόγιο έτσι και εδώ υπάρχει ένα View και ένα ViewModel. Στο ViewModel της λειτουργίας αυτής υπάρχουν οι υλοποιήσεις των τριών συναρτήσεων που είναι υπεύθυνες για την Εμφάνιση, Καταχώρηση και Διαγραφή πελάτη.

Η Εμφάνιση υλοποιείται ακριβώς με τον ίδιο τρόπο όπως στο πελατολόγιο, απλώς αντί να εμφανίσει ότι περιλαμβάνει ο πίνακας Customers, εμφανίζει όλες τις εγγραφές του πίνακα Codes. Το αποτέλεσμα της εμφάνισης κωδικών το είδαμε στην προηγούμενη εικόνα.

Η καταχώρηση γίνεται μέσω του View της διαχείρισης των προσωπικών κωδικών, όπου το βλέπουμε παρακάτω.



Εικόνα 2.7: Καταχώρηση κωδικών

Επομένως, μόλις καταχωρήσουμε τα στοιχεία στα πεδία, ενεργοποιείτε το κουμπί της αποθήκευσης. Με το πάτημα του καλείται ο κώδικας της συνάρτησης SaveCode() και ότι υπάρχει στα πεδία αποθηκεύεται στον πίνακα Codes. Η συνάρτηση SaveCode() περιλαμβάνει μόνο τον παρακάτω κώδικα:

```
public void SaveCode()
    {
        using (var dataContext = new Entities())
        {
            dataContext.Codes.Add(MCode);
            dataContext.SaveChanges();
        }

        CancelSaveCode();
    }
```

Εδώ βλέπουμε ότι ανοίγει αρχικά τη σύνδεση με τη βάση δεδομένων, ενώ στην συνέχεια προσθέτει και σώζει μία εγγραφή μεταβλητής τύπου κλάσης Code, χωρίς να φαίνεται από πού έχει γεμίσει με δεδομένα η συγκεκριμένη μεταβλητή. Η συνάρτηση CancelSaveCode() υπάρχει μόνο για να ελέγχει αν υπάρχουν συμπληρωμένα τα πεδία του Τίτλου, του Ονόματος Χρήστη και του Κωδικού της εγγραφής, ώστε να ενεργοποιήσει το κουμπί της Αποθήκευσης.
Ας δούμε τώρα με ποιο τρόπο το αντικείμενο της κλάσης Code (MCode) συμπληρώνεται. Η συμπλήρωση γίνεται απευθείας από το View της καταχώρησης με Data Bindings σε όλα τα πεδία, όπως φαίνεται στον κώδικα XAML που ακολουθεί.

```
<TextBox x:Name="txbTitle" Text="{Binding Path=Title}".../>
<TextBox x:Name="txbUserName" Text="{Binding Path=UserName}"…/>
<TextBox x:Name="txbPassword" Text="{Binding Path=Password}"…>
<TextBox x:Name="txbUrl" Text="{Binding Path=Url}"…/>
<TextBox x:Name="txbNotes" Text="{Binding Path=Notes}"…/>
```

Όπως μπορούμε να παρατηρήσουμε κάθε TextBox WPF Control, το οποίο αποτελεί ένα πεδίο στο View μας, βλέπει απευθείας πίσω στον κώδικα το συγκεκριμένο πεδίο του αντικειμένου τύπου κλάσης Code. Οπότε με το που πληκτρολογούμε σε ένα πεδίο, για παράδειγμα του Τίτλου, αυτόματα περνιέται στο πεδίο Title του αντικειμένου μας. Έτσι όταν καλέσουμε την συνάρτηση SaveCode(), ότι υπάρχει στα πεδία του αντικειμένου, θα αποθηκευτούν σε μία νέα εγγραφή στη βάση δεδομένων.

Με το κουμπί καθαρισμός, διαγράφονται όλα τα δεδομένα που έχουμε περάσει στα πεδία του View.

Η διαγραφή ενός κώδικα γίνεται με τον ίδιο τρόπο όπως στην διαγραφή των πελατών. Δηλαδή, ενώ βρισκόμαστε στην εμφάνιση των προσωπικών κωδικών, επιλέγοντας έναν από αυτούς, ενεργοποιείτε το κουμπί της διαγραφής και πατώντας το, περνιέται στη συνάρτηση RemoveCode() το ID της εγγραφής, Στη συνέχεια γίνεται η σύνδεση με την βάση δεδομένων μας και διαγράφεται η εγγραφή με το συγκεκριμένο ID.

2.2.3 Εμφάνιση ομολόγων

Η τρίτη και τελευταία λειτουργία της εφαρμογής περιλαμβάνει τον υπολογισμό και την εμφάνιση κάποιων βασικών στοιχείων των ομολόγων, που είναι απαραίτητα να τα γνωρίζει ένας σύμβουλος επενδύσεων. Κάθε ομόλογο από την στιγμή που βγαίνει στην αγορά, διαθέτει κάποια χαρακτηριστικά στοιχεία, όπως η ημερομηνία έκδοσης, ο κλάδος που ανήκει, το χρηματιστήριο όπου παίζει, η πιστοληπτική του αξία, το ελάχιστο ποσό αγοράς και πολλά άλλα.

Όλα αυτά τα στοιχεία είναι διαθέσιμα μέσω κάποιων εταιρειών που έχουν συμβάσεις με τα χρηματιστήρια. Τέτοιες εταιρείες είναι η Thomson Reuters, η Bloomberg και άλλες. Για να μπορέσεις να έχεις αυτές τις πληροφορίες, θα πρέπει να υπογράψεις συγκεκριμένα συμβόλαια και να δίνεις κάποιο χρηματικό πόσο. Υπάρχουν βέβαια και κάποιες ελεύθερες τέτοιες βάσεις στο διαδίκτυο που σου παρέχουν κάποια από τα στοιχεία δωρεάν, όπως είναι για παράδειγμα η ιστοσελίδα της Yahoo, www.finance.yahoo.com.

Οπότε με έναν από τους προηγούμενους τρόπους μπορείς να έχεις στην διάθεση σου κάποια από τα στοιχεία των ομολόγων, ώστε μετέπειτα να υπολογίσεις διάφορα δευτερεύοντα στοιχεία που σε ενδιαφέρουν, όπως το κέρδος που θα έχω με κάποια συγκεκριμένη αγορά ή πόσο επιτόκιο θα μου επιφέρει ένα ομόλογο σε 5 χρόνια από τώρα.

Για τη συγκεκριμένη εφαρμογή χρησιμοποιήθηκαν διάσπαρτα δεδομένα από την σελίδα της Yahoo και της Google. Τα δεδομένα με ένα άλλο τρίτο πρόγραμμα, αποθηκεύονται στη βάση δεδομένων μας, ανά διαστήματα που ορίζουμε εμείς, και στην συνέχεια μέσω του ClientManager τα διαχειριζόμαστε και υπολογίζουμε τις τιμές που μας ενδιαφέρουν. Ο πίνακας που περιέχει τα δεδομένα που έχουν αποθηκευτεί από το εξωτερικό τρίτο πρόγραμμα είναι ο Rawdata και το μοντέλο του περιλαμβάνει το παρακάτω πεδία.

```
public partial class RawData
  {
     public long Id { get; set; }
     public string RiC { get; set; }
     public string Isin { get; set; }
     public System.DateTime IssueDate { get; set; }
```

```
public System.DateTime MatureDate { get; set; }
public string Name { get; set; }
public Nullable<decimal> Last { get; set; }
public Nullable<decimal> PctChange { get; set; }
public decimal Bid { get; set; }
public string BSize { get; set; }
public decimal Ask { get; set; }
public string ASize { get; set; }
public Nullable<long> Volume { get; set; }
public Nullable<System.DateTime> Date { get; set; }
public Nullable<System.TimeSpan> Time { get; set; }
public decimal Coupon { get; set; }
public string Exchange { get; set; }
public string DeptTimeDescription { get; set; }
public string Type { get; set; }
public string Ccy { get; set; }
public long AmountOutstanding { get; set; }
public long OriginalAmountIssued { get; set; }
public Nullable<System.DateTime> NextPayDate { get; set; }
public long DenominationMinimum { get; set; }
public long DenominationIncrement { get; set; }
public string PaymentFrequency { get; set; }
public long CouponFrequency { get; set; }
public string FitchsRating { get; set; }
public System.DateTime FitchsRatingDate { get; set; }
public string SpRating { get; set; }
public System.DateTime SpRatingDate { get; set; }
public string MoodysRating { get; set; }
public System.DateTime MoodysRatingDate { get; set; }
public Nullable<long> RedemptionValue { get; set; }
public string TypeOfReder { get; set; }
public string RedeemedValue { get; set; }
public Nullable<System.DateTime> CalledDate { get; set; }
public Nullable<decimal> CallPrice { get; set; }
```

```
public System.DateTime FirstCouponDate { get; set; }
public Nullable<System.DateTime> LastUpdate { get; set; }
public string Country { get; set; }
public string IndustrySector { get; set; }
  }
```

Στο ViewModel της εμφάνισης των ομολόγων, αποθηκεύονται τα στοιχεία αυτά σε ένα άλλο αντικείμενο που περιέχει τα ίδια πεδία συν τα πεδία για να μπουν τα αποτελέσματα από κάποιες πράξεις. Το μοντέλο της κλάσης αυτού του αντικειμένου είναι το Bond.

Θα παρουσιαστούν τώρα κάποια απλά παραδείγματα τέτοιων πράξεων ώστε να γίνει πιο κατανοητό. Ας δούμε τον υπολογισμό της διασποράς της τιμής του ομολόγου, το λεγόμενο Spread. Η πράξη είναι η εξής:

```
mBond.Spread = mBond.Ask - mBond.Bid;
```

Το mBond είναι ένα αντικείμενο τύπου κλάσης Bond. Τα πεδία mBond.Ask και mBond.Bid έχουν πάρει τιμή από το αντίστοιχο αντικείμενο τύπου κλάσης Rawdata, mrawData.Ask και mrawData.Bid. Το mBond.Ask περιέχει την τιμή που ζητούν οι πωλητές του ομολόγου και το mBond.Bid περιέχει την τιμή που ζητούν οι αγοραστές για να αποκτήσουν το ομόλογο. Καταλαβαίνουμε ότι αφαιρώντας τις δύο αυτές τις τιμές, όπου προκύπτει το Spread, μπορούμε να βγάλουμε διάφορα συμπεράσματα για την κατάσταση της αγοράς και αν μας συμφέρει να αγοράσουμε ή να πουλήσουμε.

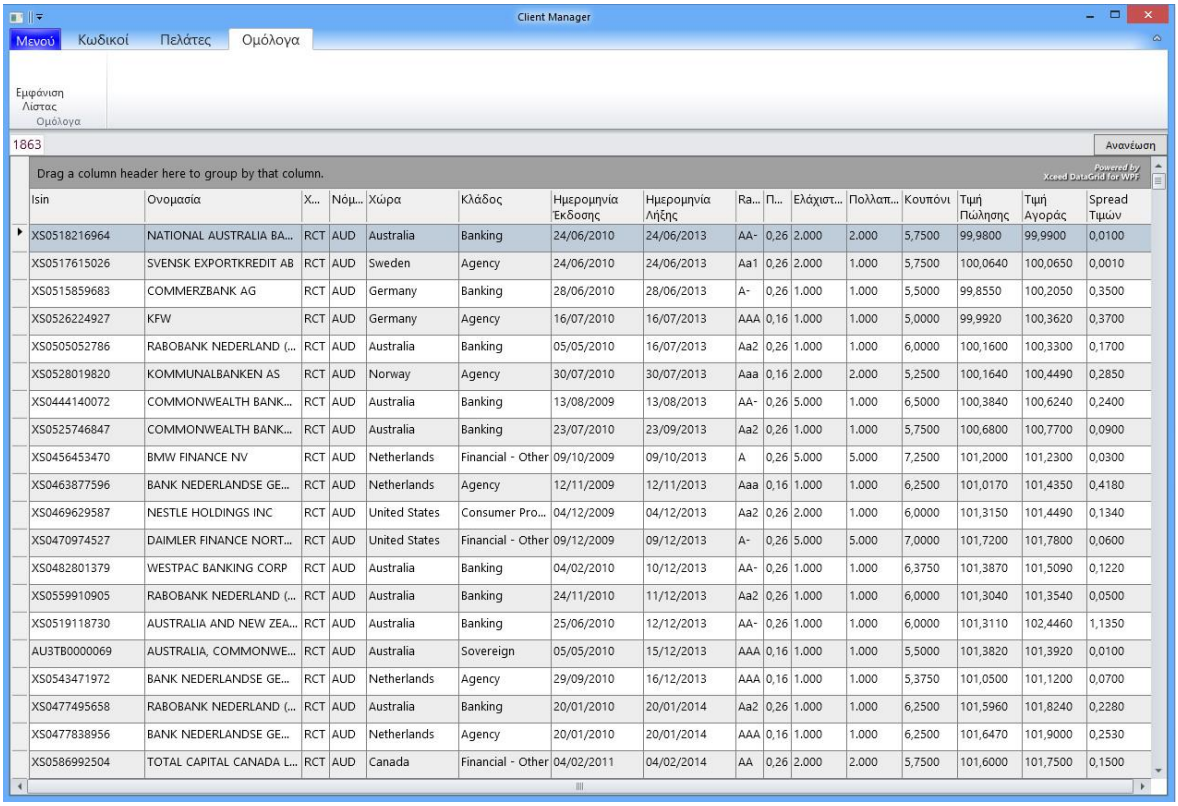
Στο επόμενο παράδειγμα μας θα υπολογίσουμε πόσο κοστίζει η προμήθεια αγοράς και πώλησης του ομολόγου. Ας δούμε τον τύπο:

```
mBond.Commision=Commision(mBond.Rating,mBond.Years,
mrawData.MoodysRating, mrawData.FitchsRating, mrawData.SpRating);
```

Η συνάρτηση Commision() δέχεται ως παραμέτρους την πιστοληπτική αξία του ομολόγου, πόσα χρόνια διαρκεί συνολικά και τις τιμές προμηθειών που ορίζουν τρεις μεγάλοι χρηματιστηριακοί οίκοι. Έτσι με αυτά τα στοιχεία υπολογίζει πόσο

κοστίζει η προμήθεια αγοράς και πώλησης κάθε χρόνο που περνάει, διότι εξαρτάται από τα χρόνια που βρίσκεται το ομόλογο στην αγορά. Το αποτέλεσμα αποθηκεύεται στο πεδίο Commision του αντικειμένου mBond. Κατά αυτόν τον τρόπο υπολογίζονται και αποθηκεύονται όλα τα πεδία, αντίστοιχα.

Όταν επιλέξουμε την Εμφάνιση Λίστας από την καρτέλα Ομόλογα μας εμφανίζεται το παρακάτω αποτέλεσμα:



Isin	Ονομασία	Χ...	Νόμ...	Χώρα	Κλάδος	Ημερομηνία Έκδοσης	Ημερομηνία Λήξης	Ra...	Π...	Ελάχιστ...	Πολλαπ...	Κουπόνι	Τιμή Πώλησης	Τιμή Αγοράς	Spread Τιμών
XS0518216964	NATIONAL AUSTRALIA BA...	RCT	AUD	Australia	Banking	24/06/2010	24/06/2013	AA-	0,26	2.000	2.000	5,7500	99,9800	99,9900	0,0100
XS0517615026	SVENSK EXPORTKREDIT AB	RCT	AUD	Sweden	Agency	24/06/2010	24/06/2013	Aa1	0,26	2.000	1.000	5,7500	100,0640	100,0650	0,0010
XS0515859683	COMMERZBANK AG	RCT	AUD	Germany	Banking	28/06/2010	28/06/2013	A-	0,26	1.000	1.000	5,5000	99,8550	100,2050	0,3500
XS0526224927	KFW	RCT	AUD	Germany	Agency	16/07/2010	16/07/2013	AAA	0,16	1.000	1.000	5,0000	99,9920	100,3620	0,3700
XS0505052786	RABOBANK NEDERLAND (...	RCT	AUD	Australia	Banking	05/05/2010	16/07/2013	Aa2	0,26	1.000	1.000	6,0000	100,1600	100,3300	0,1700
XS0528019820	KOMMUNALBANKEN AS	RCT	AUD	Norway	Agency	30/07/2010	30/07/2013	Aaa	0,16	2.000	2.000	5,2500	100,1640	100,4490	0,2850
XS0444140072	COMMONWEALTH BANK...	RCT	AUD	Australia	Banking	13/08/2009	13/08/2013	AA-	0,26	5.000	1.000	6,5000	100,3840	100,6240	0,2400
XS0525746847	COMMONWEALTH BANK...	RCT	AUD	Australia	Banking	23/07/2010	23/09/2013	Aa2	0,26	1.000	1.000	5,7500	100,6800	100,7700	0,0900
XS0456453470	BMW FINANCE NV	RCT	AUD	Netherlands	Financial - Other	09/10/2009	09/10/2013	A	0,26	5.000	5.000	7,2500	101,2000	101,2300	0,0300
XS0463877596	BANK NEDERLANDSE GE...	RCT	AUD	Netherlands	Agency	12/11/2009	12/11/2013	Aaa	0,16	1.000	1.000	6,2500	101,0170	101,4350	0,4180
XS0469629587	NESTLE HOLDINGS INC	RCT	AUD	United States	Consumer Pro...	04/12/2009	04/12/2013	Aa2	0,26	2.000	1.000	6,0000	101,3150	101,4490	0,1340
XS0470974527	DAIMLER FINANCE NORT...	RCT	AUD	United States	Financial - Other	09/12/2009	09/12/2013	A-	0,26	5.000	5.000	7,0000	101,7200	101,7800	0,0600
XS0482801379	WESTPAC BANKING CORP	RCT	AUD	Australia	Banking	04/02/2010	10/12/2013	AA-	0,26	1.000	1.000	6,3750	101,3870	101,5090	0,1220
XS0559910905	RABOBANK NEDERLAND (...	RCT	AUD	Australia	Banking	24/11/2010	11/12/2013	Aa2	0,26	1.000	1.000	6,0000	101,3040	101,3540	0,0500
XS0519118730	AUSTRALIA AND NEW ZEA...	RCT	AUD	Australia	Banking	25/06/2010	12/12/2013	AA-	0,26	1.000	1.000	6,0000	101,3110	102,4460	1,1350
AU3TB0000069	AUSTRALIA, COMMONWE...	RCT	AUD	Australia	Sovereign	05/05/2010	15/12/2013	AAA	0,16	1.000	1.000	5,5000	101,3820	101,3920	0,0100
XS0543471972	BANK NEDERLANDSE GE...	RCT	AUD	Netherlands	Agency	29/09/2010	16/12/2013	AAA	0,16	1.000	1.000	5,3750	101,0500	101,1200	0,0700
XS0477495658	RABOBANK NEDERLAND (...	RCT	AUD	Australia	Banking	20/01/2010	20/01/2014	Aa2	0,26	1.000	1.000	6,2500	101,5960	101,8240	0,2280
XS0477838956	BANK NEDERLANDSE GE...	RCT	AUD	Netherlands	Agency	20/01/2010	20/01/2014	AAA	0,16	1.000	1.000	6,2500	101,6470	101,9000	0,2530
XS0586992504	TOTAL CAPITAL CANADA L...	RCT	AUD	Canada	Financial - Other	04/02/2011	04/02/2014	AA	0,26	2.000	2.000	5,7500	101,6000	101,7500	0,1500

Εικόνα 2.8:Εμφάνιση ομολόγων

Το WPF Control που είναι υπεύθυνο για την εμφάνιση των ομολόγων, είναι το DataGrid που συναντήσαμε και στην εμφάνιση της λίστας των πελατών. Έτσι με τον ίδιο τρόπο, με Data Binding συνδέουμε το αντικείμενο τύπου Bond που περιέχει όλες τις τιμές, με την ιδιότητα ItemsSource του DataGrid Control.

```
Name="BondsDatagrid" ItemsSource="{Binding finishedBonds}"
```

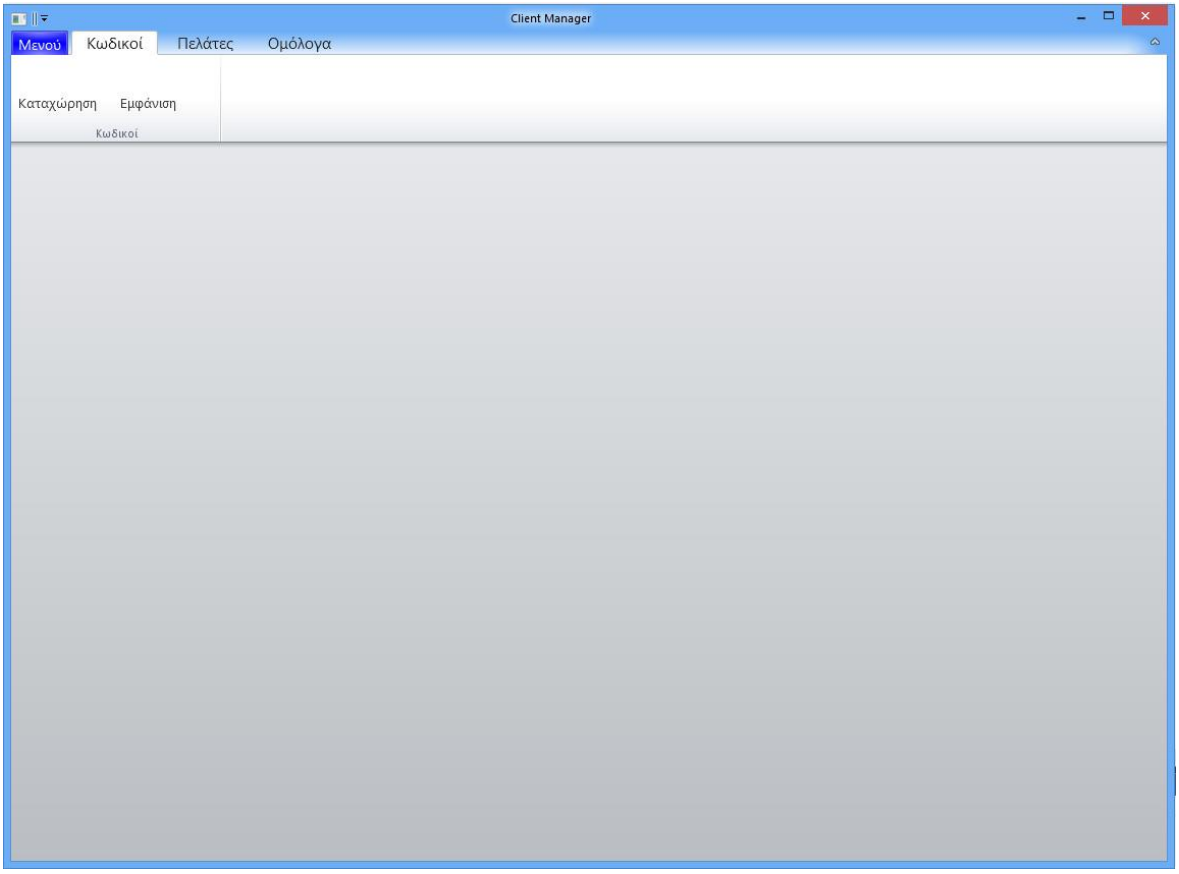
Οπότε μόλις καλούμε την συνάρτηση CalculateValues(), η οποία βρίσκεται στο ViewModel, υπολογίζονται οι τιμές και αποθηκεύονται στο finishedBonds που είναι

ένα αντικείμενο τύπου κλάσης Bond. Μόλις γίνει η αποθήκευση το Datagrid Control ενημερώνεται και μας εμφανίζει τις καινούργιες τιμές.

Πάνω αριστερά στην γωνία, μας εμφανίζει πόσα ομόλογα βρίσκονται στη λίστα και με το κουμπί Ανανέωση μπορούμε οποιαδήποτε στιγμή να δούμε αν έχει αλλάξει κάποια από τις τιμές, στην περίπτωση που υπάρχει κάποιο σφάλμα και δε έγινε αυτόματα.

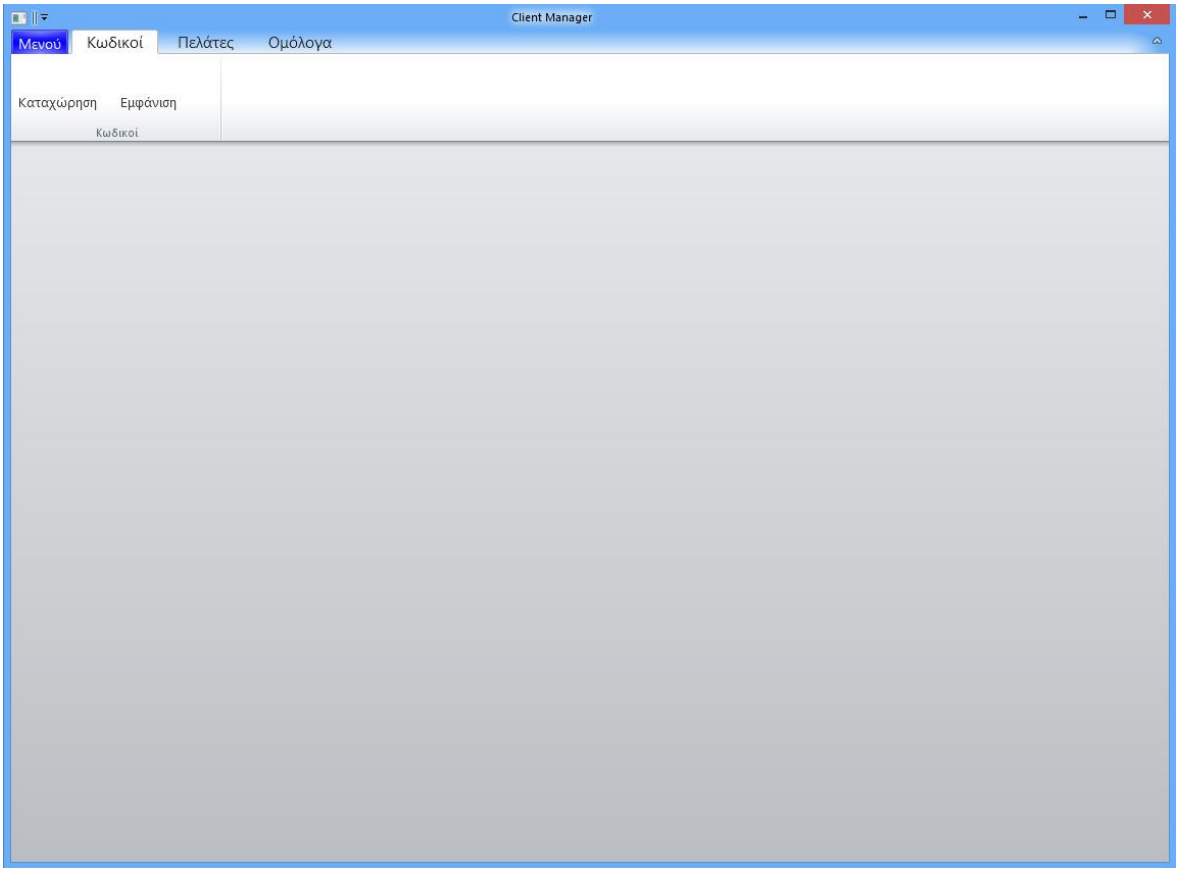
Κεφάλαιο 3
ΠΑΡΟΥΣΙΑΣΗ ΔΙΕΠΑΦΗΣ ΕΦΑΡΜΟΓΗΣ

Ξεκινώντας την εφαρμογή το αρχικό μενού που βλέπουμε είναι το εξής:



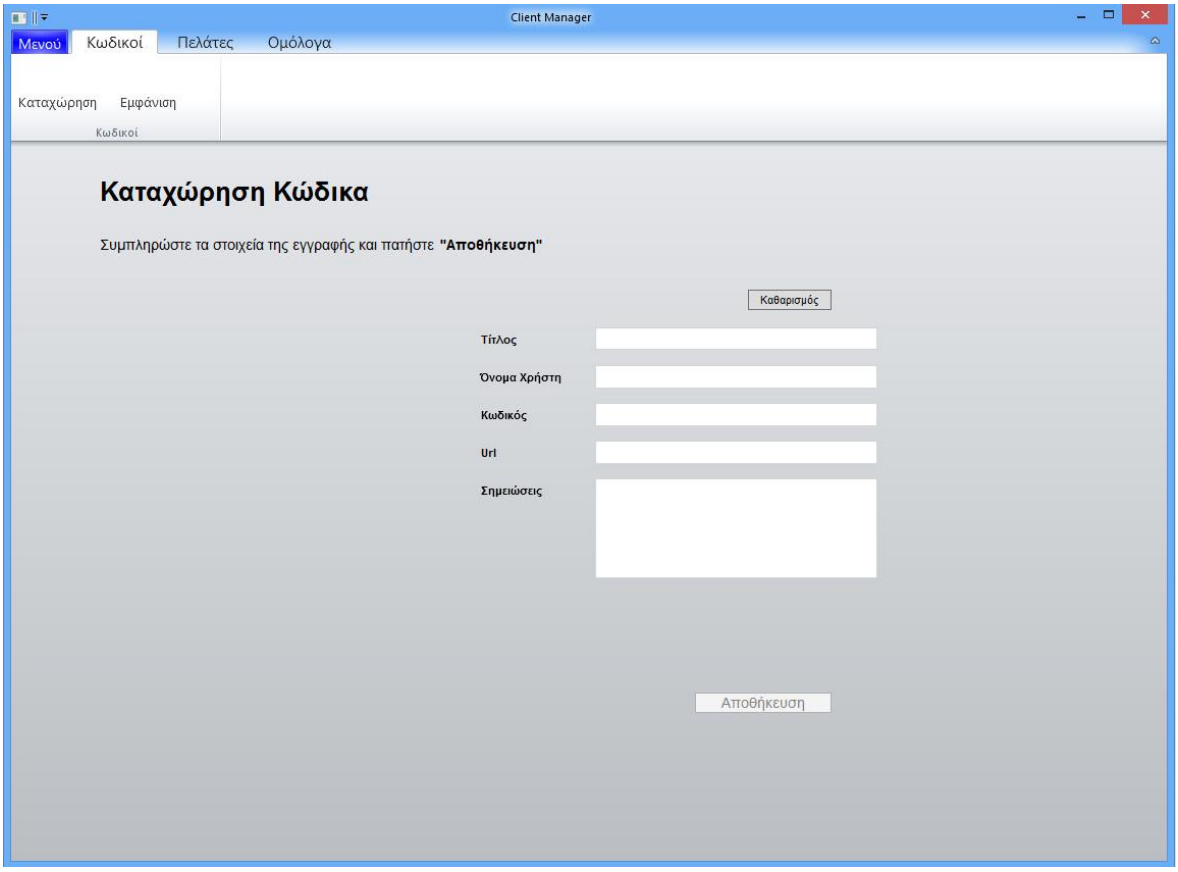
Εικόνα 3.1: Αρχικό μενού

Επιλέγοντας την καρτέλα Κωδικοί, οδηγούμαστε στο κομμάτι της εφαρμογής που αφορά την καταχώρηση και θέαση ονομάτων χρήστη και κωδικών για διάφορες χρήσιμες ιστοσελίδες που χρησιμοποιούμε συχνά.



Εικόνα 3.2: Μενού κωδικών

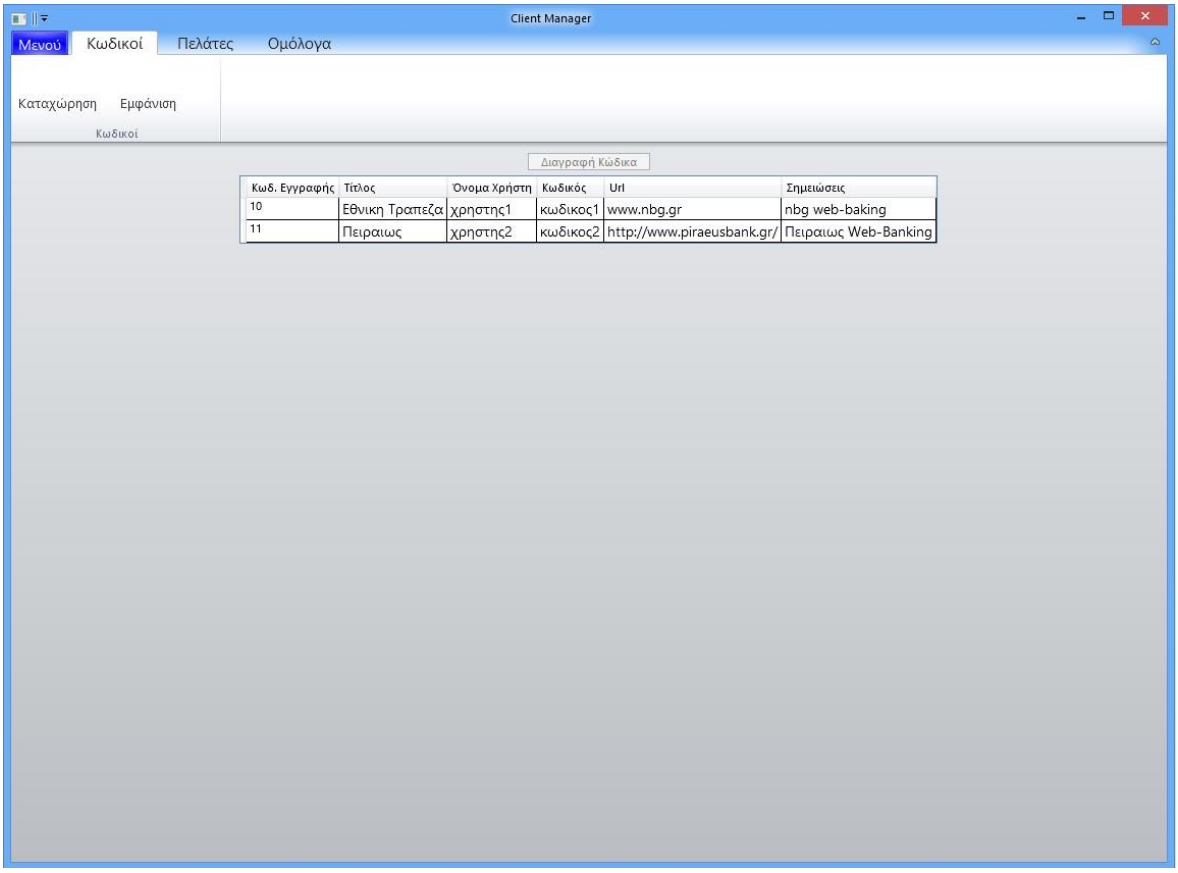
Επιλέγοντας την καταχώρηση, μεταβαίνουμε στην εξής σελίδα για καταχώρηση ενός νέου κωδικού.



Εικόνα 3.3: Καταχώρηση κωδικών

Εδώ μπορούμε να καταχωρήσουμε τίτλο για τον κωδικό (πχ.Εθνική τράπεζα), το όνομα χρήστη(username), τον κωδικό(password), την τοποθεσία του ιστότοπου (url) καθώς και κάποια σημείωση αν το επιθυμούμε. Επίσης, μπορούμε να κάνουμε καθαρισμό των στοιχείων αν έχουμε κάνει λάθη πατώντας το κουμπί καθαρισμός πάνω από το πεδίο εισαγωγής του τίτλου, αλλιώς επιλέγουμε αποθήκευση για να αποθηκευθούν τα στοιχεία που εισαγάγαμε.

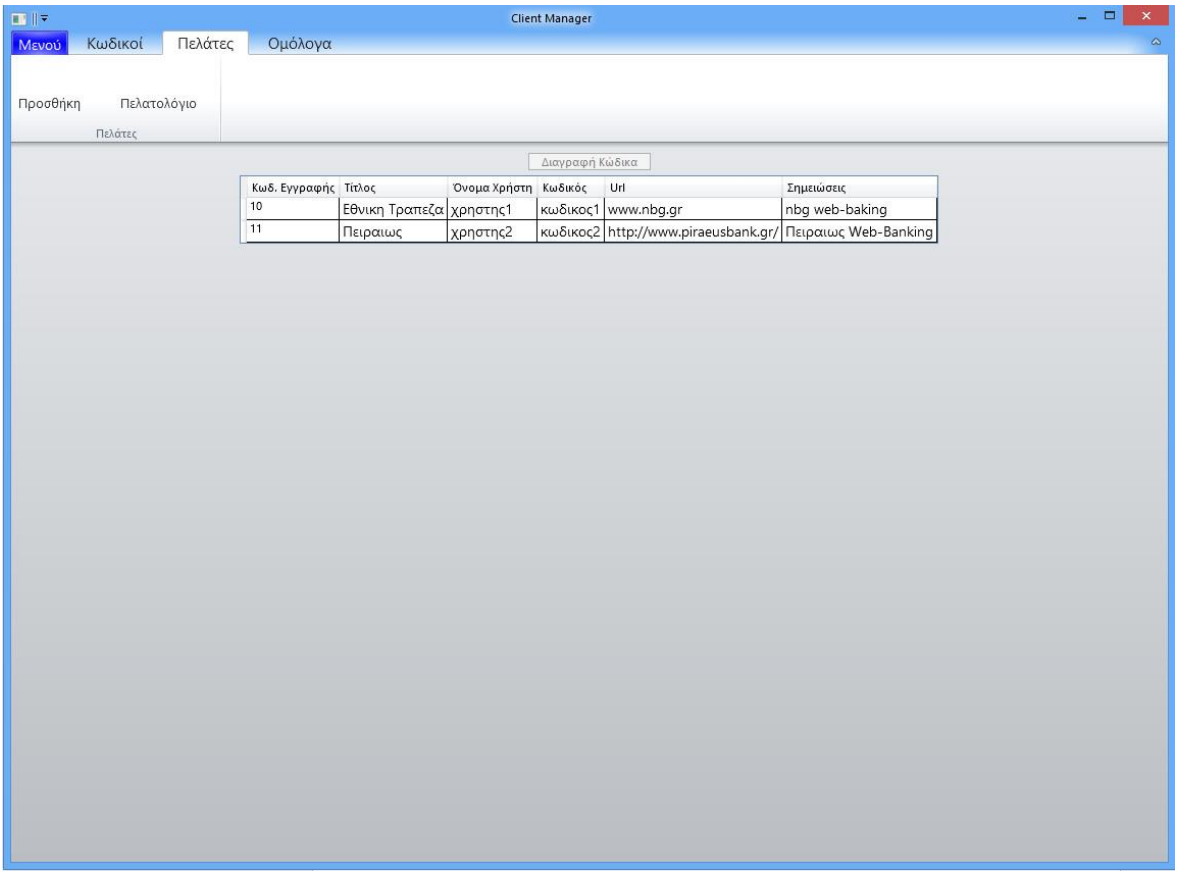
Επιλέγοντας εμφάνιση πάνω στην καρτέλα κωδικοί μεταφερόμαστε στην σελίδα όπου βλέπουμε τι κωδικούς έχουμε ήδη καταχωρήσει.



Εικόνα 3.4: Εμφάνιση κωδικών

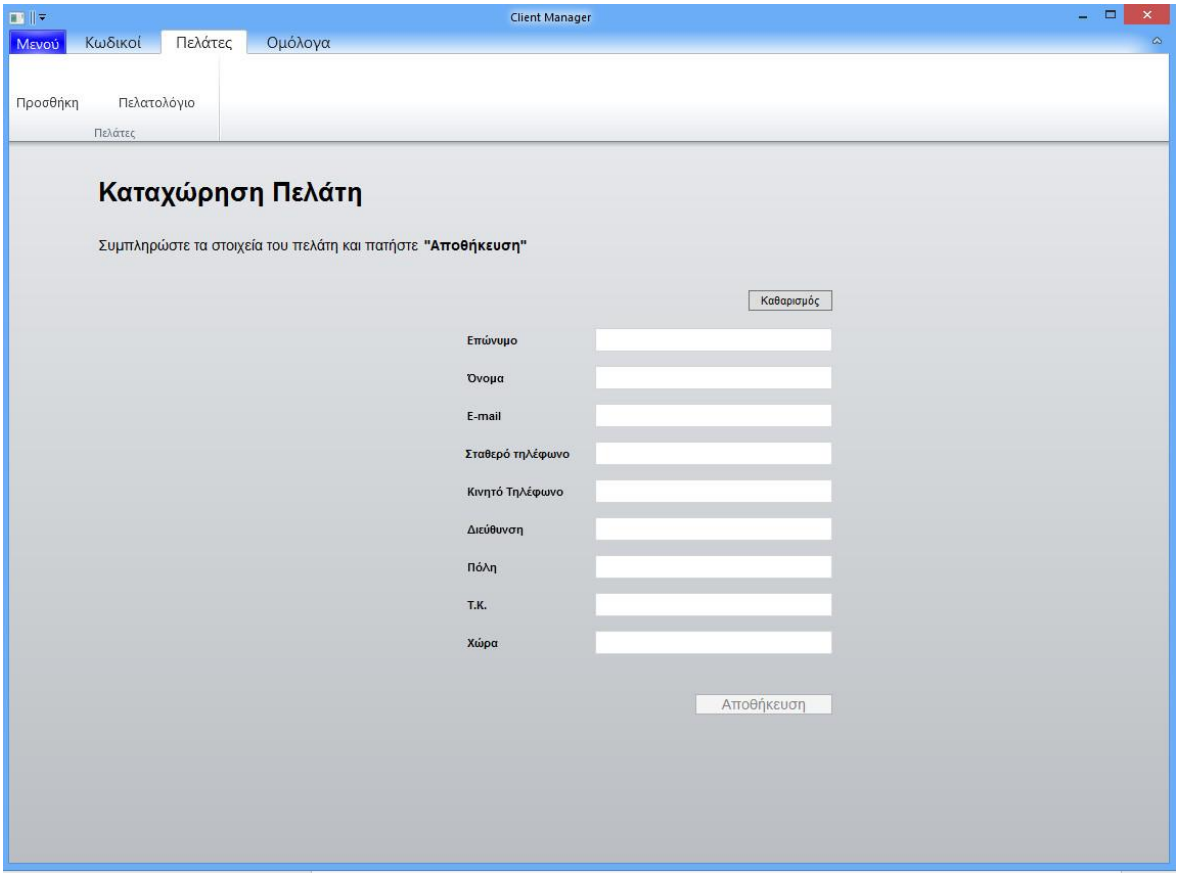
Σε αυτή την σελίδα βλέπουμε τους τίτλους και τα υπόλοιπα στοιχεία από τους καταχωρημένους μας κωδικούς.

Επιλέγοντας την καρτέλα πελάτες μπορούμε να κάνουμε καταχώρηση νέων πελατών ή να δούμε τους ήδη υπάρχοντες.



Εικόνα 3.5: Καρτέλα πελάτες

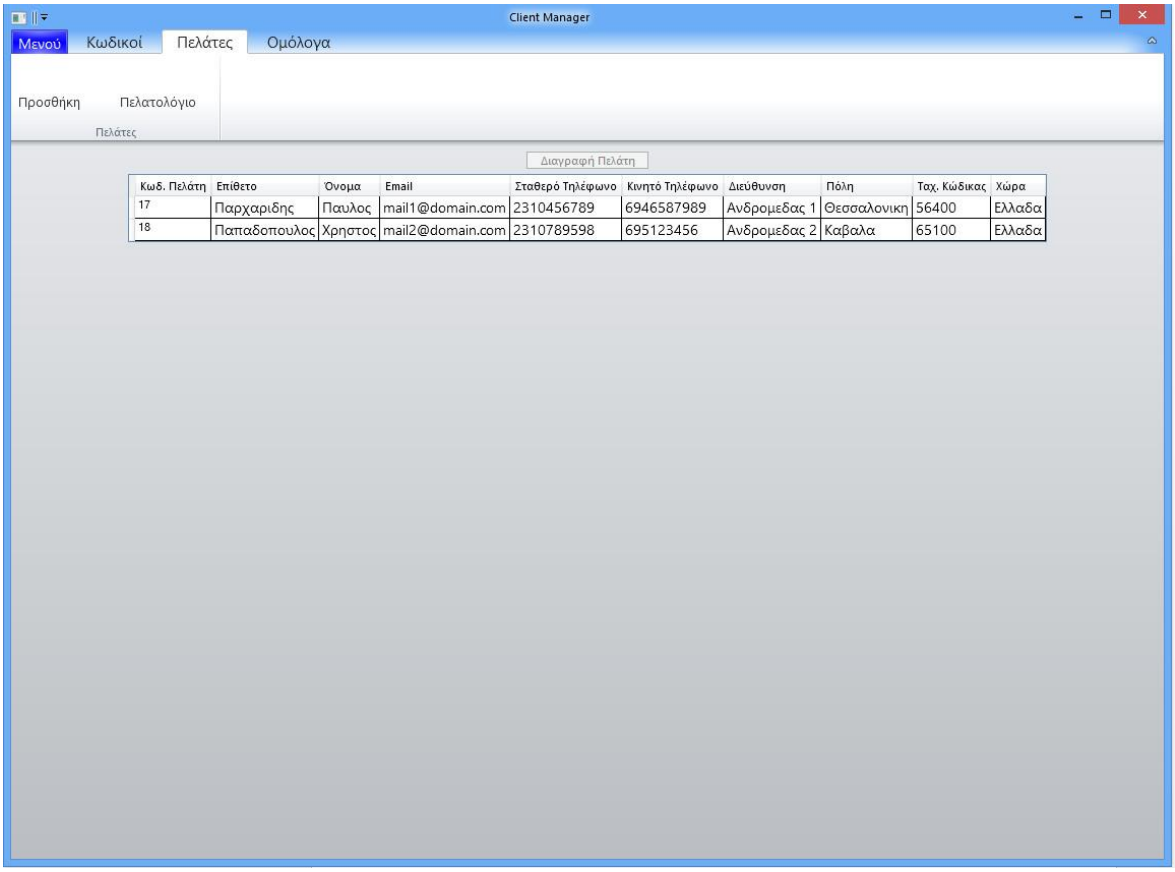
Επιλέγοντας προσθήκη μεταφερόμαστε στην σελίδα όπου μπορούμε να καταχωρήσουμε νέους πελάτες.



Εικόνα 3.6: Προσθήκη πελάτη

Εδώ για κάθε πελάτη μπορούμε να καταχωρήσουμε επώνυμο, όνομα, e-mail, σταθερό τηλέφωνο, κινητό τηλέφωνο, διεύθυνση, πόλη, ταχυδρομικό κωδικό της πόλης και την χώρα του. Μπορούμε να κάνουμε καθαρισμό των στοιχείων με το κουμπί καθαρισμός αλλιώς να αποθηκεύσουμε τον νέο πελάτη με το κουμπί αποθήκευση.

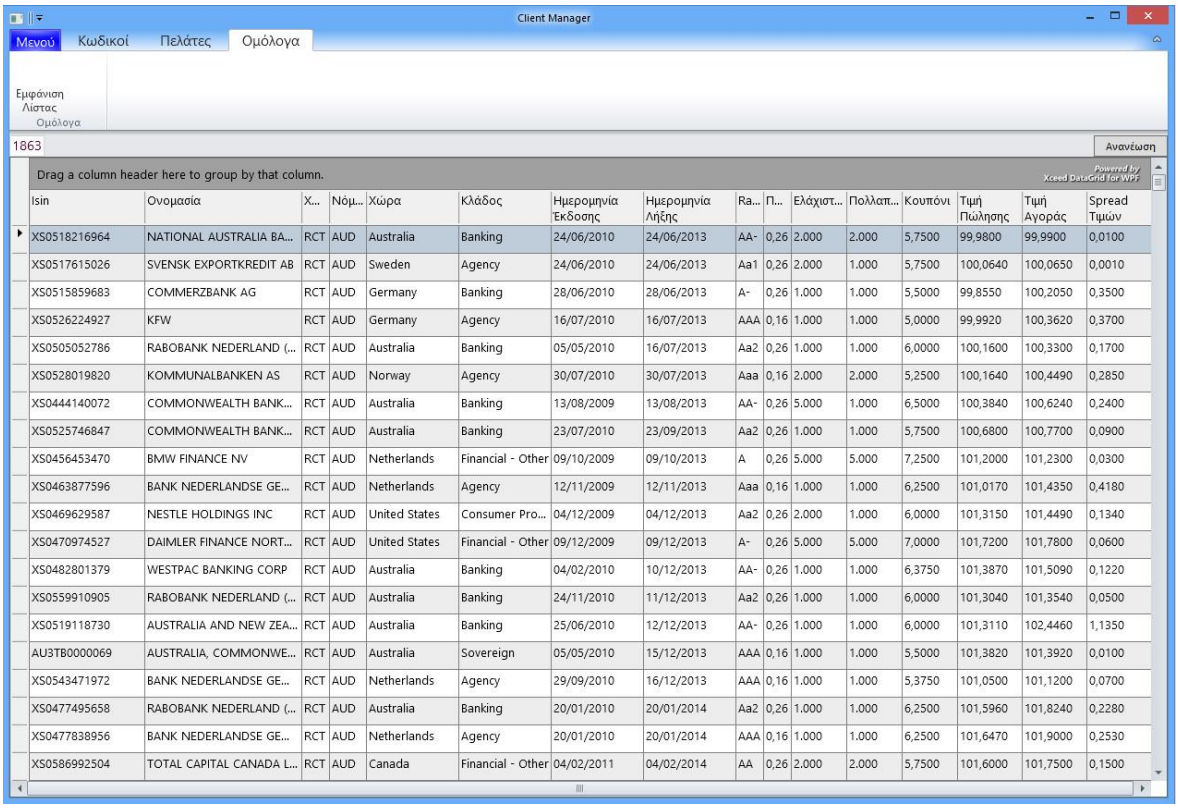
Επιλέγοντας το πελατολόγιο από την καρτέλα πελάτες μεταφερόμαστε στην σελίδα όπου βλέπουμε όλους τους καταχωρημένους πελάτες μας.



Εικόνα 3.7: Εμφάνιση πελατών

Εδώ φαίνονται όλα τα στοιχεία για κάθε καταχωρημένο πελάτη και επίσης επιλέγοντας έναν πελάτη μπορούμε να τον διαγράψουμε πατώντας στο κουμπί διαγραφή πελάτη.

Επιλέγοντας την καρτέλα ομόλογα και πατώντας στην εμφάνιση λίστας, μπορούμε να δούμε την λίστα με όλα τα ομόλογα με τις τρέχουσες τιμές τους(live) .

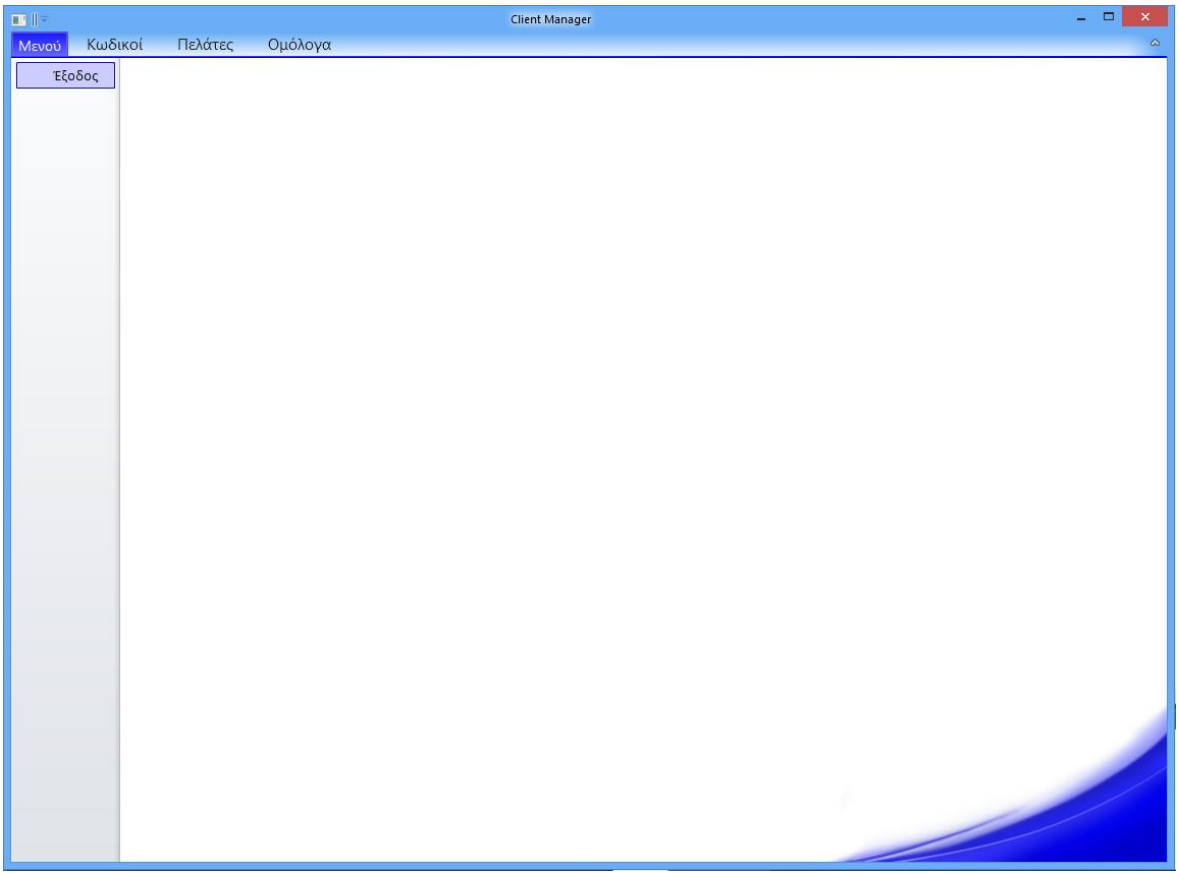


Isin	Ονομασία	Χ...	Νόμ...	Χώρα	Κλάδος	Ημερομηνία Έκδοσης	Ημερομηνία Λήξης	Ra...	Π...	Ελάχιστ...	Πολλαπ...	Κουπόνι	Τιμή Πώλησης	Τιμή Αγοράς	Spread Τιμών
XS0518216964	NATIONAL AUSTRALIA BA...	RCT	AUD	Australia	Banking	24/06/2010	24/06/2013	AA-	0,26	2.000	2.000	5,7500	99,9800	99,9900	0,0100
XS0517615026	SVENSK EXPORTKREDIT AB	RCT	AUD	Sweden	Agency	24/06/2010	24/06/2013	Aa1	0,26	2.000	1.000	5,7500	100,0640	100,0650	0,0010
XS0515859683	COMMERZBANK AG	RCT	AUD	Germany	Banking	28/06/2010	28/06/2013	A-	0,26	1.000	1.000	5,5000	99,8550	100,2050	0,3500
XS0526224927	KFW	RCT	AUD	Germany	Agency	16/07/2010	16/07/2013	AAA	0,16	1.000	1.000	5,0000	99,9920	100,3620	0,3700
XS0505052786	RABOBANK NEDERLAND (...	RCT	AUD	Australia	Banking	05/05/2010	16/07/2013	Aa2	0,26	1.000	1.000	6,0000	100,1600	100,3300	0,1700
XS0528019820	KOMMUNALBANKEN AS	RCT	AUD	Norway	Agency	30/07/2010	30/07/2013	Aaa	0,16	2.000	2.000	5,2500	100,1640	100,4490	0,2850
XS0444140072	COMMONWEALTH BANK...	RCT	AUD	Australia	Banking	13/08/2009	13/08/2013	AA-	0,26	5.000	1.000	6,5000	100,3840	100,6240	0,2400
XS0525746847	COMMONWEALTH BANK...	RCT	AUD	Australia	Banking	23/07/2010	23/09/2013	Aa2	0,26	1.000	1.000	5,7500	100,6800	100,7700	0,0900
XS0456453470	BMW FINANCE NV	RCT	AUD	Netherlands	Financial - Other	09/10/2009	09/10/2013	A	0,26	5.000	5.000	7,2500	101,2000	101,2300	0,0300
XS0463877596	BANK NEDERLANDSE GE...	RCT	AUD	Netherlands	Agency	12/11/2009	12/11/2013	Aaa	0,16	1.000	1.000	6,2500	101,0170	101,4350	0,4180
XS0469629587	NESTLE HOLDINGS INC	RCT	AUD	United States	Consumer Pro...	04/12/2009	04/12/2013	Aa2	0,26	2.000	1.000	6,0000	101,3150	101,4490	0,1340
XS0470974527	DAIMLER FINANCE NORT...	RCT	AUD	United States	Financial - Other	09/12/2009	09/12/2013	A-	0,26	5.000	5.000	7,0000	101,7200	101,7800	0,0600
XS0482801379	WESTPAC BANKING CORP	RCT	AUD	Australia	Banking	04/02/2010	10/12/2013	AA-	0,26	1.000	1.000	6,3750	101,3870	101,5090	0,1220
XS0559910905	RABOBANK NEDERLAND (...	RCT	AUD	Australia	Banking	24/11/2010	11/12/2013	Aa2	0,26	1.000	1.000	6,0000	101,3040	101,3540	0,0500
XS0519118730	AUSTRALIA AND NEW ZEA...	RCT	AUD	Australia	Banking	25/06/2010	12/12/2013	AA-	0,26	1.000	1.000	6,0000	101,3110	102,4460	1,1350
AU3TB0000069	AUSTRALIA, COMMONWE...	RCT	AUD	Australia	Sovereign	05/05/2010	15/12/2013	AAA	0,16	1.000	1.000	5,5000	101,3820	101,3920	0,0100
XS0543471972	BANK NEDERLANDSE GE...	RCT	AUD	Netherlands	Agency	29/09/2010	16/12/2013	AAA	0,16	1.000	1.000	5,3750	101,0500	101,1200	0,0700
XS0477495658	RABOBANK NEDERLAND (...	RCT	AUD	Australia	Banking	20/01/2010	20/01/2014	Aa2	0,26	1.000	1.000	6,2500	101,5960	101,8240	0,2280
XS0477838956	BANK NEDERLANDSE GE...	RCT	AUD	Netherlands	Agency	20/01/2010	20/01/2014	AAA	0,16	1.000	1.000	6,2500	101,6470	101,9000	0,2530
XS0586992504	TOTAL CAPITAL CANADA L...	RCT	AUD	Canada	Financial - Other	04/02/2011	04/02/2014	AA	0,26	2.000	2.000	5,7500	101,6000	101,7500	0,1500

Εικόνα 3.8: Εμφάνιση ομολόγων

Σε αυτή την καρτέλα βλέπουμε για κάθε ομόλογο τον κωδικό του (Isin), την ονομασία του, το χρηματιστήριο στο οποίο παίζει, το νόμισμα του, την χώρα, τον κλάδο στον οποίο ανήκει, την ημερομηνία έκδοσής του, την ημερομηνία λήξης του, την αξιολόγηση του (rating), την προμήθεια του, την ελάχιστη αγορά του, το πολλαπλάσιο, το κουπόνι του, την τρέχουσα τιμή πώλησης, την τιμή αγοράς του, και το spread των τιμών.

Επιλέγοντας την καρτέλα Μενού με το μπλε χρώμα μπορούμε να βγούμε από την εφαρμογή.



Εικόνα 3.9: Έξοδος από την εφαρμογή

ΣΥΜΠΕΡΑΣΜΑΤΑ

Όπως είδαμε με την ανάλυση των τεχνολογιών και της εφαρμογής τους στα επιμέρους κομμάτια της εφαρμογής, μπορούμε πια να δημιουργούμε εφαρμογές οι οποίες να είναι πολύ πιο εύκολα συντηρήσιμες και να υλοποιούνται γρηγορότερα διότι τα διαφορετικά κομμάτια τους μπορούν να αναπτύσσονται από διαφορετικές ομάδες παράλληλα χωρίς να επηρεάζεται το έργο της μιας ομάδας από την άλλη.

Πιο συγκεκριμένα, με το Windows Presentation Foundation (WPF) γίνεται σαφής διαχωρισμός μεταξύ του γραφικού περιβάλλον χρήστη (GUI) και του προγραμματιστικού κομματιού, περαιτέρω το πρότυπο Model View ViewModel (MVVM) το οποίο βασίζεται στην ιδεολογία του WPF και την πηγαίνει ένα βήμα παρακάτω μας βοηθάει ακόμα περισσότερο να εφαρμόσουμε τον διαχωρισμό ανάπτυξης του GUI από την λογική του προγράμματος.

Επίσης, το toolkit της Galasoft, το MVVM Light Toolkit, μας παρέχει έτοιμες πολλές βιβλιοθήκες που αλλιώς έπρεπε να τις ετοιμάσουμε εμείς. Από αυτό το toolkit έχουμε έτοιμες βιβλιοθήκες για τα bindings και τα commands που απαιτούνται για την υλοποίηση του MVVM.

Αυτό, προγενέστερα με την χρήση των Windows Forms ήταν πολύ πιο δύσκολο να επιτευχθεί διότι η λογική του προγράμματος ήταν στενά συνδεδεμένη με το γραφικό περιβάλλον του χρήστη. Πίσω από κάθε γραφικό κομμάτι της εφαρμογής μας υπήρχε ο κώδικας και σε κάθε αλλαγή του γραφικού περιβάλλοντος έπρεπε να ξαναγραφεί ο κώδικας, πράγμα που έκανε πιο πολύπλοκη την διαδικασία και χρονοβόρα.

Με το WPF και το πρότυπο MVVM, αποφεύγουμε τέτοιου είδους προβλήματα γιατί ο κώδικας μας πια, η λογική του προγράμματος, γίνεται τελείως ανεξάρτητη το γραφικό περιβάλλον της εφαρμογής μας. Μπορούμε να συγγράψουμε κώδικα ο οποίος να χρησιμοποιείται από διάφορες εφαρμογές με ελάχιστες μετατροπές πάνω σε αυτόν.

Το Entity Framework από την πλευρά του, μας επιτρέπει να σχεδιάζουμε τις βάσεις δεδομένων πολύ γρηγορότερα είτε μέσω κώδικα είτε μέσω του SQL Management Studio, και να τις συνδέουμε με το πρόγραμμα μας αυτόματα με την δημιουργία των αναγκαίων κλάσεων που δημιουργούνται από το Entity Framework αυτόματα για εμάς, και κάθε αλλαγή στην βάση μας ανανεώνει αυτόματα επίσης και το γραφικό κομμάτι της εφαρμογής μας αλλά και την ίδια την βάση.

Μια παρατήρηση που μπορεί να γίνει πάνω στην νέα ιδεολογία που μας προσφέρει το WPF και ειδικότερα το πρότυπο MVVM είναι ότι σε εφαρμογές μικρού μεγέθους, όπως είναι η συγκεκριμένη εφαρμογή αυτής της πτυχιακής εργασίας, δημιουργείται περισσότερη εργασία για τον προγραμματιστή ίσως από το να δούλευε με τον κλασσικό τρόπο σκέψης των Windows Forms, που μπορούμε να το κάνουμε και με το WPF αυτό φυσικά, αλλά στην συγκεκριμένη περίπτωση η εργασία αυτή είχε σκοπό να παρουσιάσει αυτές τις νέες τεχνολογίες και για αυτό η εφαρμογή υλοποιήθηκε έτσι.

Εν τέλει, όταν πρόκειται να αναπτύξουμε μια εφαρμογή πρέπει να βλέπουμε το μέγεθος της (scope) και να διαλέγουμε ποιες μεθοδολογίες θα μας βοηθήσουν ώστε η εφαρμογή μας να γίνει πιο γρήγορα, πιο αποδοτικά και να είναι πιο εύκολα συντηρήσιμη. Το WPF και το MVVM, αν και μας προσφέρουν την πιο καθαρή λογική για την ανάπτυξη εφαρμογών, προς το παρόν, μπορούν αν δεν έχουμε κατανοήσει καλά την λογική τους να οδηγήσουν σε εφαρμογές που απαιτούν περισσότερους πόρους από το σύστημα σε σχέση με το να τις είχαμε αναπτύξει με προγενέστερες ιδεολογίες ανάπτυξης λογισμικού.

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Για την χρήση της εφαρμογής πρέπει να έχουμε εγκατεστημένη μια έκδοση του Microsoft Visual Studio,προτιμότερο είναι να έχουμε από την έκδοση 2012 και πάνω πχ. Microsoft Visual Studio 2012 Express. Επίσης, θα χρειαστούμε τον Microsoft SQL Server 2008 R2 ή κάποιον νεότερο. Επίσης, στο Visual Studio θα πρέπει να έχουμε εγκατεστημένες τις βιβλιοθήκες (libraries) Galasoft Mvvm Light 4.1.24.0 και πάνω, καθώς και το Microsoft Ribbon for WPF.

Έπειτα, στον SQL Server δημιουργούμε μια νέα βάση δεδομένων με ονομασία BondTracker, και κάνουμε restore database σε αυτήν το αρχείο backup το οποίο βρίσκουμε στον φάκελο database μέσα στο project ClientManager2 που έχει ονομασία Bond, έχοντας επιλεγμένη την επιλογή να αντικατασταθεί η βάση με το backup, επιλέγουμε το πλήρες backup με την πιο πρόσφατη ημερομηνία.

Επίσης, μέσα από το Visual Studio πρέπει να αλλάξουμε στα αρχεία App.Config που βρίσκονται ένα στον φάκελο Views και ένα στον φάκελο Models το path για τον SQL Server μας, έτσι αν το instance του server μας είναι Paul\SQLEXPRESS το ορίζουμε ως path.

Τελικώς, κάνουμε build solution και τρέχουμε την εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Matthew MacDonald (2012), Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5, Professional Apress, pp. 1-269, pp. 555-601.

Internet Links:

http://www.codeproject.com/Articles/100175/Model-View-ViewModel-MVVM-Explained (Επίσκεψη Δεκέμβριος 2014)

https://entityframework.codeplex.com/ (Επίσκεψη Δεκέμβριος 2014)

http://msdn.microsoft.com/en-us/library/aa970268%28v=vs.110%29.aspx (Επίσκεψη Δεκέμβριος 2014)

http://reedcopsey.com/series/windows-forms-to-mvvm/ (Επίσκεψη Δεκέμβριος 2014)

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ

1. Όψεις (Views):

1.1 BondsView (Εμφάνιση ομολόγων):

```
<UserControl x:Class="BondTracker.Views.BondsView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:xcdg="http://schemas.xceed.com/wpf/xaml/datagrid">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="1">
            <xcdg:DataGridControl Name="BondsDatagrid" ItemsSource="{Binding
finishedBonds}" AutoCreateColumns="False" FontSize="13" ReadOnly="True"   >

                <xcdg:DataGridControl.Columns>
                    <xcdg:Column   FieldName="Isin"  Title="Isin"    />
                    <xcdg:Column   Width="170" FieldName="Name" Title="Ονομασία"
/>
                    <xcdg:Column   Width="30" FieldName="Exchange"
Title="Χρηματιστήριο"    />
                    <xcdg:Column   Width="40" FieldName="Ccy" Title="Νόμισμα"    />
                    <xcdg:Column   Width="100" FieldName="Country" Title="Χώρα"    />
                    <xcdg:Column   Width="100" FieldName="IndustrySector"
Title="Κλάδος"    />
                    <xcdg:Column   Width="100" FieldName="IssueDate"
Title="Ημερομηνία Έκδοσης">
                        <xcdg:Column.CellContentTemplate>
                            <DataTemplate>
                                <TextBlock Text="{Binding StringFormat=dd/MM/yyyy}"/>
                            </DataTemplate>
                        </xcdg:Column.CellContentTemplate>
                        <xcdg:Column.TitleTemplate>
                       <DataTemplate>
                                <TextBlock TextWrapping="Wrap" Text="{Binding}" />
                            </DataTemplate>
                        </xcdg:Column.TitleTemplate>
                    </xcdg:Column>
                    <xcdg:Column   Width="100" FieldName="MatureDate"
Title="Ημερομηνία Λήξης">
                        <xcdg:Column.CellContentTemplate>
                            <DataTemplate>
```

```
                        <TextBlock Text="{Binding StringFormat=dd/MM/yyyy}"/>
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
                <xcdg:Column.TitleTemplate>
                    <DataTemplate>
                        <TextBlock TextWrapping="Wrap" Text="{Binding}" />
                    </DataTemplate>
                </xcdg:Column.TitleTemplate>
            </xcdg:Column>
            <xcdg:Column   Width="30" FieldName="Rating" Title="Rating"    >
            </xcdg:Column>
            <xcdg:Column   Width="30" FieldName="Commision"
Title="Προμήθεια"   >
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding ConverterCulture=da-DK}" />
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
            </xcdg:Column>
          <xcdg:Column   Width="60" FieldName="DenominationMinimum"
Title="Ελάχιστη Αγορά"   >
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding StringFormat={}{0:N0},
ConverterCulture=da-DK}" />
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
            </xcdg:Column>
            <xcdg:Column   Width="60" FieldName="DenominationIncrement"
Title="Πολλαπλάσια"  >
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding StringFormat={}{0:N0},
ConverterCulture=da-DK}" />
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
            </xcdg:Column>
            <xcdg:Column   Width="60" FieldName="Coupon" Title="Κουπόνι"   >
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding StringFormat={}{0:N4},
ConverterCulture=da-DK}" />
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
            </xcdg:Column>
            <xcdg:Column   Width="70" FieldName="Bid" Title="Τιμή Πώλησης">
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding StringFormat={}{0:N4},
ConverterCulture=da-DK}" />
```

```
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
                <xcdg:Column.TitleTemplate>
                    <DataTemplate>
                        <TextBlock TextWrapping="Wrap" Text="{Binding}" />
                    </DataTemplate>
                </xcdg:Column.TitleTemplate>
            </xcdg:Column>
            <xcdg:Column   Width="70" FieldName="Ask" Title="Τιμή Αγοράς"   >
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding StringFormat={}{0:N4},
ConverterCulture=da-DK}" />
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
                <xcdg:Column.TitleTemplate>
                    <DataTemplate>
                        <TextBlock TextWrapping="Wrap" Text="{Binding}" />
                    </DataTemplate>
                </xcdg:Column.TitleTemplate>
            </xcdg:Column>
            <xcdg:Column   Width="70" FieldName="Spread" Title="Spread
Τιμών">
                <xcdg:Column.CellContentTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding StringFormat={}{0:N4},
ConverterCulture=da-DK}" />
                    </DataTemplate>
                </xcdg:Column.CellContentTemplate>
                <xcdg:Column.TitleTemplate>
                    <DataTemplate>
                        <TextBlock TextWrapping="Wrap" Text="{Binding}" />
                    </DataTemplate>
                </xcdg:Column.TitleTemplate>
            </xcdg:Column>
        </xcdg:DataGridControl.Columns>
    </xcdg:DataGridControl>
</DockPanel>
<Button Grid.Row="0" Name="btnRefresh" Height="23" Width="80"
HorizontalAlignment="Right" Content="Ανανέωση" Command="{Binding
Path=RefreshBondsViewCommand}"  />
<TextBox Grid.Row="0" Name="txbCountRows" HorizontalAlignment="Left"
Height="23" Width="Auto" Text="{Binding Path=NumberOfRows}"
IsReadOnly="True" Cursor="Arrow" FontSize="14"/>
</Grid>
</UserControl>
```

1.2 CodesView (Εμφάνιση κωδικών):

```
<UserControl x:Class="BondTracker.Views.CodesView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" >
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <DockPanel Grid.Row="1">
            <DataGrid Name="CodeDatagrid" Height="Auto"  Margin="5"
                      AutoGenerateColumns="False" ItemsSource="{Binding
Codes,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}"
                      VerticalAlignment="Top" HorizontalAlignment="Center"
Width="Auto" IsReadOnly="True" SelectionMode="Single" SelectedItem="{Binding
SelectedCode}"  >
                <DataGrid.Columns>
                    <DataGridTextColumn   Header="Κωδ. Εγγραφής"  FontSize="11"
Binding="{Binding Id}"   />
                    <DataGridTextColumn Header="Τίτλος" FontSize="14"
Binding="{Binding Title}"/>
                    <DataGridTextColumn Header="Όνομα Χρήστη" FontSize="14"
Binding="{Binding UserName}" />
                    <DataGridTextColumn Header="Κωδικός" FontSize="14"
Binding="{Binding Password}" />
                    <DataGridTextColumn Header="Url" FontSize="14" Binding="{Binding
Url}" />
                    <DataGridTextColumn Header="Σημειώσεις" FontSize="14"
Binding="{Binding Notes}" />
                </DataGrid.Columns>
            </DataGrid>
        </DockPanel>

        <Button Grid.Row="0" Name="btnRemoveCode" Command="{Binding
Path=RemoveCodeCommand}" IsEnabled="{Binding
ElementName=CodeDatagrid, Path=SelectedItems.Count}" Content="Διαγραφή
Κώδικα" HorizontalAlignment="Center" VerticalAlignment="Top" Width="120"
Margin="278,5,278,0"  />
    </Grid>
</UserControl>
```

1.3 CustomersView (Εμφάνιση πελατών):

```
<UserControl x:Class="BondTracker.Views.CustomersView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             >
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <DockPanel Grid.Row="1">
            <DataGrid Name="CustDatagrid" Height="Auto"  Margin="5"
                      AutoGenerateColumns="False" ItemsSource="{Binding
Customers,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}"
                      VerticalAlignment="Top" HorizontalAlignment="Center"
Width="Auto" IsReadOnly="True" SelectionMode="Single" SelectedItem="{Binding
SelectedCustomer}"  >

                <DataGrid.Columns>
                    <DataGridTextColumn   Header="Κωδ. Πελάτη"  FontSize="11"
Binding="{Binding CustomerId}"   />
                    <DataGridTextColumn Header="Επίθετο" FontSize="14"
Binding="{Binding LastName}"/>
                    <DataGridTextColumn Header="Όνομα" FontSize="14"
Binding="{Binding FirstName}" />
                    <DataGridTextColumn Header="Email" FontSize="14"
Binding="{Binding Email}" />
                    <DataGridTextColumn Header="Σταθερό Τηλέφωνο" FontSize="14"
Binding="{Binding LocalPhoneNumb}" />
                    <DataGridTextColumn Header="Κινητό Τηλέφωνο" FontSize="14"
Binding="{Binding CellPhoneNumb}" />
                    <DataGridTextColumn Header="Διεύθυνση" FontSize="14"
Binding="{Binding Address}" />
                    <DataGridTextColumn Header="Πόλη" FontSize="14"
Binding="{Binding City}" />
                    <DataGridTextColumn Header="Ταχ. Κώδικας" FontSize="14"
Binding="{Binding ZipCode}" />
                    <DataGridTextColumn Header="Χώρα" FontSize="14"
Binding="{Binding Country}" />
                </DataGrid.Columns>
            </DataGrid>
        </DockPanel>

        <Button Grid.Row="0" Name="btnRemoveCust" Command="{Binding
Path=RemoveCustomerCommand}" IsEnabled="{Binding
ElementName=CustDatagrid, Path=SelectedItems.Count}" Content="Διαγραφή
```

```
Πελάτη" HorizontalAlignment="Center" VerticalAlignment="Top" Width="120"
Margin="278,5,278,0"  />
   </Grid>
</UserControl>
```

1.4 RegisterCodeView (Προσθήκη κωδικού):

```
<UserControl x:Class="BondTracker.Views.RegisterCodeView"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
     mc:Ignorable="d">
   <Grid >

       <TextBox x:Name="txbTitle" Text="{Binding Path=Title}"
TextWrapping="Wrap" Margin="0,177,284,0" FontFamily="Arial"
HorizontalAlignment="Right" Width="277" Height="22" VerticalAlignment="Top"/>
       <TextBox x:Name="txbUserName" Text="{Binding Path=UserName}"
TextWrapping="Wrap" Margin="0,214,284,0" FontFamily="Arial" Height="23"
VerticalAlignment="Top" HorizontalAlignment="Right" Width="277"/>
       <TextBox x:Name="txbPassword" Text="{Binding Path=Password}"
HorizontalAlignment="Right" Height="23" TextWrapping="Wrap"
VerticalAlignment="Top" Width="277" Margin="0,251,284,0" FontFamily="Arial"/>
       <TextBox x:Name="txbUrl" Text="{Binding Path=Url}" Height="23"
TextWrapping="Wrap" VerticalAlignment="Top" Margin="0,288,284,0"
FontFamily="Arial" HorizontalAlignment="Right" Width="277"/>
       <TextBox x:Name="txbNotes" Text="{Binding Path=Notes}" Height="98"
TextWrapping="Wrap" VerticalAlignment="Top" Margin="0,325,284,0"
FontFamily="Arial" HorizontalAlignment="Right" Width="277"/>

       <Label Content=" Τίτλος" FontWeight="Bold" VerticalAlignment="Top"
Margin="0,177,575,0" Height="32" FontFamily="Arial" HorizontalAlignment="Right"
Width="106"/>
       <Label Content="    Όνομα Χρήστη" FontWeight="Bold"
VerticalAlignment="Top" Margin="0,214,575,0" Height="23" FontFamily="Arial"
HorizontalAlignment="Right" Width="116"/>
       <Label Content=" Κωδικός" FontWeight="Bold" VerticalAlignment="Top"
Margin="0,251,575,0" Height="23" FontFamily="Arial" HorizontalAlignment="Right"
Width="106"/>
       <Label Content=" Url" FontWeight="Bold" VerticalAlignment="Top"
Margin="0,288,576,0" Height="32" FontFamily="Arial" HorizontalAlignment="Right"
Width="105"/>
       <Label Content=" Σημειώσεις" FontWeight="Bold" VerticalAlignment="Top"
Margin="0,325,575,0" Height="23" FontFamily="Arial" HorizontalAlignment="Right"
Width="106"/>
```

```
        <Label Content="Καταχώρηση Κώδικα" HorizontalAlignment="Left"
FontSize="26" FontWeight="Bold" FontFamily="Arial" Margin="82,23,0,0"
Height="46" VerticalAlignment="Top"/>
        <Label Content="Συμπληρώστε τα στοιχεία της εγγραφής και πατήστε"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="82,83,0,0"
FontFamily="Arial" FontSize="14"/>
        <Label Content=""Αποθήκευση"" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="416,83,0,0" FontSize="14" FontWeight="Bold"
FontFamily="Arial"/>

        <Button x:Name="btnSave" Content="Αποθήκευση"  Command="{Binding
Path=SaveCodeCommand}" HorizontalAlignment="Right" VerticalAlignment="Top"
Width="134" Margin="0,535,329,0" FontFamily="Arial" FontSize="14"  />
        <Button Name="btnCancel" Content="Καθαρισμός" Command="{Binding
Path=CancelSaveCodeCommand}" HorizontalAlignment="Right" Width="82"
Margin="0,140,329,0" FontFamily="Arial" Height="20" VerticalAlignment="Top"  />


    </Grid>
</UserControl>
```

1.5 RegisterCustomerView (Προσθήκη πελάτη):

```
<UserControl x:Class="BondTracker.Views.RegisterCustomerView"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d">

    <Grid >

        <TextBox x:Name="txbLastName" Text="{Binding Path=LastName}"
TextWrapping="Wrap" Margin="0,177,329,0" FontFamily="Arial"
HorizontalAlignment="Right" Width="232" Height="23" VerticalAlignment="Top"/>
        <TextBox x:Name="txbFirstName" Text="{Binding Path=FirstName}"
TextWrapping="Wrap" Margin="0,214,329,0" FontFamily="Arial" Height="23"
VerticalAlignment="Top" HorizontalAlignment="Right" Width="232"/>
        <TextBox x:Name="txbEmail" Text="{Binding Path=Email}"
HorizontalAlignment="Right" Height="23" TextWrapping="Wrap"
VerticalAlignment="Top" Width="232" Margin="0,251,329,0" FontFamily="Arial"/>
        <TextBox x:Name="txbLocalPhoneNumb" Text="{Binding
Path=LocalPhoneNumb}" Height="23" TextWrapping="Wrap"
VerticalAlignment="Top" Margin="0,288,329,0" FontFamily="Arial"
HorizontalAlignment="Right" Width="232"/>
```

```
        <TextBox x:Name="txbCellPhoneNumb" Text="{Binding
Path=CellPhoneNumb}" Height="23" TextWrapping="Wrap"
VerticalAlignment="Top" Margin="0,325,329,0" FontFamily="Arial"
HorizontalAlignment="Right" Width="232"/>
        <TextBox x:Name="txbAddress" Text="{Binding Path=Address}" Height="23"
TextWrapping="Wrap" VerticalAlignment="Top" Margin="0,362,329,0"
FontFamily="Arial" HorizontalAlignment="Right" Width="232"/>
        <TextBox x:Name="txbCity" Height="23" Text="{Binding Path=City}"
TextWrapping="Wrap" VerticalAlignment="Top" Margin="0,399,329,0"
FontFamily="Arial" HorizontalAlignment="Right" Width="232"/>
        <TextBox x:Name="txbZipCode" Text="{Binding Path=ZipCode}" Height="23"
TextWrapping="Wrap" VerticalAlignment="Top" Margin="0,436,329,0"
FontFamily="Arial" HorizontalAlignment="Right" Width="232"/>
        <TextBox x:Name="txbCountry" Text="{Binding Path=Country}" Height="23"
TextWrapping="Wrap" VerticalAlignment="Top" Margin="0,473,329,0"
FontFamily="Arial" HorizontalAlignment="Right" Width="232"/>

        <Label Content="Επώνυμο" FontWeight="Bold" VerticalAlignment="Top"
Margin="-9,177,576,0" Height="32" FontFamily="Arial"
HorizontalAlignment="Right" Width="115"/>
        <Label Content="Όνομα" FontWeight="Bold" VerticalAlignment="Top"
Margin="-9,214,576,0" Height="23" FontFamily="Arial"
HorizontalAlignment="Right" Width="115"/>
        <Label Content="E-mail" FontWeight="Bold" VerticalAlignment="Top"
Margin="-9,251,576,0" Height="23" FontFamily="Arial"
HorizontalAlignment="Right" Width="115"/>
        <Label Content="Σταθερό τηλέφωνο" FontWeight="Bold"
VerticalAlignment="Top" Margin="-11,288,576,0" Height="32" FontFamily="Arial"
HorizontalAlignment="Right" Width="117"/>
        <Label Content="Κινητό Τηλέφωνο" FontWeight="Bold"
VerticalAlignment="Top" Margin="-9,325,575,0" Height="23" FontFamily="Arial"
HorizontalAlignment="Right" Width="116"/>
        <Label Content="Διεύθυνση" FontWeight="Bold" VerticalAlignment="Top"
Margin="-9,362,585,0" Height="23" FontFamily="Arial"
HorizontalAlignment="Right" Width="106"/>
        <Label Content="Πόλη" FontWeight="Bold" VerticalAlignment="Top"
Margin="-9,399,585,0" Height="23" FontFamily="Arial"
HorizontalAlignment="Right" Width="106"/>
        <Label Content="T.K." FontWeight="Bold" VerticalAlignment="Top" Margin="-
9,436,585,0" Height="25" FontFamily="Arial" HorizontalAlignment="Right"
Width="106"/>
        <Label Content="Χώρα" FontWeight="Bold" VerticalAlignment="Top"
Margin="-9,473,585,0" Height="33" FontFamily="Arial"
HorizontalAlignment="Right" Width="106"/>
        <Label Content="Καταχώρηση Πελάτη" HorizontalAlignment="Left"
FontSize="26" FontWeight="Bold" FontFamily="Arial" Margin="82,23,0,0"
Height="46" VerticalAlignment="Top"/>
        <Label Content="Συμπληρώστε τα στοιχεία του πελάτη και πατήστε"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="82,83,0,0"
FontFamily="Arial" FontSize="14"/>
```

```
        <Label Content=""Αποθήκευση"" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="401,83,0,0" FontSize="14" FontWeight="Bold"
FontFamily="Arial"/>
        <Button x:Name="btnSave" Content="Αποθήκευση"  Command="{Binding
Path=SaveCustomerCommand}" HorizontalAlignment="Right"
VerticalAlignment="Top" Width="134" Margin="0,535,329,0" FontFamily="Arial"
FontSize="14"  />
        <Button Name="btnCancel" Content="Καθαρισμός" Command="{Binding
Path=CancelSaveCustomerCommand}" HorizontalAlignment="Right" Width="82"
Margin="0,140,329,0" FontFamily="Arial" Height="20" VerticalAlignment="Top"  />


    </Grid>
</UserControl>
```

2. Όψεις Μοντέλο (View Models):

2.1 BondsViewModel:

```
using System.Windows;
using System.Threading.Tasks;
using System.Threading;
using System.Data;
using BondTracker.Common;
using Models;
using System.Linq;
using System.Collections.ObjectModel;
using System;
using System.Windows.Input;
using System.Windows.Controls;
using System.Windows.Data;
using System.Collections.Generic;
using System.Text;
using BondTracker.Views;
using GalaSoft.MvvmLight.Command;
using System.ComponentModel;
using System.Windows.Documents;


namespace BondTracker.ViewModels
{
    public class BondsViewModel : ViewModelBase, IPageViewModel
    {
```

```
        public BondsViewModel()
        {

            CalculateValues();
            InitializeBondsViewCommands();
        }
        private void CalculateValues()
        {

            try
            {
                var bonds = new List<Bond>();

                using (var dataContext = new Entities())
                {
                    var rawDataList = new System.Collections.Generic.List<RawData>();
                    rawDataList = dataContext.RawDatas.ToList();
                    rawData = new ObservableCollection<RawData>(rawDataList);
                }
                foreach (var mrawData in rawData)
                {
                    if(mrawData.Exchange !=null)
                    {
                        var mBond = new Bond();

                        mBond.Country = mrawData.Country;
                        mBond.IndustrySector = mrawData.IndustrySector;
                        mBond.ASize = mrawData.ASize;
                        mBond.BSize = mrawData.BSize;
                        mBond.BondId = mrawData.Id;
                        mBond.Exchange = mrawData.Exchange;
                        mBond.Isin = mrawData.Isin;
                        mBond.Name = mrawData.Name;
                        mBond.IssueDate = mrawData.IssueDate;
                        mBond.MatureDate = mrawData.MatureDate;
                        mBond.Rating = Rating(mrawData.MoodysRating,
mrawData.FitchsRating, mrawData.SpRating, mrawData.MoodysRatingDate,
mrawData.FitchsRatingDate, mrawData.SpRatingDate);
                        mBond.DenominationMinimum =
mrawData.DenominationMinimum;
                        mBond.DenominationIncrement =
mrawData.DenominationIncrement;
                        mBond.Bid = mrawData.Bid;
                        mBond.Ask = mrawData.Ask;
                        mBond.Spread = mBond.Ask - mBond.Bid;
                        mBond.Ccy = mrawData.Ccy;
                        mBond.Commision = Commision(mBond.Rating, mBond.Years,
mrawData.MoodysRating, mrawData.FitchsRating, mrawData.SpRating);
                        mBond.Coupon = mrawData.Coupon;
```

```
                mBond.CurrentDate = CurrentDate();

                bonds.Add(mBond);
            }
            }

        finishedBonds = new ObservableCollection<Bond>(bonds);
        NumberOfRows = finishedBonds.Count.ToString();

    }
    catch {

    }
}
//Synartiseis aparaitites gia ton ypologismo ton prakseon...

public DateTime CurrentDate()
{
    DateTime mCurrentDate=new DateTime();

    switch(DateTime.Today.DayOfWeek)
    {
        case DayOfWeek.Monday:
            mCurrentDate=DateTime.Today.AddDays(3).Date;
            break;
        case DayOfWeek.Tuesday:
            mCurrentDate=DateTime.Today.AddDays(3).Date;
            break;
        case DayOfWeek.Wednesday:
            mCurrentDate=DateTime.Today.AddDays(5).Date;
            break;
        case DayOfWeek.Thursday:
            mCurrentDate=DateTime.Today.AddDays(5).Date;
            break;
        case DayOfWeek.Friday:
            mCurrentDate=DateTime.Today.AddDays(5).Date;
            break;
        case DayOfWeek.Saturday:
            mCurrentDate = DateTime.Today.AddDays(4).Date;
            break;
        case DayOfWeek.Sunday:
            mCurrentDate = DateTime.Today.AddDays(3).Date;
            break;
    }
    return mCurrentDate;
}

    public string Rating(string moodysRating, string fitchsRating, string spRating,
DateTime moodysRatingDate, DateTime fitchsRatingDate, DateTime
spRatingDate)
```

```
        {
            if ((fitchsRatingDate.CompareTo(moodysRatingDate) < 0) &&
(spRatingDate.CompareTo(moodysRatingDate) < 0))
                rating = moodysRating;
            else if ((moodysRatingDate.CompareTo(fitchsRatingDate) < 0) &&
(spRatingDate.CompareTo(fitchsRatingDate) < 0))
                rating = fitchsRating;
            else if ((moodysRatingDate.CompareTo(spRatingDate) < 0) &&
(fitchsRatingDate.CompareTo(spRatingDate) < 0))
                rating = spRating;
            else
                rating = moodysRating;


            return rating;
        }


        public decimal Commision(string rating, decimal years, string moodysRating,
string fitchsRating, string spRating)
        {
            decimal  ratingValue=new decimal();

            if (rating == moodysRating)
            {
                if (rating.Contains("Aaa"))
                {
                    if (years < 2)
                        ratingValue = 0.16m;
                    else if (years >= 2 && years < 3)
                        ratingValue = 0.20m;
                    else if (years >= 3 && years < 4)
                        ratingValue = 0.20m;
                    else if (years >= 4 && years < 5)
                        ratingValue = 0.24m;
                    else if (years >= 5 && years < 6)
                        ratingValue = 0.32m;
                    else if (years >= 6 && years < 7)
                        ratingValue = 0.32m;
                    else if (years >= 7 && years < 8)
                        ratingValue = 0.32m;
                    else if (years >= 8 && years < 9)
                        ratingValue = 0.32m;
                    else if (years >= 9 && years < 10)
                        ratingValue = 0.32m;
                    else
                        ratingValue = 0.40m;
                }
```

```
            else if ((rating.Contains("Aa1")) || (rating.Contains("Aa2")) ||
(rating.Contains("Aa3")) || (rating.Contains("A1")) || (rating.Contains("A2")) ||
(rating.Contains("A3")) || (rating.Contains("Baa1")) || (rating.Contains("Baa2")) ||
(rating.Contains("Baa3")))
            {
                if (years < 2)
                    ratingValue = 0.26m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.33m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.33m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.39m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.39m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.52m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.52m;
                else if (years >= 8 && years < 9)
                    ratingValue = 0.52m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.52m;
                else
                    ratingValue = 0.66m;
            }

            else if ((rating.Contains("Ba1")) || (rating.Contains("Ba2")) ||
(rating.Contains("Ba3")) || (rating.Contains("B1")) || (rating.Contains("B2")) ||
(rating.Contains("B3")) || (rating.Contains("Caa1")) || (rating.Contains("Caa2")) ||
(rating.Contains("Caa3")) || (rating.Contains("DD")) || (rating.Contains("D")))
            {
                if (years < 2)
                    ratingValue = 0.4m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.5m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.5m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.6m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.6m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.8m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.8m;
                else if (years >= 8 && years < 9)
                    ratingValue = 0.8m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.8m;
```

```
                else
                    ratingValue = 1m;
            }
            else if ((!rating.Contains("Aaa")) && (!rating.Contains("Aa1")) &&
(!rating.Contains("Aa2")) && (!rating.Contains("Aa3")) && (!rating.Contains("A1"))
&& (!rating.Contains("A2")) && (!rating.Contains("A3")) &&
(!rating.Contains("Baa1")) && (!rating.Contains("Baa2")) &&
(!rating.Contains("Baa3")) && (!rating.Contains("Ba1")) &&
(!rating.Contains("Ba2")) && (!rating.Contains("Ba3")) && (!rating.Contains("B1"))
&& (!rating.Contains("B2")) && (!rating.Contains("B3")) &&
(!rating.Contains("Caa1")) && (!rating.Contains("Caa2")) &&
(!rating.Contains("Caa3")) && (!rating.Contains("DD")) && (!rating.Contains("D")))
            {
                if (years < 2)
                    ratingValue = 0.4m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.5m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.5m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.6m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.6m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.8m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.8m;
                else if (years >= 8 && years < 9)
                    ratingValue = 0.8m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.8m;
                else
                    ratingValue = 1m;
            }
        }
        if (rating == spRating)
        {
            if (rating.Contains("AAA"))
            {
                if (years < 2)
                    ratingValue = 0.16m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.20m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.20m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.24m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.32m;
                else if (years >= 6 && years < 7)
```

```
                ratingValue = 0.32m;
            else if (years >= 7 && years < 8)
                ratingValue = 0.32m;
            else if (years >= 8 && years < 9)
                ratingValue = 0.32m;
            else if (years >= 9 && years < 10)
                ratingValue = 0.32m;
            else
                ratingValue = 0.40m;
        }

        else if ((rating.Contains("AA+")) || (rating.Contains("AA")) ||
(rating.Contains("AA-")) || (rating.Contains("A+")) || (rating.Contains("A")) ||
(rating.Contains("A-")) || (rating.Contains("BBB+")) || (rating.Contains("BBB")) ||
(rating.Contains("BBB-")))
        {
            if (years < 2)
                ratingValue = 0.26m;
            else if (years >= 2 && years < 3)
                ratingValue = 0.33m;
            else if (years >= 3 && years < 4)
                ratingValue = 0.33m;
            else if (years >= 4 && years < 5)
                ratingValue = 0.39m;
            else if (years >= 5 && years < 6)
                ratingValue = 0.39m;
            else if (years >= 6 && years < 7)
                ratingValue = 0.52m;
            else if (years >= 7 && years < 8)
                ratingValue = 0.52m;
            else if (years >= 8 && years < 9)
                ratingValue = 0.52m;
            else if (years >= 9 && years < 10)
                ratingValue = 0.52m;
            else
                ratingValue = 0.66m;
        }

        else if ((rating.Contains("BB+")) || (rating.Contains("BB")) ||
(rating.Contains("BB-")) || (rating.Contains("B+")) || (rating.Contains("B")) ||
(rating.Contains("B-")) || (rating.Contains("CCC+")) || (rating.Contains("CCC")) ||
(rating.Contains("CCC-")) || (rating.Contains("CCC")) || (rating.Contains("CCC-")) ||
(rating.Contains("D/SD")))
        {
            if (years < 2)
                ratingValue = 0.4m;
            else if (years >= 2 && years < 3)
                ratingValue = 0.5m;
            else if (years >= 3 && years < 4)
                ratingValue = 0.5m;
```

```
                else if (years >= 4 && years < 5)
                    ratingValue = 0.6m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.6m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.8m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.8m;
                else if (years >= 8 && years < 9)
                    ratingValue = 0.8m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.8m;
                else
                    ratingValue = 1m;
            }
            else if ((!rating.Contains("AAA")) && (!rating.Contains("AA+")) &&
(!rating.Contains("AA")) && (!rating.Contains("AA-")) && (!rating.Contains("A+"))
&& (!rating.Contains("A")) && (!rating.Contains("A-")) && (!rating.Contains("BB+"))
&& (!rating.Contains("BBB")) && (!rating.Contains("BBB-")) &&
(!rating.Contains("BB+")) && (!rating.Contains("BB")) && (!rating.Contains("BB-"))
&& (!rating.Contains("B+")) && (!rating.Contains("B")) && (!rating.Contains("B-"))
&& (!rating.Contains("CCC+")) && (!rating.Contains("CCC")) &&
(!rating.Contains("CCC-")) && (!rating.Contains("D/SD")))
            {
                if (years < 2)
                    ratingValue = 0.4m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.5m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.5m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.6m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.6m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.8m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.8m;
                else if (years >= 8 && years < 9)
                    ratingValue = 0.8m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.8m;
                else
                    ratingValue = 1m;
            }
        }
        if (rating == fitchsRating)
        {
            if (rating.Contains("AAA"))
            {
```

```
            if (years < 2)
                ratingValue = 0.16m;
            else if (years >= 2 && years < 3)
                ratingValue = 0.20m;
            else if (years >= 3 && years < 4)
                ratingValue = 0.20m;
            else if (years >= 4 && years < 5)
                ratingValue = 0.24m;
            else if (years >= 5 && years < 6)
                ratingValue = 0.32m;
            else if (years >= 6 && years < 7)
                ratingValue = 0.32m;
            else if (years >= 7 && years < 8)
                ratingValue = 0.32m;
            else if (years >= 8 && years < 9)
                ratingValue = 0.32m;
            else if (years >= 9 && years < 10)
                ratingValue = 0.32m;
            else
                ratingValue = 0.40m;
        }

        else if ((rating.Contains("AA+")) || (rating.Contains("Aa3")) ||
(rating.Contains("AA-")) || (rating.Contains("A+")) || (rating.Contains("A")) ||
(rating.Contains("A-")) || (rating.Contains("BBB+")) || (rating.Contains("Baa3")) ||
(rating.Contains("BBB-")))
        {
            if (years < 2)
                ratingValue = 0.26m;
            else if (years >= 2 && years < 3)
                ratingValue = 0.33m;
            else if (years >= 3 && years < 4)
                ratingValue = 0.33m;
            else if (years >= 4 && years < 5)
                ratingValue = 0.39m;
            else if (years >= 5 && years < 6)
                ratingValue = 0.39m;
            else if (years >= 6 && years < 7)
                ratingValue = 0.52m;
            else if (years >= 7 && years < 8)
                ratingValue = 0.52m;
            else if (years >= 8 && years < 9)
                ratingValue = 0.52m;
            else if (years >= 9 && years < 10)
                ratingValue = 0.52m;
            else
                ratingValue = 0.66m;
        }
```

```
            else if ((rating.Contains("BB+")) || (rating.Contains("Ba3")) ||
(rating.Contains("BB-")) || (rating.Contains("B+")) || (rating.Contains("B")) ||
(rating.Contains("B-")) || (rating.Contains("CCC+")) || (rating.Contains("CCC")) ||
(rating.Contains("CCC-")) || (rating.Contains("DDD")))
            {
                if (years < 2)
                    ratingValue = 0.4m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.5m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.5m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.6m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.6m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.8m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.8m;
                else if (years >= 8 && years < 9)
                    ratingValue = 0.8m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.8m;
                else
                    ratingValue = 1m;
            }
            else if ((!rating.Contains("AAA")) && (!rating.Contains("AA+")) &&
(!rating.Contains("Aa3")) && (!rating.Contains("AA-")) && (!rating.Contains("A+"))
&& (!rating.Contains("A")) && (!rating.Contains("A-")) &&
(!rating.Contains("BBB+")) && (!rating.Contains("Baa3")) &&
(!rating.Contains("BBB-")) && (!rating.Contains("BB+")) &&
(!rating.Contains("Ba3")) && (!rating.Contains("BB-")) && (!rating.Contains("B+"))
&& (!rating.Contains("B")) && (!rating.Contains("B-")) &&
(!rating.Contains("CCC+")) && (!rating.Contains("CCC")) &&
(!rating.Contains("CCC-")) && (!rating.Contains("DDD")))
            {
                if (years < 2)
                    ratingValue = 0.4m;
                else if (years >= 2 && years < 3)
                    ratingValue = 0.5m;
                else if (years >= 3 && years < 4)
                    ratingValue = 0.5m;
                else if (years >= 4 && years < 5)
                    ratingValue = 0.6m;
                else if (years >= 5 && years < 6)
                    ratingValue = 0.6m;
                else if (years >= 6 && years < 7)
                    ratingValue = 0.8m;
                else if (years >= 7 && years < 8)
                    ratingValue = 0.8m;
```

```
                else if (years >= 8 && years < 9)
                    ratingValue = 0.8m;
                else if (years >= 9 && years < 10)
                    ratingValue = 0.8m;
                else
                    ratingValue = 1m;
                }
            }

            return ratingValue;
        }



        protected bool ValidDate(int dateYear,int dateMonth,int dateDay)
        {
            int
daysInMonthDate=System.DateTime.DaysInMonth(dateYear,dateMonth);

            if (dateDay <= daysInMonthDate)
                return true;
            else
                return false;
        }


        private void InitializeBondsViewCommands()
        {
            RefreshBondsViewCommand = new RelayCommand(CalculateValues);
        }



        #region Commands
        public ICommand RefreshBondsViewCommand
        {
            get;
            private set;
        }




        #endregion


        public ObservableCollection<Bond> finishedBonds
        {
            get;
```

```
            set;
        }

        public string NumberOfRows
        {
            get;
            set;
        }

        public ObservableCollection<RawData> rawData
        {
            get;
            set;
        }


        #region Fields

        private string rating;


        public string Name
        {
            get { return "Bonds View"; }
        }
        public dynamic Data { get; set; }

            #endregion


    }
}
```

2.2 CodesViewModel:

```
using System;
using System.Windows.Input;
using System.Windows.Controls;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BondTracker.Common;
using System.Collections.ObjectModel;
using Models;
using BondTracker.Views;
using GalaSoft.MvvmLight.Command;
using System.ComponentModel.Composition;
```

```
namespace BondTracker.ViewModels
{
   public class CodesViewModel : ViewModelBase, IPageViewModel
   {

        #region Constractor

      public CodesViewModel()
      {
         ShowCodesList();
         InitializeCodesCommands();
      }
      #endregion

      #region Commands
      public ICommand RemoveCodeCommand
      {
         get;
         private set;

      }
      public RelayCommand<Code> ModifyCodeCommand
      {
         get;
         private set;
      }

      #endregion

      #region Properties

      public string Name
      {
         get { return "Codes"; }
      }
      public void RemoveCode()
      {
         using (var dataContext = new Entities())
         {
            int ID = SelectedCode.Id;
            Code mCode = dataContext.Codes.First(i => i.Id == ID);
            dataContext.Codes.Remove(mCode);
            dataContext.SaveChanges();
         }
         ShowCodesList();
      }

      public void ShowCodesList()
      {
```

```
            using (var dataContext = new Entities())
            {
                var list = dataContext.Codes.ToList();
                Codes = new ObservableCollection<Code>(list);
            }
            OnPropertyChanged("Codes");
        }


        public void ModifyCode(Code code)
        {
            using (var dataContext = new Entities())
            {
                int ID = SelectedCode.Id;
                Code mCode = dataContext.Codes.First(i => i.Id== ID);
            }
        }

        private void InitializeCodesCommands()
        {
            RemoveCodeCommand = new RelayCommand(RemoveCode);
            ModifyCodeCommand = new RelayCommand<Code>(ModifyCode);
        }

        private Code mSelectedCode;
        public Code SelectedCode
        {
            get { return mSelectedCode; }
            set
            {
                mSelectedCode = value;
                OnPropertyChanged("SelectedCode");
            }
        }

        public ObservableCollection<Code> Codes
        {
            get;
            set;
        }

        #endregion

        //Necessare For the IpageViewModel
        public dynamic Data { get; set; }
    }
}
```

2.3 CustomersViewModel:

```
using System;
using System.Windows.Input;
using System.Windows.Controls;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BondTracker.Common;
using System.Collections.ObjectModel;
using Models;
using BondTracker.Views;
using GalaSoft.MvvmLight.Command;
using System.ComponentModel.Composition;


namespace BondTracker.ViewModels
{
   public class CustomersViewModel : ViewModelBase, IPageViewModel
   {


      #region Constractor

      public CustomersViewModel()
      {

         ShowCustomersList();
         InitializeCustomersCommands();
      }

      #endregion
      #region Fields


      #endregion
      #region Commands
      public ICommand RemoveCustomerCommand
      {
         get;
         private set;

      }
      public RelayCommand<Customer> ModifyCustomerCommand
      {
         get;
         private set;
      }

      #endregion
```

```
        #region Properties

        public string Name
        {
            get { return "Customers"; }
        }
        public void RemoveCustomer()
        {
            using (var dataContext = new Entities())
            {
                int ID = SelectedCustomer.CustomerId;
                Customer cust = dataContext.Customers.First(i => i.CustomerId == ID);
                dataContext.Customers.Remove(cust);
                dataContext.SaveChanges();
            }
            ShowCustomersList();
        }

        public void ShowCustomersList()
        {
            using (var dataContext = new Entities())
            {
                var list = dataContext.Customers.ToList();
                Customers = new ObservableCollection<Customer>(list);
            }
            OnPropertyChanged("Customers");
        }


        public void ModifyCustomer(Customer customer)
        {
            using (var dataContext = new Entities())
            {
                int ID = SelectedCustomer.CustomerId;
                Customer cust = dataContext.Customers.First(i => i.CustomerId == ID);
            }

        }

        private void InitializeCustomersCommands()
        {
            RemoveCustomerCommand = new RelayCommand(RemoveCustomer);
            ModifyCustomerCommand = new
RelayCommand<Customer>(ModifyCustomer);
        }

        private Customer mSelectedCustomer;
        public Customer SelectedCustomer
        {
            get { return mSelectedCustomer; }
```

```
            set
            {
                mSelectedCustomer = value;
                OnPropertyChanged("SelectedCustomer");
            }
        }
        public ObservableCollection<Customer> Customers
        {
            get;
            set;
        }
        public dynamic Data { get; set; }

        #endregion
    }

}
```

2.4 RegisterCodeViewModel:

```
using System.Runtime.CompilerServices;
using BondTracker.Common;
using Models;
using System.Windows.Input;
using GalaSoft.MvvmLight.Command;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections.ObjectModel;
using System.ComponentModel.Composition;

namespace BondTracker.ViewModels
{
    class RegisterCodeViewModel : ViewModelBase, IPageViewModel
    {
        public RegisterCodeViewModel()
        {
            MCode = new Code();

            InitializeRegisterCodeCommands();

        }

        #region Properties

        public Code MCode { set; get; }

        public string Title
```

```
{
    get { return MCode.Title; }
    set
    {
        MCode.Title = value;
        OnPropertyChanged("Title");
    }
}

public string UserName
{
    get { return MCode.UserName; }
    set
    {
        MCode.UserName = value;
        OnPropertyChanged("UserName");
    }
}

public string Password
{
    get { return MCode.Password; }
    set
        {
        MCode.Password = value;
        OnPropertyChanged("Password");
    }
}

public string Url
{
    get { return MCode.Url; }
    set
    {
        MCode.Url = value;
        OnPropertyChanged("Url");
    }
}

public string Notes
{
    get { return MCode.Notes; }
    set
    {
        MCode.Notes = value;
        OnPropertyChanged("Notes");
    }
}

#endregion
```

```
        #region ICommands

        public ICommand SaveCodeCommand
        {
            get;
            private set;
        }

        public ICommand CancelSaveCodeCommand
        {
            get;
            private set;
        }

        #endregion

        private void InitializeRegisterCodeCommands()
        {
            SaveCodeCommand = new RelayCommand(SaveCode, CanSaveCode);
            CancelSaveCodeCommand = new RelayCommand(CancelSaveCode);
        }

        private bool CanSaveCode()
        {
            return !string.IsNullOrEmpty(Title) && !string.IsNullOrEmpty(UserName)
&& !string.IsNullOrEmpty(Password);

        }

        public void SaveCode()
        {
            using (var dataContext = new Entities())
            {
                dataContext.Codes.Add(MCode);
                dataContext.SaveChanges();
            }

            CancelSaveCode();
        }
        public void CancelSaveCode()
        {
            Title = string.Empty;
            UserName = string.Empty;
            Password = string.Empty;
            Url = string.Empty;
            Notes = string.Empty;
        }

        #region Implementation for IPageViewModel
        public string Name
```

```
        {
            get { return "Register Code"; }
        }

        public dynamic Data { get; set; }
        #endregion
    }
}
```

2.5 RegisterCustomerViewModel:

```
using BondTracker.Common;
using Models;
using System.Windows.Input;
using GalaSoft.MvvmLight.Command;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections.ObjectModel;
using System.ComponentModel.Composition;



namespace BondTracker.ViewModels
{
    public class RegisterCustomerViewModel : ViewModelBase, IPageViewModel
    {
        #region Properties
        public Customer Cust
        {
            set;
            get;
        }


        public string FirstName
        {
            get { return Cust.FirstName; }
            set
            {
                Cust.FirstName = value;
                OnPropertyChanged("FirstName");
            }
        }
```

```
public string LastName
{
    get { return Cust.LastName; }
    set
    {
        Cust.LastName = value;
        OnPropertyChanged("LastName");
    }
}
public string Email
{
    get { return Cust.Email; }
    set
    {
        Cust.Email = value;
        OnPropertyChanged("Email");
    }
}
public string LocalPhoneNumb
{
    get { return Cust.LocalPhoneNumb; }
    set
    {
        Cust.LocalPhoneNumb = value;
        OnPropertyChanged("LocalPhoneNumb");
    }
}
public string CellPhoneNumb
{
    get { return Cust.CellPhoneNumb; }
    set
    {
        Cust.CellPhoneNumb = value;
        OnPropertyChanged("CellPhoneNumb");
    }
}

public string Address
{
    get { return Cust.Address; }
    set
    {
        Cust.Address = value;
        OnPropertyChanged("Address");
    }
}
public string City
{
```

```
            get { return Cust.City; }
            set
            {
                Cust.City = value;
                OnPropertyChanged("City");
            }
        }
        public string ZipCode
        {
            get { return Cust.ZipCode; }
            set
            {
                Cust.ZipCode = value;
                OnPropertyChanged("ZipCode");
            }
        }
        public string Country
        {
            get { return Cust.Country; }
            set
            {
                Cust.Country = value;
                OnPropertyChanged("Country");
            }
        }

        #endregion
        #region ICommands

        public ICommand SaveCustomerCommand
        {
            get;
            private set;
        }

        public ICommand CancelSaveCustomerCommand
        {
            get;
            private set;
        }
        #endregion

        private bool CanSaveCustomer()
        {
            return !string.IsNullOrEmpty(LastName) &&
!string.IsNullOrEmpty(FirstName);

        }

        private void InitializeRegisterCustomerCommands()
```

```
        {
            SaveCustomerCommand = new
RelayCommand(SaveCustomer,CanSaveCustomer);
            CancelSaveCustomerCommand = new
RelayCommand(CancelSaveCustomer);
        }

        public void SaveCustomer()
        {
            using (var dataContext = new Entities())
            {

                dataContext.Customers.Add(Cust);
                dataContext.SaveChanges();
            }
            CancelSaveCustomer();
        }
        public void CancelSaveCustomer()
        {
            LastName = string.Empty;
            FirstName = string.Empty;
            Email = string.Empty;
            LocalPhoneNumb = string.Empty;
            CellPhoneNumb = string.Empty;
            City = string.Empty;
            ZipCode = string.Empty;
            Country = string.Empty;
            Address = string.Empty;

        }

        public RegisterCustomerViewModel()
        {
            Cust = new Customer();

            InitializeRegisterCustomerCommands();

        }


        public string Name
        {
            get { return "Register Customer"; }
        }

        public dynamic Data { get; set; }
    }
}
```

3. Μοντέλο (Model):

3.1 BondTracker_Entities.tt:

```
<#@ template language="C#" debug="false" hostspecific="true"#>
<#@ include file="EF.Utility.CS.ttinclude"#><#@
 output extension=".cs"#><#

const string inputFile = @"BondTracker_Entities.edmx";
var textTransform = DynamicTextTransformation.Create(this);
var code = new CodeGenerationTools(this);
var ef = new MetadataTools(this);
var typeMapper = new TypeMapper(code, ef, textTransform.Errors);
var      fileManager = EntityFrameworkTemplateFileManager.Create(this);
var itemCollection = new EdmMetadataLoader(textTransform.Host,
textTransform.Errors).CreateEdmItemCollection(inputFile);
var codeStringGenerator = new CodeStringGenerator(code, typeMapper, ef);

if
(!typeMapper.VerifyCaseInsensitiveTypeUniqueness(typeMapper.GetAllGlobalIte
ms(itemCollection), inputFile))
{
    return string.Empty;
}

WriteHeader(codeStringGenerator, fileManager);

foreach (var entity in
typeMapper.GetItemsToGenerate<EntityType>(itemCollection))
{
    fileManager.StartNewFile(entity.Name + ".cs");
    BeginNamespace(code);
#>
<#=codeStringGenerator.UsingDirectives(inHeader: false)#>
<#=codeStringGenerator.EntityClassOpening(entity)#>
{
<#
    var propertiesWithDefaultValues =
typeMapper.GetPropertiesWithDefaultValues(entity);
    var collectionNavigationProperties =
typeMapper.GetCollectionNavigationProperties(entity);
    var complexProperties = typeMapper.GetComplexProperties(entity);
```

```
    if (propertiesWithDefaultValues.Any() || collectionNavigationProperties.Any() ||
complexProperties.Any())
    {
#>
    public <#=code.Escape(entity)#>()
    {
<#
        foreach (var edmProperty in propertiesWithDefaultValues)
        {
#>
        this.<#=code.Escape(edmProperty)#> =
<#=typeMapper.CreateLiteral(edmProperty.DefaultValue)#>;
<#
        }

        foreach (var navigationProperty in collectionNavigationProperties)
        {
#>
        this.<#=code.Escape(navigationProperty)#> = new
HashSet<<#=typeMapper.GetTypeName(navigationProperty.ToEndMember.GetE
ntityType())#>>();
<#
        }

        foreach (var complexProperty in complexProperties)
        {
#>
        this.<#=code.Escape(complexProperty)#> = new
<#=typeMapper.GetTypeName(complexProperty.TypeUsage)#>();
<#
        }
#>
    }

<#
    }

    var simpleProperties = typeMapper.GetSimpleProperties(entity);
    if (simpleProperties.Any())
    {
        foreach (var edmProperty in simpleProperties)
        {
#>
    <#=codeStringGenerator.Property(edmProperty)#>
<#
        }
    }

    if (complexProperties.Any())
    {
```

```
#>

<#
        foreach(var complexProperty in complexProperties)
        {
#>
    <#=codeStringGenerator.Property(complexProperty)#>
<#
        }
    }

    var navigationProperties = typeMapper.GetNavigationProperties(entity);
    if (navigationProperties.Any())
    {
#>

<#
        foreach (var navigationProperty in navigationProperties)
        {
#>
    <#=codeStringGenerator.NavigationProperty(navigationProperty)#>
<#
        }
    }
#>
}
<#
    EndNamespace(code);
}

foreach (var complex in
typeMapper.GetItemsToGenerate<ComplexType>(itemCollection))
{
    fileManager.StartNewFile(complex.Name + ".cs");
    BeginNamespace(code);
#>
<#=codeStringGenerator.UsingDirectives(inHeader: false, includeCollections:
false)#>
<#=Accessibility.ForType(complex)#> partial class <#=code.Escape(complex)#>
{
<#
    var complexProperties = typeMapper.GetComplexProperties(complex);
    var propertiesWithDefaultValues =
typeMapper.GetPropertiesWithDefaultValues(complex);

    if (propertiesWithDefaultValues.Any() || complexProperties.Any())
    {
#>
    public <#=code.Escape(complex)#>()
    {
```

```
<#
        foreach (var edmProperty in propertiesWithDefaultValues)
        {
#>
        this.<#=code.Escape(edmProperty)#> =
<#=typeMapper.CreateLiteral(edmProperty.DefaultValue)#>;
<#
        }

        foreach (var complexProperty in complexProperties)
        {
#>
        this.<#=code.Escape(complexProperty)#> = new
<#=typeMapper.GetTypeName(complexProperty.TypeUsage)#>();
<#
        }
#>
    }

<#
    }

    var simpleProperties = typeMapper.GetSimpleProperties(complex);
    if (simpleProperties.Any())
    {
        foreach(var edmProperty in simpleProperties)
        {
#>
    <#=codeStringGenerator.Property(edmProperty)#>
<#
        }
    }

    if (complexProperties.Any())
    {
#>

<#
        foreach(var edmProperty in complexProperties)
        {
#>
    <#=codeStringGenerator.Property(edmProperty)#>
<#
        }
    }
#>
}
<#
    EndNamespace(code);
}
```

```
foreach (var enumType in typeMapper.GetEnumItemsToGenerate(itemCollection))
{
    fileManager.StartNewFile(enumType.Name + ".cs");
    BeginNamespace(code);
#>
<#=codeStringGenerator.UsingDirectives(inHeader: false, includeCollections:
false)#>
<#
    if (typeMapper.EnumIsFlags(enumType))
    {
#>
[Flags]
<#
    }
#>
<#=codeStringGenerator.EnumOpening(enumType)#>
{
<#
    var foundOne = false;

    foreach (MetadataItem member in typeMapper.GetEnumMembers(enumType))
    {
        foundOne = true;
#>
    <#=code.Escape(typeMapper.GetEnumMemberName(member))#> =
<#=typeMapper.GetEnumMemberValue(member)#>,
<#
    }

    if (foundOne)
    {
        this.GenerationEnvironment.Remove(this.GenerationEnvironment.Length - 3,
1);
    }
#>
}
<#
    EndNamespace(code);
}

fileManager.Process();

#>
<#+

public void WriteHeader(CodeStringGenerator codeStringGenerator,
EntityFrameworkTemplateFileManager fileManager)
{
    fileManager.StartHeader();
```

```
#>
//------------------------------------------------------------------------------
// <auto-generated>
// <#=GetResourceString("Template_GeneratedCodeCommentLine1")#>
//
// <#=GetResourceString("Template_GeneratedCodeCommentLine2")#>
// <#=GetResourceString("Template_GeneratedCodeCommentLine3")#>
// </auto-generated>
//------------------------------------------------------------------------------
<#=codeStringGenerator.UsingDirectives(inHeader: true)#>
<#+
    fileManager.EndBlock();
}

public void BeginNamespace(CodeGenerationTools code)
{
    var codeNamespace = code.VsNamespaceSuggestion();
    if (!String.IsNullOrEmpty(codeNamespace))
    {
#>
namespace <#=code.EscapeNamespace(codeNamespace)#>
{
<#+
        PushIndent("    ");
    }
}

public void EndNamespace(CodeGenerationTools code)
{
    if (!String.IsNullOrEmpty(code.VsNamespaceSuggestion()))
    {
        PopIndent();
#>
}
<#+
    }
}

public const string TemplateId = "CSharp_DbContext_Types_EF5";

public class CodeStringGenerator
{
    private readonly CodeGenerationTools _code;
    private readonly TypeMapper _typeMapper;
    private readonly MetadataTools _ef;

    public CodeStringGenerator(CodeGenerationTools code, TypeMapper
typeMapper, MetadataTools ef)
    {
        ArgumentNotNull(code, "code");
```

```
        ArgumentNotNull(typeMapper, "typeMapper");
        ArgumentNotNull(ef, "ef");

        _code = code;
        _typeMapper = typeMapper;
        _ef = ef;
    }

    public string Property(EdmProperty edmProperty)
    {
        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} {1} {2} {{ {3}get; {4}set; }}",
            Accessibility.ForProperty(edmProperty),
            _typeMapper.GetTypeName(edmProperty.TypeUsage),
            _code.Escape(edmProperty),
            _code.SpaceAfter(Accessibility.ForGetter(edmProperty)),
            _code.SpaceAfter(Accessibility.ForSetter(edmProperty)));
    }

    public string NavigationProperty(NavigationProperty navigationProperty)
    {
        var endType =
_typeMapper.GetTypeName(navigationProperty.ToEndMember.GetEntityType());
        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} {1} {2} {{ {3}get; {4}set; }}",
            AccessibilityAndVirtual(Accessibility.ForProperty(navigationProperty)),
            navigationProperty.ToEndMember.RelationshipMultiplicity ==
RelationshipMultiplicity.Many ? ("ICollection<" + endType + ">") : endType,
            _code.Escape(navigationProperty),
            _code.SpaceAfter(Accessibility.ForGetter(navigationProperty)),
            _code.SpaceAfter(Accessibility.ForSetter(navigationProperty)));
    }

    public string AccessibilityAndVirtual(string accessibility)
    {
        return accessibility + (accessibility != "private" ? " virtual" : "");
    }

    public string EntityClassOpening(EntityType entity)
    {
        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} {1}partial class {2}{3}",
            Accessibility.ForType(entity),
            _code.SpaceAfter(_code.AbstractOption(entity)),
            _code.Escape(entity),
            _code.StringBefore(" : ", _typeMapper.GetTypeName(entity.BaseType)));
    }
```

```
    public string EnumOpening(SimpleType enumType)
    {
        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} enum {1} : {2}",
            Accessibility.ForType(enumType),
            _code.Escape(enumType),
            _code.Escape(_typeMapper.UnderlyingClrType(enumType)));
        }

    public void WriteFunctionParameters(EdmFunction edmFunction, Action<string,
string, string, string> writeParameter)
    {
        var parameters = FunctionImportParameter.Create(edmFunction.Parameters,
_code, _ef);
        foreach (var parameter in parameters.Where(p => p.NeedsLocalVariable))
        {
            var isNotNull = parameter.IsNullableOfT ?
parameter.FunctionParameterName + ".HasValue" :
parameter.FunctionParameterName + " != null";
            var notNullInit = "new ObjectParameter(\"" +
parameter.EsqlParameterName + "\", " + parameter.FunctionParameterName +
")";
            var nullInit = "new ObjectParameter(\"" + parameter.EsqlParameterName +
"\", typeof(" + parameter.RawClrTypeName + "))";
            writeParameter(parameter.LocalVariableName, isNotNull, notNullInit,
nullInit);
        }
    }

    public string ComposableFunctionMethod(EdmFunction edmFunction, string
modelNamespace)
    {
        var parameters = _typeMapper.GetParameters(edmFunction);

        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} IQueryable<{1}> {2}({3})",
            AccessibilityAndVirtual(Accessibility.ForMethod(edmFunction)),
            _typeMapper.GetTypeName(_typeMapper.GetReturnType(edmFunction),
modelNamespace),
            _code.Escape(edmFunction),
            string.Join(", ", parameters.Select(p => p.FunctionParameterType + " " +
p.FunctionParameterName).ToArray()));
    }

    public string ComposableCreateQuery(EdmFunction edmFunction, string
modelNamespace)
    {
```

```
        var parameters = _typeMapper.GetParameters(edmFunction);

        return string.Format(
            CultureInfo.InvariantCulture,
            "return
((IObjectContextAdapter)this).ObjectContext.CreateQuery<{0}>(\"[{1}].[{2}]({3})\"{4}
);",
            _typeMapper.GetTypeName(_typeMapper.GetReturnType(edmFunction),
modelNamespace),
            edmFunction.NamespaceName,
            edmFunction.Name,
            string.Join(", ", parameters.Select(p => "@" +
p.EsqlParameterName).ToArray()),
            _code.StringBefore(", ", string.Join(", ", parameters.Select(p =>
p.ExecuteParameterName).ToArray())));
    }

    public string FunctionMethod(EdmFunction edmFunction, string
modelNamespace, bool includeMergeOption)
    {
        var parameters = _typeMapper.GetParameters(edmFunction);
        var returnType = _typeMapper.GetReturnType(edmFunction);

        var paramList = String.Join(", ", parameters.Select(p =>
p.FunctionParameterType + " " + p.FunctionParameterName).ToArray());
        if (includeMergeOption)
        {
            paramList = _code.StringAfter(paramList, ", ") + "MergeOption
mergeOption";
        }

        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} {1} {2}({3})",
            AccessibilityAndVirtual(Accessibility.ForMethod(edmFunction)),
            returnType == null ? "int" : "ObjectResult<" +
_typeMapper.GetTypeName(returnType, modelNamespace) + ">",
            _code.Escape(edmFunction),
            paramList);
    }

    public string ExecuteFunction(EdmFunction edmFunction, string
modelNamespace, bool includeMergeOption)
    {
        var parameters = _typeMapper.GetParameters(edmFunction);
        var returnType = _typeMapper.GetReturnType(edmFunction);

        var callParams = _code.StringBefore(", ", String.Join(", ", parameters.Select(p
=> p.ExecuteParameterName).ToArray()));
        if (includeMergeOption)
```

```
        {
            callParams = ", mergeOption" + callParams;
        }

        return string.Format(
            CultureInfo.InvariantCulture,
            "return
((IObjectContextAdapter)this).ObjectContext.ExecuteFunction{0}(\"{1}\"{2});",
            returnType == null ? "" : "<" + _typeMapper.GetTypeName(returnType,
modelNamespace) + ">",
            edmFunction.Name,
            callParams);
    }

    public string DbSet(EntitySet entitySet)
    {
        return string.Format(
            CultureInfo.InvariantCulture,
            "{0} DbSet<{1}> {2} {{ get; set; }}",
            Accessibility.ForReadOnlyProperty(entitySet),
            _typeMapper.GetTypeName(entitySet.ElementType),
            _code.Escape(entitySet));
    }

    public string UsingDirectives(bool inHeader, bool includeCollections = true)
    {
        return inHeader == string.IsNullOrEmpty(_code.VsNamespaceSuggestion())
            ? string.Format(
                CultureInfo.InvariantCulture,
                "{0}using System;{1}" +
                "{2}",
                inHeader ? Environment.NewLine : "",
                includeCollections ? (Environment.NewLine + "using
System.Collections.Generic;") : "",
                inHeader ? "" : Environment.NewLine)
            : "";
    }
}

public class TypeMapper
{
    private const string ExternalTypeNameAttributeName =
@"http://schemas.microsoft.com/ado/2006/04/codegeneration:ExternalTypeName"
;

    private readonly System.Collections.IList _errors;
    private readonly CodeGenerationTools _code;
    private readonly MetadataTools _ef;
```

```
    public TypeMapper(CodeGenerationTools code, MetadataTools ef,
System.Collections.IList errors)
    {
        ArgumentNotNull(code, "code");
        ArgumentNotNull(ef, "ef");
        ArgumentNotNull(errors, "errors");

        _code = code;
        _ef = ef;
        _errors = errors;
    }

    public string GetTypeName(TypeUsage typeUsage)
    {
        return typeUsage == null ? null : GetTypeName(typeUsage.EdmType,
_ef.IsNullable(typeUsage), modelNamespace: null);
    }

    public string GetTypeName(EdmType edmType)
    {
        return GetTypeName(edmType, isNullable: null, modelNamespace: null);
    }

    public string GetTypeName(TypeUsage typeUsage, string modelNamespace)
    {
        return typeUsage == null ? null : GetTypeName(typeUsage.EdmType,
_ef.IsNullable(typeUsage), modelNamespace);
    }

    public string GetTypeName(EdmType edmType, string modelNamespace)
    {
        return GetTypeName(edmType, isNullable: null, modelNamespace:
modelNamespace);
    }

    public string GetTypeName(EdmType edmType, bool? isNullable, string
modelNamespace)
    {
        if (edmType == null)
        {
            return null;
        }

        var collectionType = edmType as CollectionType;
        if (collectionType != null)
        {
            return String.Format(CultureInfo.InvariantCulture, "ICollection<{0}>",
GetTypeName(collectionType.TypeUsage, modelNamespace));
        }
```

```
        var typeName = _code.Escape(edmType.MetadataProperties
                            .Where(p => p.Name == ExternalTypeNameAttributeName)
                            .Select(p => (string)p.Value)
                            .FirstOrDefault())
            ?? (modelNamespace != null && edmType.NamespaceName !=
modelNamespace ?

_code.CreateFullName(_code.EscapeNamespace(edmType.NamespaceName),
_code.Escape(edmType)) :
                _code.Escape(edmType));

        if (edmType is StructuralType)
        {
            return typeName;
        }

        if (edmType is SimpleType)
        {
            var clrType = UnderlyingClrType(edmType);
            if (!IsEnumType(edmType))
            {
                typeName = _code.Escape(clrType);
            }

            return clrType.IsValueType && isNullable == true ?
                String.Format(CultureInfo.InvariantCulture, "Nullable<{0}>", typeName) :
                typeName;
        }

        throw new ArgumentException("edmType");
    }

    public Type UnderlyingClrType(EdmType edmType)
    {
        ArgumentNotNull(edmType, "edmType");

        var primitiveType = edmType as PrimitiveType;
        if (primitiveType != null)
        {
            return primitiveType.ClrEquivalentType;
        }

        if (IsEnumType(edmType))
        {
            return GetEnumUnderlyingType(edmType).ClrEquivalentType;
        }

        return typeof(object);
    }
```

```
    public object GetEnumMemberValue(MetadataItem enumMember)
    {
        ArgumentNotNull(enumMember, "enumMember");

        var valueProperty = enumMember.GetType().GetProperty("Value");
        return valueProperty == null ? null : valueProperty.GetValue(enumMember,
null);
    }

    public string GetEnumMemberName(MetadataItem enumMember)
    {
        ArgumentNotNull(enumMember, "enumMember");

        var nameProperty = enumMember.GetType().GetProperty("Name");
        return nameProperty == null ? null :
(string)nameProperty.GetValue(enumMember, null);
    }

    public System.Collections.IEnumerable GetEnumMembers(EdmType
enumType)
    {
        ArgumentNotNull(enumType, "enumType");

        var membersProperty = enumType.GetType().GetProperty("Members");
        return membersProperty != null
            ?
(System.Collections.IEnumerable)membersProperty.GetValue(enumType, null)
            : Enumerable.Empty<MetadataItem>();
    }

    public bool EnumIsFlags(EdmType enumType)
    {
        ArgumentNotNull(enumType, "enumType");

        var isFlagsProperty = enumType.GetType().GetProperty("IsFlags");
        return isFlagsProperty != null && (bool)isFlagsProperty.GetValue(enumType,
null);
    }

    public bool IsEnumType(GlobalItem edmType)
    {
        ArgumentNotNull(edmType, "edmType");

        return edmType.GetType().Name == "EnumType";
    }

    public PrimitiveType GetEnumUnderlyingType(EdmType enumType)
    {
        ArgumentNotNull(enumType, "enumType");
```

```
        return
(PrimitiveType)enumType.GetType().GetProperty("UnderlyingType").GetValue(en
umType, null);
    }

    public string CreateLiteral(object value)
    {
        if (value == null || value.GetType() != typeof(TimeSpan))
        {
            return _code.CreateLiteral(value);
        }

        return string.Format(CultureInfo.InvariantCulture, "new TimeSpan({0})",
((TimeSpan)value).Ticks);
    }

    public bool VerifyCaseInsensitiveTypeUniqueness(IEnumerable<string> types,
string sourceFile)
    {
        ArgumentNotNull(types, "types");
        ArgumentNotNull(sourceFile, "sourceFile");

        var hash = new
HashSet<string>(StringComparer.InvariantCultureIgnoreCase);
        if (types.Any(item => !hash.Add(item)))
        {
            _errors.Add(
                new CompilerError(sourceFile, -1, -1, "6023",
                    String.Format(CultureInfo.CurrentCulture,
GetResourceString("Template_CaseInsensitiveTypeConflict"))));
            return false;
        }
        return true;
    }

    public IEnumerable<SimpleType>
GetEnumItemsToGenerate(IEnumerable<GlobalItem> itemCollection)
    {
        return GetItemsToGenerate<SimpleType>(itemCollection)
            .Where(e => IsEnumType(e));
    }

    public IEnumerable<T> GetItemsToGenerate<T>(IEnumerable<GlobalItem>
itemCollection) where T: EdmType
    {
        return itemCollection
            .OfType<T>()
            .Where(i => !i.MetadataProperties.Any(p => p.Name ==
ExternalTypeNameAttributeName))
            .OrderBy(i => i.Name);
```

```
    }

    public IEnumerable<string> GetAllGlobalItems(IEnumerable<GlobalItem>
itemCollection)
    {
        return itemCollection
            .Where(i => i is EntityType || i is ComplexType || i is EntityContainer ||
IsEnumType(i))
            .Select(g => GetGlobalItemName(g));
    }

    public string GetGlobalItemName(GlobalItem item)
    {
        if (item is EdmType)
        {
            return ((EdmType)item).Name;
        }
        else
        {
            return ((EntityContainer)item).Name;
        }
    }

    public IEnumerable<EdmProperty> GetSimpleProperties(EntityType type)
    {
        return type.Properties.Where(p => p.TypeUsage.EdmType is SimpleType &&
p.DeclaringType == type);
    }

    public IEnumerable<EdmProperty> GetSimpleProperties(ComplexType type)
    {
        return type.Properties.Where(p => p.TypeUsage.EdmType is SimpleType &&
p.DeclaringType == type);
    }

    public IEnumerable<EdmProperty> GetComplexProperties(EntityType type)
    {
        return type.Properties.Where(p => p.TypeUsage.EdmType is ComplexType
&& p.DeclaringType == type);
    }

    public IEnumerable<EdmProperty> GetComplexProperties(ComplexType type)
    {
        return type.Properties.Where(p => p.TypeUsage.EdmType is ComplexType
&& p.DeclaringType == type);
    }

    public IEnumerable<EdmProperty> GetPropertiesWithDefaultValues(EntityType
type)
    {
```

```
        return type.Properties.Where(p => p.TypeUsage.EdmType is SimpleType &&
p.DeclaringType == type && p.DefaultValue != null);
    }

    public IEnumerable<EdmProperty>
GetPropertiesWithDefaultValues(ComplexType type)
    {
        return type.Properties.Where(p => p.TypeUsage.EdmType is SimpleType &&
p.DeclaringType == type && p.DefaultValue != null);
    }

    public IEnumerable<NavigationProperty> GetNavigationProperties(EntityType
type)
    {
        return type.NavigationProperties.Where(np => np.DeclaringType == type);
    }

    public IEnumerable<NavigationProperty>
GetCollectionNavigationProperties(EntityType type)
    {
        return type.NavigationProperties.Where(np => np.DeclaringType == type &&
np.ToEndMember.RelationshipMultiplicity == RelationshipMultiplicity.Many);
    }

    public FunctionParameter GetReturnParameter(EdmFunction edmFunction)
    {
        ArgumentNotNull(edmFunction, "edmFunction");

        var returnParamsProperty =
edmFunction.GetType().GetProperty("ReturnParameters");
        return returnParamsProperty == null
            ? edmFunction.ReturnParameter
            :
((IEnumerable<FunctionParameter>)returnParamsProperty.GetValue(edmFunctio
n, null)).FirstOrDefault();
    }

    public bool IsComposable(EdmFunction edmFunction)
    {
        ArgumentNotNull(edmFunction, "edmFunction");

        var isComposableProperty =
edmFunction.GetType().GetProperty("IsComposableAttribute");
        return isComposableProperty != null &&
(bool)isComposableProperty.GetValue(edmFunction, null);
    }

    public IEnumerable<FunctionImportParameter> GetParameters(EdmFunction
edmFunction)
    {
```

```
        return FunctionImportParameter.Create(edmFunction.Parameters, _code,
_ef);
    }

    public TypeUsage GetReturnType(EdmFunction edmFunction)
    {
        var returnParam = GetReturnParameter(edmFunction);
        return returnParam == null ? null :
_ef.GetElementType(returnParam.TypeUsage);
    }

    public bool GenerateMergeOptionFunction(EdmFunction edmFunction, bool
includeMergeOption)
    {
        var returnType = GetReturnType(edmFunction);
        return !includeMergeOption && returnType != null &&
returnType.EdmType.BuiltInTypeKind == BuiltInTypeKind.EntityType;
    }
}

public class EdmMetadataLoader
{
    private readonly IDynamicHost _host;
    private readonly System.Collections.IList _errors;

    public EdmMetadataLoader(IDynamicHost host, System.Collections.IList errors)
    {
        ArgumentNotNull(host, "host");
        ArgumentNotNull(errors, "errors");

        _host = host;
        _errors = errors;
    }

    public IEnumerable<GlobalItem> CreateEdmItemCollection(string sourcePath)
    {
        ArgumentNotNull(sourcePath, "sourcePath");

        if (!ValidateInputPath(sourcePath))
        {
            return new EdmItemCollection();
        }

        var schemaElement = LoadRootElement(_host.ResolvePath(sourcePath));
        if (schemaElement != null)
        {
            using (var reader = schemaElement.CreateReader())
            {
                IList<EdmSchemaError> errors;
```

```
            var itemCollection =
MetadataItemCollectionFactory.CreateEdmItemCollection(new[] { reader }, out
errors);

            ProcessErrors(errors, sourcePath);

            return itemCollection;
         }
      }
      return new EdmItemCollection();
   }

   public string GetModelNamespace(string sourcePath)
   {
      ArgumentNotNull(sourcePath, "sourcePath");

      if (!ValidateInputPath(sourcePath))
      {
         return string.Empty;
      }

      var model = LoadRootElement(_host.ResolvePath(sourcePath));
      if (model == null)
      {
         return string.Empty;
      }

      var attribute = model.Attribute("Namespace");
      return attribute != null ? attribute.Value : "";
   }

   private bool ValidateInputPath(string sourcePath)
   {
      if (sourcePath == "$" + "edmxInputFile" + "$")
      {
         _errors.Add(
            new CompilerError(_host.TemplateFile ?? sourcePath, 0, 0,
string.Empty,
               GetResourceString("Template_ReplaceVsItemTemplateToken")));
         return false;
      }

      return true;
   }

   public XElement LoadRootElement(string sourcePath)
   {
      ArgumentNotNull(sourcePath, "sourcePath");
```

```
        var root = XElement.Load(sourcePath, LoadOptions.SetBaseUri |
LoadOptions.SetLineInfo);
        return root.Elements()
            .Where(e => e.Name.LocalName == "Runtime")
            .Elements()
            .Where(e => e.Name.LocalName == "ConceptualModels")
            .Elements()
            .Where(e => e.Name.LocalName == "Schema")
            .FirstOrDefault()
                ?? root;
    }

    private void ProcessErrors(IEnumerable<EdmSchemaError> errors, string
sourceFilePath)
    {
        foreach (var error in errors)
        {
            _errors.Add(
                new CompilerError(
                    error.SchemaLocation ?? sourceFilePath,
                    error.Line,
                    error.Column,
                    error.ErrorCode.ToString(CultureInfo.InvariantCulture),
                    error.Message)
                {
                    IsWarning = error.Severity == EdmSchemaErrorSeverity.Warning
                });
        }
    }

    public bool IsLazyLoadingEnabled(EntityContainer container)
    {
        string lazyLoadingAttributeValue;
        var lazyLoadingAttributeName =
MetadataConstants.EDM_ANNOTATION_09_02 + ":LazyLoadingEnabled";
        bool isLazyLoading;
        return !MetadataTools.TryGetStringMetadataPropertySetting(container,
lazyLoadingAttributeName, out lazyLoadingAttributeValue)
            || !bool.TryParse(lazyLoadingAttributeValue, out isLazyLoading)
            || isLazyLoading;
    }
}

public static void ArgumentNotNull<T>(T arg, string name) where T : class
{
    if (arg == null)
    {
        throw new ArgumentNullException(name);
    }
}
```

```
private static readonly Lazy<System.Resources.ResourceManager>
ResourceManager =
    new Lazy<System.Resources.ResourceManager>(
        () => new
System.Resources.ResourceManager("System.Data.Entity.Design",
typeof(MetadataItemCollectionFactory).Assembly), isThreadSafe: true);

public static string GetResourceString(string resourceName)
{
    ArgumentNotNull(resourceName, "resourceName");

    return ResourceManager.Value.GetString(resourceName, null);
}

      #>
```