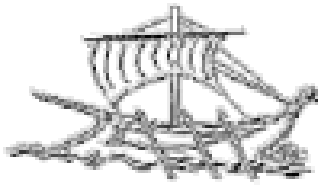


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Web Application για την ενημέρωση του χρήστη περί εκπτώσεων των συνεργαζόμενων supermarket

Λαμπαδάκης Γεώργιος – Λιούτας Χρήστος

Εισηγητής: Δρ. Παναγιώτης Γιαννακόπουλος, Καθηγητής

ΑΘΗΝΑ
Μάρτιος 2018

Web Application για την ενημέρωση του χρήστη περί εκπτώσεων των συνεργαζόμενων supermarket

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Web Application για την ενημέρωση του χρήστη περί εκπτώσεων των συνεργαζόμενων supermarket

Λαμπαδάκης Γεώργιος – Λιούτας Χρήστος
Α.Μ. 40201- 40742

Εισηγητής:

Δρ. Παναγιώτης Γιαννακόπουλος, Καθηγητής

Εξεταστική Επιτροπή:

Δρ. Παναγιώτης Πρεζεράκος, Καθηγητής
Δρ. Δημήτρης Νικολόπουλος, Αναπληρωτής Καθηγητής

Ημερομηνία Εξέτασης: 15/6/2018

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Λιούτας Χρήστος, του Βύρωνα, με αριθμό μητρώου 40742 και Λαμπαδάκης Γεώργιος του Στυλιανού, με αριθμό μητρώου 40201 φοιτητές του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από πολλή προσπάθεια και αναζήτηση γνώσεων και τεχνολογιών, σχετικά με τον προγραμματισμό. Την προσπάθεια αυτή υποστήριξε ο επιβλέπων καθηγητής, ο Δρ. Γιαννακόπουλος, τον οποίο θα θέλαμε να ευχαριστήσουμε.

Ακόμα, ξεχωριστές ευχαριστίες σε κάθε ένα από τα άτομα, και τις οικογένειές μας, που μας στήριξαν στην πτυχιακή αλλά και σε όλη την πορεία μας μέσα στη σχολή.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη μίας web εφαρμογής για την δημιουργία και διάθεση των προσφορών των supermarkets. Παρόμοιες εφαρμογές έχουν δημιουργηθεί από supermarkets για τα δικά τους προϊόντα. Η συγκεκριμένη εφαρμογή έχει ως σκοπό να φέρει τις πληροφορίες όλων των supermarkets σε ένα μέρος, ώστε να διευκολύνει τον καταναλωτή στην εύρεση των προϊόντων που έχει ανάγκη. Λειτουργεί σε όλες τις συσκευές, smartphones, tablets, υπολογιστές, καθώς οι οθόνες της είναι responsive και προσαρμόζονται άμεσα σε κάθε μέγεθος οθόνης.

ABSTRACT

The present thesis concerns the development of a web application which provides the offers that the supermarkets make about their products. These offers are available for everyone to see and the users are able to search through them in order to find what they need. Some supermarkets have made similar applications for their own products but the purpose of this application is to collect all the offers from every willing supermarket and provide them to the users. The responsive function has made this web application easily adjustable to every screen size so the user can access it from smartphones, tablets and PCs.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος Εικόνων

Κατάλογος Διαγραμμάτων

1. ΕΙΣΑΓΩΓΙΚΑ ΣΤΟΙΧΕΙΑ

1.1 Ιστορική Αναδρομή

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μίας web εφαρμογής για την άμεση εύρεση εκπτώσεων σε προϊόντα των supermarkets. Επιπλέον, το κάθε super market θα έχει το δικαίωμα, ανά πάσα στιγμή, να βάλει κάποιο νέο προϊόν με έκπτωση ή να τροποποιήσει κάποιο άλλο δικό του. Αυτό θα δώσει μια ελευθερία στα super market καθώς θα μπορούν να ανακοινώνουν τις προσφορές τους οποιαδήποτε στιγμή, χωρίς να πρέπει να περιμένουν την εκτύπωση κάποιου φυλλαδίου.

1.2 Νομικά στοιχεία

Η web εφαρμογή υλοποιήθηκε με την λειτουργικότητα πως κάθε supermarket θα συνδέετε με τα στοιχεία που θα του έχουν δοθεί και θα μπορεί να καταχωρεί μόνο του τα προϊόντα του που είναι σε έκπτωση. Ο λόγος που δεν τραβάει πληροφορίες από τα supermarket είναι ότι ενδεχομένως να ήταν παράνομο καθώς θα γινόταν χρήση της ονομασίας των supermarkets αλλά και των προϊόντων, χωρίς την συγκατάθεσή τους. Επιπλέον, σε περίπτωση εισαγωγής λανθασμένων δεδομένων θα υπήρχε κίνδυνος για κατηγορίες ψευδών στοιχείων ή προώθησης των ανταγωνιστικών προϊόντων.

1.3 Εργαλεία Υλοποίησης

Για την υλοποίηση μιας web εφαρμογής πρέπει να γίνει επιλογή κάποιων εργαλείων που θα χρησιμοποιηθούν. Επιγραμματικά αυτά είναι τα παρακάτω:

- Server
- Γλώσσα προγραμματισμού
- Integrated development environment
- Framework
- Βάση δεδομένων
- Πρόγραμμα διαχείρισης της βάσης δεδομένων

- Version control

1.3.1 Server

Η πρώτη προϋπόθεση, για μια web εφαρμογή, είναι να υπάρχει μια θέση σε έναν server όπου θα τοποθετηθεί. Στην παρούσα πτυχιακή, έγινε χρήση μιας δωρεάν υπηρεσίας που παρέχει στους εγγεγραμμένους χρήστες της έναν χώρο για την τοποθέτηση της web εφαρμογής και έναν ξεχωριστό χώρο για την βάση δεδομένων. Η διεύθυνση της εφαρμογής είναι: http://marketapp.gear.host/

1.3.2 Γλώσσα Προγραμματισμού

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η C# της Microsoft. Η C# είναι μία Object-Oriented γλώσσα προγραμματισμού που δημιούργησε η Microsoft μέσα από την πλατφόρμα .NET, τον Ιούλιο του 2000. Αργότερα αναγνωρίστηκε απο την Ecma και την ISO και επίσημα. Η τελευταία έκδοση είναι η 7.2 που βγήκε το 2017. Μέσα στη πάροδο του χρόνου και τις πολλές εκδόσεις της, στην C# έχουν προστεθεί πολλά στοιχεία όπως είναι η LINQ, ασύγχρονες μέθοδοι, Auto property initializers. Η σύνταξη της C# μοιάζει με εκείνη της C, C++ και Java. Πιο συγκεκριμένα, ένα παράδειγμα είναι η χρήση του ";" στο τέλος κάθε έκφρασης, καθώς και τα "{}" που δηλώνουν την ομαδοποίηση κάποιων εκφράσεων που ανήκουν συνήθως σε κάποια μέθοδο.

Η επιλογή της συγκεκριμένης γλώσσας έγινε για την αξιοποίηση της αρχιτεκτονικής του Model-View-Controller. Το MVC, συντομογραφία του Model-View-Controller, είναι ένα μοντέλο αρχιτεκτονικής λογισμικού που διαιρεί την εφαρμογή σε τρία μέρη που είναι συνδεδεμένα μεταξύ τους. Ο χωρισμός αυτός σε μέρη γίνεται ώστε οι πληροφορίες που θα μεταφερθούν από τη βάση να μην είναι άμεσα συνδεδεμένες με αυτές που θα εμφανιστούν στον τελικό χρήστη. Το αντικείμενο Model είναι αυτό που δέχεται και αποθηκεύει τα δεδομένα της εφαρμογής. Το View είναι το αντικείμενο που φροντίζει για την παρουσίαση των δεδομένων, του Model δηλαδή, στον χρήστη της εφαρμογής. Τέλος, το αντικείμενο Controller είναι αυτό που είναι υπεύθυνο για την φόρτωση των δεδομένων στο Model και την αλληλεπίδραση με το View. Αυτό το μοντέλο αρχιτεκτονικής, αν και

ξεκίνησε για desktop εφαρμογές, έγινε πολύ δημοφιλές για την σχεδίαση web εφαρμογών. Η C# δεν είναι η μόνη γλώσσα που αξιοποιεί την αρχιτεκτονική του MVC. Άλλα παραδείγματα τέτοιων γλωσσών είναι η Java, η Ruby και η PHP.

Για την εμφάνιση των σελίδων έγινε χρήση του Bootstrap. Το Bootsrap είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα για την σχεδίαση ιστοσελίδων και δικτυακών εφαρμογών. Είναι ένα εργαλείο που υποστηρίζει την responsive τεχνολογία, κάνοντας πολύ εύκολη την διάθεση της σελίδας σε κάθε τύπο συσκευής, αναπροσαρμόζοντας τα μεγέθη των στοιχείων της κάθε σελίδας ώστε να ταιριάζουν με το μέγεθος της οθόνης που προβάλλονται. Διαθέτει φόρμες, κουμπιά και εργαλεία γραμμένα σε HTML και CSS.

1.3.3 Integrated development environment

Αφού γίνει επιλογή της γλώσσας προγραμματισμού θα πρέπει στη συνέχεια να γίνει και η επιλογή ενός integrated development environment, ή IDE για συντομία. Το IDE είναι ένα λογισμικό που βοηθάει στην δημιουργία ενός προγράμματος. Συνήθως περιέχουν έναν επεξεργαστή για τον πηγαίο κώδικα, debbuger, build automation tools και οι πιο σύγχρονοι έχουν και αυτόματη παραγωγή κώδικα ώστε να κάνουν πιο γρήγορη τη δουλειά του προγραμματιστή. Παράδειγμα IDE αποτελεί το NetBeans και το Eclipse, που είναι κυρίως για Java.

Ως IDE, χρησιμοποιήθηκε το Visual Studio, της Microsoft. Είναι το πιο δημοφιλές εργαλείο για την C# και το Community Edition δόθηκε δωρεάν από την Microsoft στο κοινό. Το Visual Studio 2015, που είναι η έκδοση του χρησιμοποιήθηκε, βγήκε στην αγορά τον Ιούλιο του 2015. Το 1997 ήταν η χρονιά που η Microsoft έβγαλε το πρώτο Visual Studio, με όνομα Visual Studio 97. Μέσα στα χρόνια πολλά νέα χαρακτηριστικά προστέθηκαν στο Visual Studio όπως το .NET Framework, Entity Framework, ASP.NET, IntelliSense και εργαλεία Code Analysis.

1.3.4 Framework

Το framework είναι ένα σύνολο μεθόδων, κώδικα και εργαλείων που έχουν δημιουργηθεί με σκοπό την πιο γρήγορη και καλύτερα δομημένη ανάπτυξη προγραμμάτων και εφαρμογών. Το Entity Framework είναι αυτό που θα χρησιμοποιηθεί για την πτυχιακή εργασία, καθώς είναι με το Visual Studio που όπως αναφέρθηκε επιλέχθηκε για την ανάπτυξη της web εφαρμογής. Το Entity Framework ήταν μέρος του .Net Framework αλλά από την έκδοση 6 και μετά έγιναν δύο διαφορετικά frameworks. Η έκδοση 6 του Entity Framework κυκλοφόρησε στις 17 Οκτωβρίου 2013.

Το συγκεκριμένο framework επιτρέπει την δημιουργία μοντέλων γράφοντας κώδικα ή μέσω ενός γραφικού περιβάλλοντος. Στη συνέχεια μπορούν να γίνουν αλλαγές σε μία υπάρχουσα βάση δεδομένων ή να δημιουργηθεί από την αρχή μία νέα, το λεγόμενο Code First. Στην συγκεκριμένη web εφαρμογή έγινε χρήση μίας άλλης δυνατότητας του Entity Framework που ονομάζεται Database First. Με αυτήν την τεχνική, αφού δημιουργηθεί η βάση δεδομένων, τότε μέσα από το Visual Studio γίνεται η αυτόματη παραγωγή των αντίστοιχων κλάσεων που θα είναι συνδεδεμένες με του πίνακες της βάσης δεδομένων.

1.3.5 Βάση δεδομένων

Μία βάση δεδομένων είναι το σύνολο πληροφοριών που είναι κατάλληλα οργανωμένο ώστε να υπάρχει εύκολη πρόσβαση σε αυτό καθώς και παραμετροποίηση. Οι πληροφορίες, ή αλλιώς δεδομένα, που υπάρχουν σε μία βάση είναι χωρισμένα σε πίνακες και αυτοί με τη σειρά τους σε στήλες και γραμμές. Αυτό γίνεται για να υπάρχει μια σωστή κατανομή των δεδομένων και να υπάρχει ευκολία στην εύρεσή τους. Οι πρώτες βάσεις αναπτύχθηκαν την δεκαετία του 1960. Μέσα στη δεκαετία του 1980 οι βάσεις δεδομένων πήραν την μορφή object-oriented databases και πλέον έχουν γίνει cloud databases.

1.3.6 Πρόγραμμα διαχείρισης της βάσης δεδομένων

Υπάρχουν εργαλεία που βοηθούν με την επεξεργασία των δεδομένων που υπάρχουν σε μία βάση. Τα εργαλεία αυτά διαφέρουν ανάλογα με την επιλογή της

βάσης δεδομένων που θα μπει στην εφαρμογή. Σε αυτήν την περίπτωση, η βάση δεδομένων που επιλέχθηκε είναι η MSSQL (Microsoft SQL) και το εργαλείο για την επεξεργασία της είναι το Microsoft SQL Server Management Studio 17. Για να γίνει η σύνδεση στη βάση χρειάζεται η συμπλήρωση των κατάλληλων στοιχείων, τα οποία μπορούν να βρεθούν κατά την δημιουργία της στον server. Αφού γίνει η σύνδεση, μπορεί να αρχίσει η διαδικασία δημιουργίας του σχήματος.

1.3.7 Version control

Όταν σε ένα project δουλεύουν πάνω από ένα άτομα τότε πρέπει να υπάρχει μια επικοινωνία αυτών των ατόμων σε επίπεδο κώδικα. Δηλαδή, πρέπει η δουλειά που κάνει κάποιος να μοιράζεται και στους άλλους χωρίς όμως επιπλοκές στην δική τους δουλειά και στον δικό τους κώδικά. Για να το λόγο αυτό έχουν αναπτυχθεί μερικά εργαλεία version control. Τα εργαλεία αυτά δεν διαμοιράζουν μόνο τον κώδικα αλλά διασφαλίζουν και την σωστή ένταξή του σε όλα τα μέλη της ομάδας και ειδοποιούν τους χρήστες για τυχόν επιπλοκές, που συνήθως υπάρχουν όταν δύο τουλάχιστον άτομα, εργάζονται πάνω στο ίδιο κομμάτι κώδικα. Στην παρούσα πτυχιακή έγινε χρήση του Git, που δημιουργήθηκε από τον Linus Torvalds το 2005, για την ανάπτυξη του πυρήνα Linux που χρησιμοποιείται στα Unix συστήματα. Για την αποθήκευση του κώδικα, ώστε να γίνει και ο διαμερισμός, χρησιμοποιήθηκε το Bitbucket της Atlassian. Εκεί, μετά τη δημιουργία ενός repository για το project, μέσα από το Visual Studio γίνεται η σύνδεση στο συγκεκριμένο repository και ο κώδικας θα ανεβαίνει εκεί.

2. ΑΝΑΛΥΣΗ ΤΗΣ WEB ΕΦΑΡΜΟΓΗΣ

2.1 Χρήστες

Η web εφαρμογή θα έχει τριών ειδών ρόλοι για τους χρήστες ώστε να μπορούν να καλυφθούν οι βασικές ανάγκες του συστήματος. Οι ρόλοι αυτοί είναι:

- Διαχειριστής (Admin)
- Χρήστης supermarket (Supermarket User)
- Απλός χρήστης

2.1.1 Απλός χρήστης

Ο απλός χρήστης, που είναι και ο κάθε πολίτης που θα μπαίνει στην εφαρμογή, είναι ο πιο απλός ρόλος του συστήματος. Ο απλός χρήστης δεν μπορεί να κάνει εγγραφή καθώς ο σκοπός του είναι να πάρει πληροφορίες για τα προϊόντα, οι οποίες είναι διαθέσιμες για όλους.

2.1.2 Χρήστης supermarket

Ο ρόλος του χρήστης supermarket είναι ένας ρόλος που δίνεται μόνο σε όσα supermarket έχουν την επιθυμία να εγγραφούν στο σύστημα και να εμφανίζουν τις προσφορές τους σε αυτό. Ένα supermarket δεν θα μπορεί να γραφτεί μόνο του στη σελίδα αλλά μόνο μέσω επικοινωνίας με τον διαχειριστή της σελίδας. Με αυτόν τον τρόπο γίνεται προσπάθεια αποφυγής κακόβουλων ατόμων που έχουν σκοπό να εγγραφούν στο σύστημα παριστάνοντας κάποιο supermarket.

Κάθε ένας που έχει το ρόλο ενός χρήστη supermarket μπορεί να δημιουργεί προσφορές από την κατάλληλη σελίδα και να τις δημοσιεύσει στο κοινό. Ο κάθε χρήστης μπορεί να δημιουργήσει προσφορές μόνο για το δικό του supermarket και όχι για κάποιο άλλο. Επίσης, μπορεί να επεξεργαστεί μόνο τις δικές του προσφορές και να τις τροποποιήσει όπως επιθυμεί ή να τις διαγράψει.

2.1.3 Διαχειριστής

Το ρόλο του διαχειριστή τον έχουν μόνο όσοι είναι υπεύθυνοι για το σύστημα και την σωστή και ομαλή λειτουργία του. Ο διαχειριστής είναι υπεύθυνος για την εγγραφή των χρηστών supermarket καθώς, όπως αναφέρθηκε, είναι ο μόνος που μπορεί να εγγράψει κάποιον στο σύστημα. Κατά την προσθήκη ενός νέου χρήστη

supermarket, ο διαχειριστής πρέπει να επιλέξει και το supermarket που εκπροσωπεί αυτός ο χρήστης. Η επιλογή αυτή γίνεται από μια λίστα ενός παραμετρικού πίνακα.

Επιπλέον, ο διαχειριστής έχει πρόσβαση στην επεξεργασία των παραμετρικών πινάκων. Εκεί μπορεί να προσθέσει κάποια νέα εγγραφή στους πίνακες, να επεξεργαστεί κάποια ή να διαγράψει κάποια. Η διαγραφή θα μπορεί να γίνει μόνο για κάποια εγγραφή ενός πίνακα που δεν χρησιμοποιείται καθώς σε διαφορετική περίπτωση γίνεται παραβίαση των κανόνων χρήσης των foreign keys. Οπότε ο διαχειριστής θα πρέπει να έχει και κάποιες τεχνικές γνώσεις.

2.2 Δημιουργία μιας προσφοράς

Η δημιουργία μιας προσφοράς μπορεί να γίνει από τους χρήστες supermarket. Για να γίνει αυτό πρέπει να μεταβούν στην σελίδα των προϊόντων όπου με το αντίστοιχο κουμπί μπορεί να γίνει προσθήκη μίας νέας προσφοράς αφού συμπληρωθούν τα εξής πεδία:

- Το όνομα προϊόντος
- Η κατηγορία που ανήκει το προϊόν (επιλογή από λίστα παραμετρικού πίνακα)
- Η αρχική τιμή του προϊόντος (€)
- Το ποσοστό προσφοράς (%)
- Η τελική τιμή του προϊόντος (€)
- Την ημερομηνία έναρξης της προσφοράς
- Την ημερομηνία λήξης της προσφοράς

Επιπλέον υπάρχει ένα πεδίο που δεν είναι προς συμπλήρωση και ονομάζεται “Supermarket”. Αυτό συμπληρώνεται αυτόματα κατά την αποθήκευση και παίρνει την τιμή που έχει ο χρήστης supermarket.

2.3 Αναζήτηση προσφορών

Η αναζήτηση των προσφορών γίνεται μέσα από την σελίδα με τα προϊόντα. Εκεί υπάρχουν κριτήρια αναζήτησης τα οποία μπορεί να συμπληρώσει ένας χρήστης ώστε να βρει το προϊόν που τον ενδιαφέρει. Τα κριτήρια αυτά είναι:

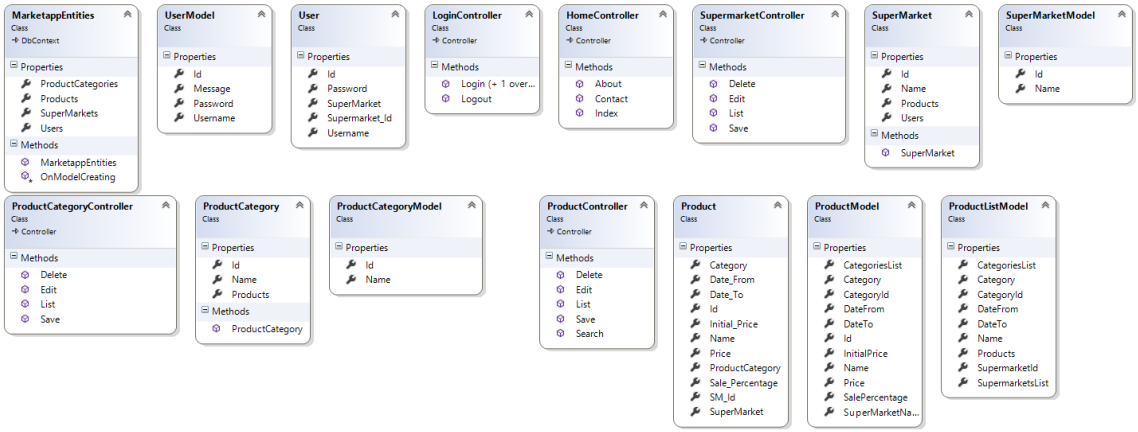
- Το όνομα προϊόντος
- Η κατηγορία που ανήκει το προϊόν (επιλογή από λίστα παραμετρικού πίνακα)
- Την ημερομηνία έναρξης της προσφοράς
- Την ημερομηνία λήξης της προσφοράς
- Το supermarket (επιλογή από λίστα παραμετρικού πίνακα)

Αφού συμπληρωθούν τα επιθυμητά κρητικιά και πατώντας το κουμπί της αναζήτησης γεμίζει ο πίνακας που βρίσκεται από κάτω με όλα τα στοιχεία που πληρούν τα κριτήρια που δόθηκαν. Οι στήλες που εμφανίζονται έχουν όλες τις πληροφορίες που έχει ένα προϊόν. Ανάλογα με την συσκευή με την οποία βλέπουμε την web εφαρμογή, εμφανίζονται λιγότερες στήλες ώστε να μείνουν στην οθόνη οι πιο βασικές για τον χρήστη. Πατώντας στο όνομα του προϊόντος ο χρήστης μπορεί να δει όλες τις πληροφορίες του σε οποιαδήποτε συσκευή.

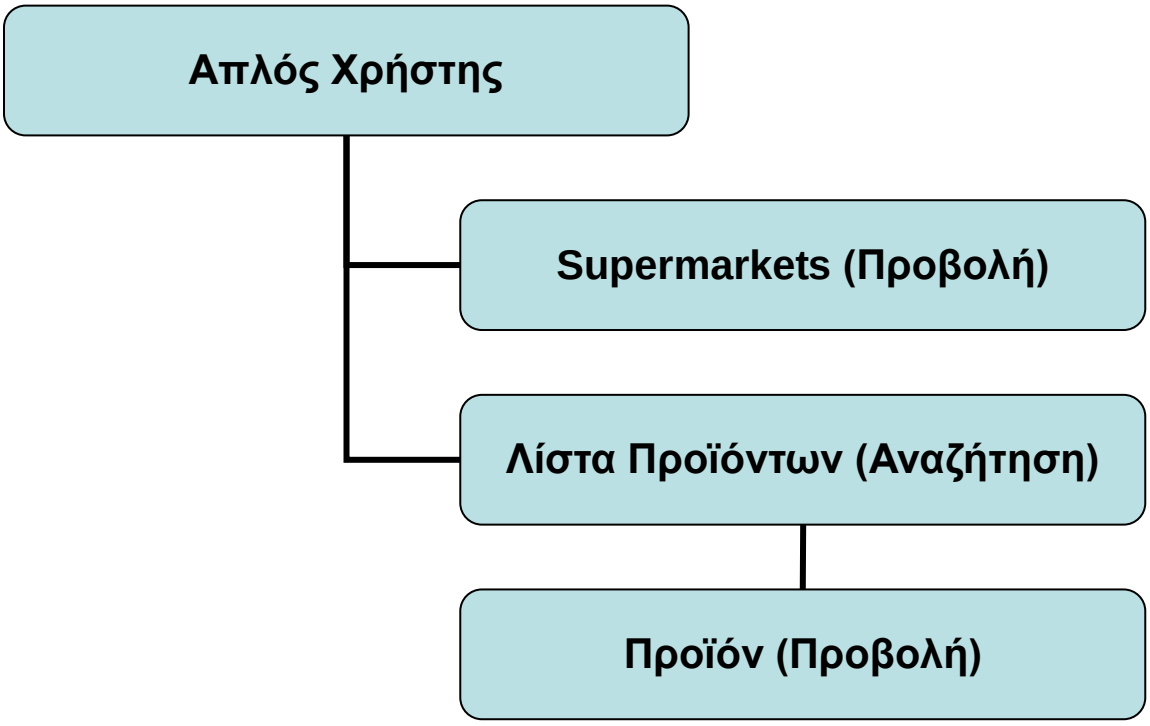
2.4 Παραμετρικοί Πίνακες

Οι παραμετρικοί πίνακες περιέχουν πληροφορίες που μπορεί να επεξεργαστεί μόνο ο διαχειριστής της εφαρμογής. Κάποιοι πίνακες είναι διαθέσιμοι για το κοινό, όπως είναι ο πίνακας με όλα τα supermarkets που υπάρχουν στην web εφαρμογή, και άλλοι πίνακες δεν φαίνονται αλλά φαίνονται κάποιες εγγραφές τους στις λίστες επιλογής, όπως είναι οι κατηγορίες των προϊόντων.

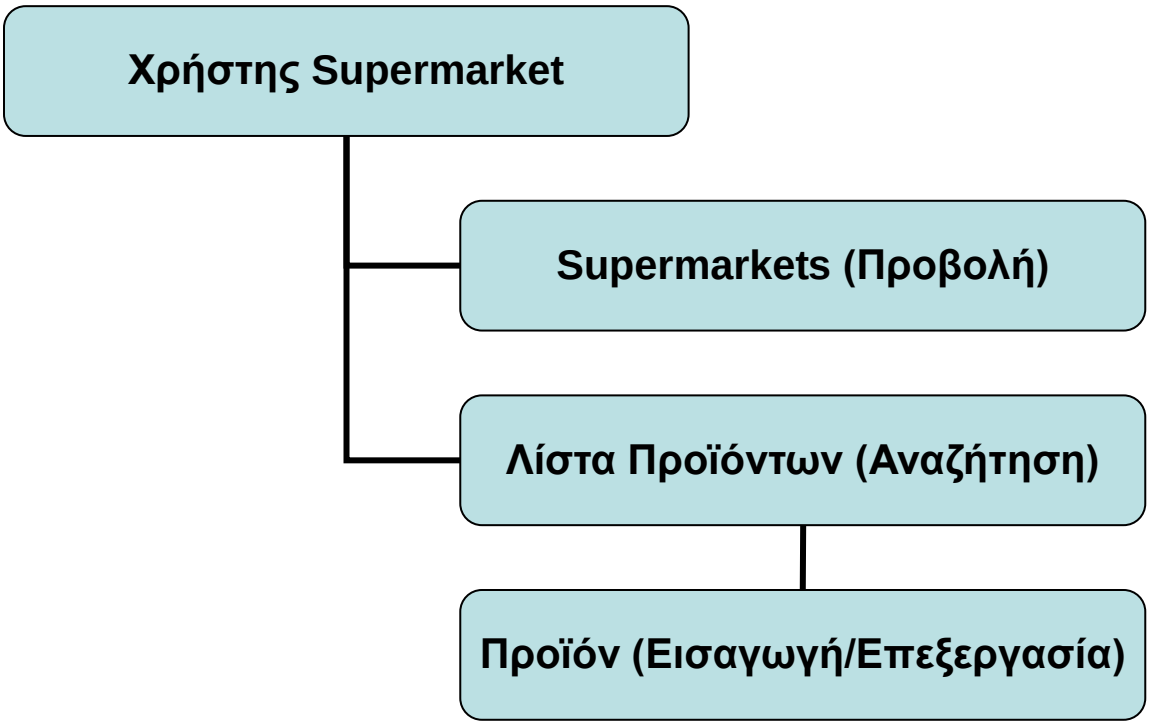
2.5 Διαγράμματα



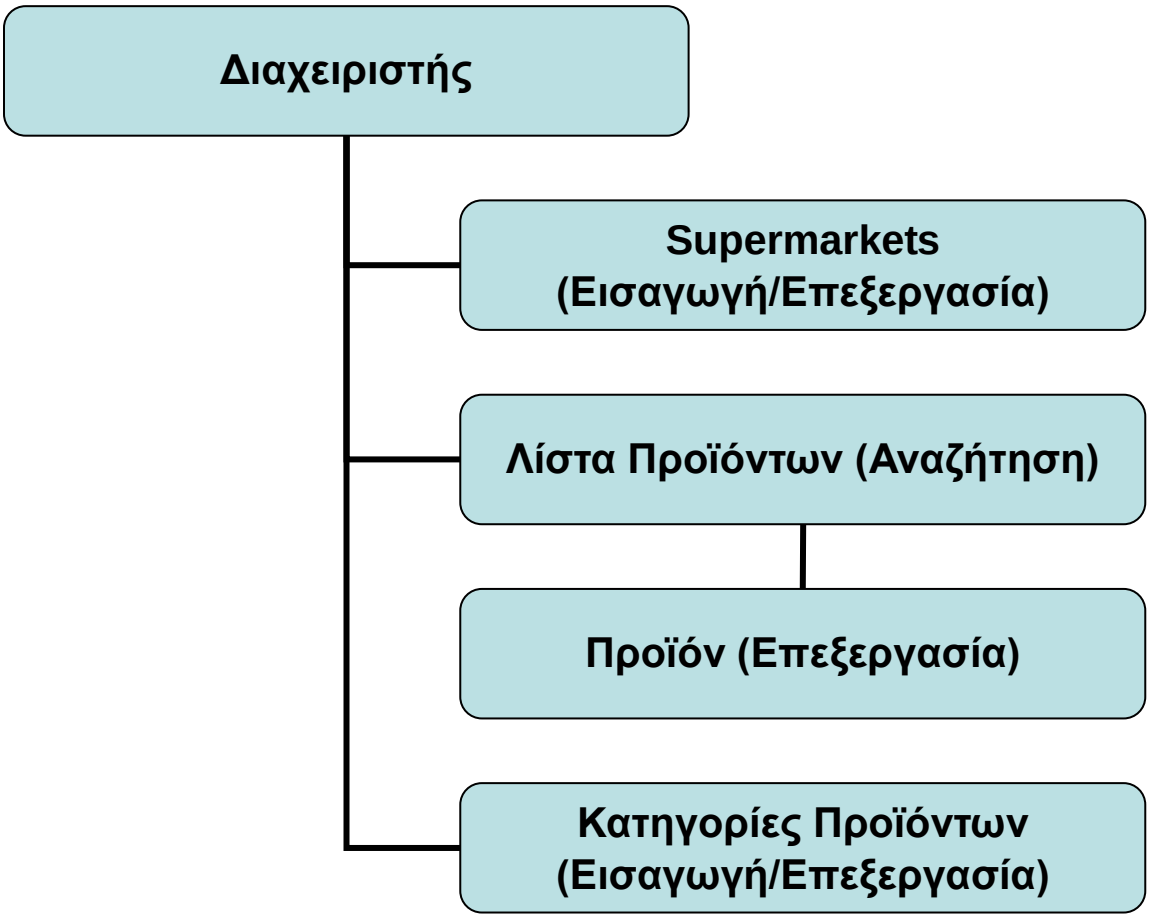
Class Diagram 1



Δικαιώματα Πρόσβασης Απλού Χρήστη 2



Δικαιώματα Πρόσβασης Χρήστη Supermarket 3



Δικαιώματα Πρόσβασης Διαχειριστή 4

3. ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΣΧΗΜΑΤΟΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Η δημιουργία του σχήματος της βάσης δεδομένων προϋποθέτει μια πρώτη ανάλυση των διαδικασιών που θα κάνει η web εφαρμογή καθώς και των αντικειμένων που θα χρησιμοποιηθούν σε αυτήν. Το σχήμα της βάσης μπορεί να επεξεργαστεί οποιαδήποτε στιγμή καθώς δεν γίνεται να γνωρίζει κάποιος όλα όσα θα χρειαστούν από την αρχή.

3.1 Πίνακες

Οι πίνακες που χρειάστηκαν για την υλοποίηση της web εφαρμογής είναι οι ακόλουθοι:

i. Ένας πίνακας με τα supermarkets, που ονομάστηκε SuperMarkets. Τα πεδία που έχει ο πίνακας είναι το Id και το όνομα του supermarket (Name).
ii. Ο επόμενος πίνακας είναι αυτός που περιέχει τα προϊόντα, με όνομα Products. Τα πεδία του πίνακα είναι:
    a. Id
    b. Το όνομα του προϊόντος (Name)
    c. Η κατηγορία του προϊόντος (Category, foreign key)
    d. Η αρχική τιμή του προϊόντος (Initial_Price)
    e. Η τελική τιμή του προϊόντος (Price)
    f. Το ποσοστό έκπτωσης (Sales_Percentage)
    g. Η ημερομηνία έναρξης της προσφοράς (Date_From)
    h. Η ημερομηνία λήξης της προσφοράς (Date_To)
iii. Οι κατηγορίες των προϊόντων μπήκαν σε έναν ξεχωριστό, παραμετρικό, πίνακα ώστε προϊόν να μπορεί να μπει σε μία από αυτές τις κατηγορίες και να γίνει πιο εύκολα η λειτουργία της εύρεσης προϊόντων. Ο πίνακας αυτός ονομάζεται ProductCategories και έχει για πεδία ένα Id και ένα όνομα (Name).
iv. Τέλος χρειάστηκε ένας πίνακας χρηστών (Users) που περιέχει το username, το password και το supermarket του χρήστη. Οι χρήστες που θα είναι εγγεγραμμένοι θα είναι μόνο αυτοί που θα βάζουν τις προσφορές στην web εφαρμογή, δηλαδή τα supermarkets. Το πεδίο αυτό είναι foreign key

στον πίνακα SuperMarkets. Κανένας χρήστης δεν θα μπορεί κάνει εγγραφή μόνος του, αλλά μόνο ο διαχειριστής της web εφαρμογής θα έχει τη δυνατότητα να προσθέσει έναν νέο χρήστη.

3.2 Κώδικας SQL

Τα SQL που χρησιμοποιήθηκαν για την δημιουργία των προαναφερθέντων πινάκων είναι τα εξής:

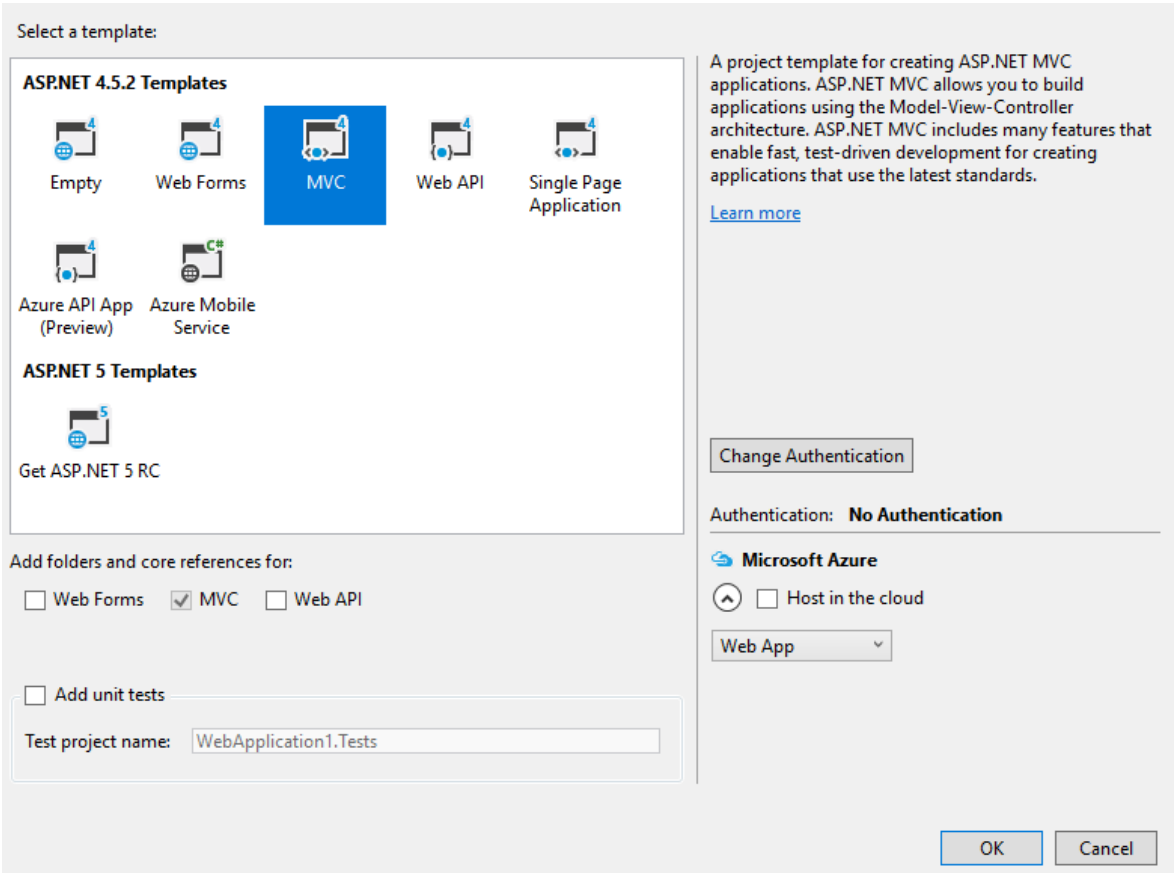
```
create table dbo.SuperMarkets(
        Id int primary key Identity(1,1) NOT NULL,
        Name nvarchar(200) NULL,
 CONSTRAINT PK_SuperMarkets PRIMARY KEY CLUSTERED
(


create table Products(
        Id int primary key Identity(1,1) NOT NULL,
        Price decimal(5, 2) NULL,
        Sale_Percentage decimal(5, 2) NULL,
        Date_From date NULL,
        Date_To date NULL,
        Category int NULL,
        SM_Id int NULL,
        Initial_Price decimal(5, 2) NULL,
        Name nvarchar(500) NULL,
PRIMARY KEY CLUSTERED
(
```

```
alter table Products add constraint FK_SuperMarkets foreign key(SM_Id) references dbo.SuperMarkets (Id)

alter table Products add constraint FK_Prs_PrCateg foreign key (Category) references ProductCategories (Id)

create table ProductCategories (
        Id int primary key Identity(1,1),
        Name nvarchar(100)
)

create table Users (
        Id int primary key Identity(1,1),
        Username nvarchar(100),
        Password nvarchar(100),
        Supermarket_Id int
)

alter table Users add constraint FK_Users_Supermarkets foreign key (Supermarket_Id) references SuperMarkets (Id)
```

4. Η WEB ΕΦΑΡΜΟΓΗ ΣΤΟ VISUAL STUDIO

4.1 Δημιουργία του MVC project

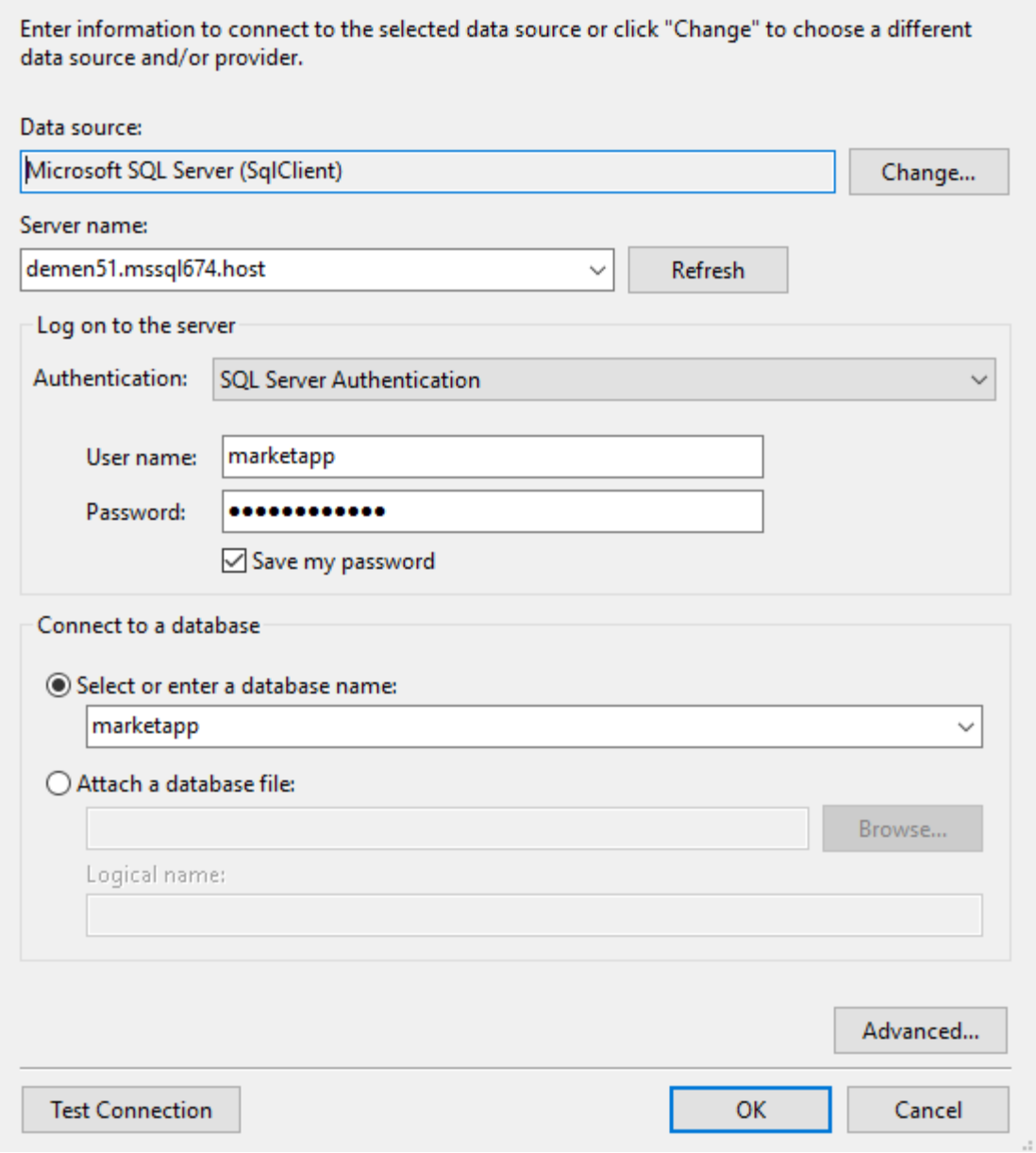
Το πρώτο πράγμα που χρειάζεται να γίνει είναι να δημιουργηθεί ένα νέο project μέσα στο Visual Studio. Όπως αναφέρθηκε η συγκεκριμένη web εφαρμογή θα γίνει με την χρήση της αρχιτεκτονικής του MVC, για το οποίο το Visual Studio παρέχει κάποια βασικά πράγματα προς χρήση. Έτσι κατά την δημιουργία του νέου project γίνεται η επιλογή του “Web Application” και μετά του MVC όπως φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 4.1 MVC Wizard

4.2 Σύνδεση του project με τη βάση δεδομένων

Το επόμενο σημαντικό βήμα είναι να γίνει η σύνδεση του project με τη βάση δεδομένων στον server. Μέσα στο Visual Studio υπάρχει μία οθόνη που ονομάζεται “Server Explorer”, από όπου γίνεται και η δημιουργία μίας νέας σύνδεσης με μία βάση δεδομένων. Πατώντας στο αντίστοιχο κουμπί εμφανίζεται η φόρμα συμπλήρωσης των στοιχείων για την σύνδεση με τον server και κατ’ επέκταση με τη βάση δεδομένων. Η εικόνα 4.2 δείχνει πως θα φαίνεται μια τέτοια φόρμα συμπληρωμένη. Επιπλέον, αφού γίνει η σύνδεση με τον server, εμφανίζεται από κάτω και η επιλογή της βάσης δεδομένων που έχει δημιουργηθεί.



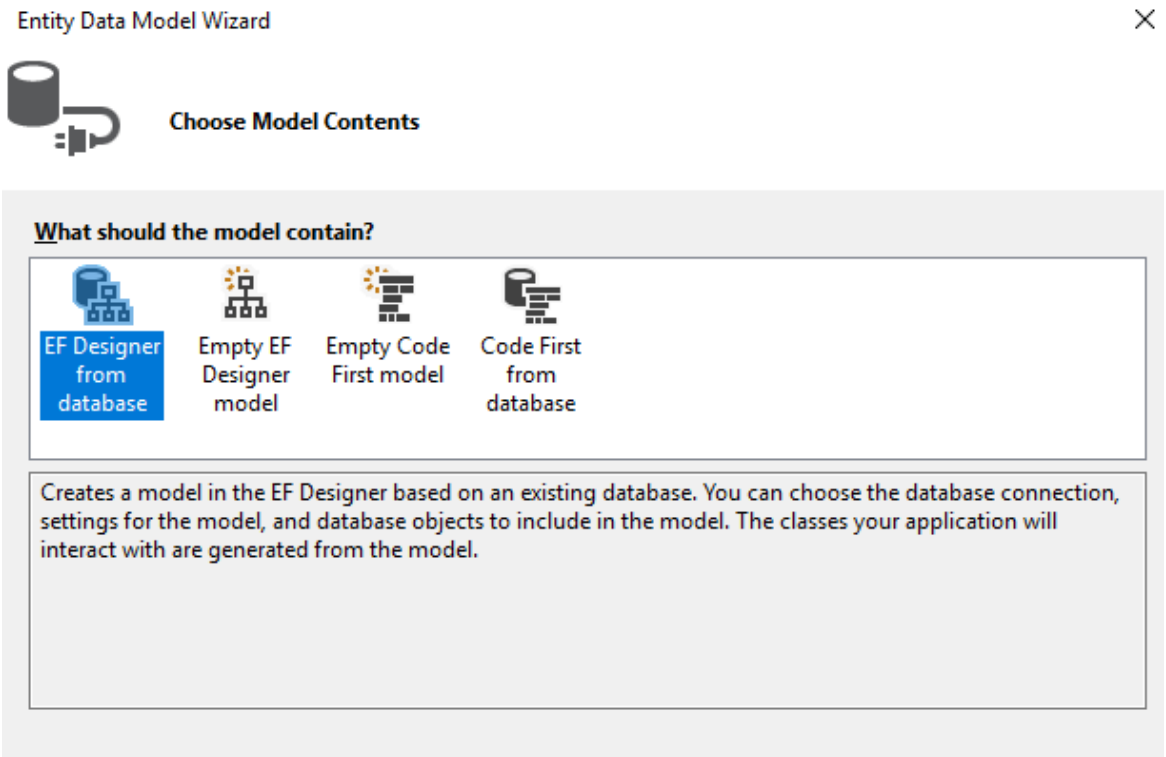
Εικόνα 4.2 Server Connection

Αφού ολοκληρωθεί η διαδικασία σύνδεσης με τον server, θα εμφανιστεί στο Data Connections του παραθύρου “Server Explorer” το όνομα του server και μέσα θα περιέχει τους πίνακες που έχει η βάση δεδομένων. Από εκεί μπορεί να γίνει και επεξεργασία των στοιχείων που συμπληρώθηκαν στην φόρμα, σε περίπτωση αλλαγής τους από τον server.

4.3 Δημιουργία κλάσεων

Σε συνέχεια της σύνδεσης με τη βάση δεδομένων, πρέπει να δημιουργηθούν οι κλάσεις οι οποίες θα περιέχουν τα δεδομένα που έρχονται από τον server. Για να επιτευχθεί αυτό έγινε χρήση των ADO.NET Entity Data Model Tools. Τα εργαλεία αυτά σχεδιάστηκαν για να βοηθούν με την δημιουργία εφαρμογών του Entity Framework. Ένα από τα χαρακτηριστικά τους είναι η χρήση του Database First, που αναφέρθηκε νωρίτερα και θα αναλυθεί εκτενώς.

Για διευκόλυνση του προγραμματιστή αλλά και για την καλύτερη οργάνωση του project, γίνεται δημιουργία ενός φακέλου μέσα στο project όπου θα αποθηκεύονται όσα έχουν σχέση με τη βάση δεδομένων. Μέσα σε αυτόν τον φάκελο, γίνεται επιλογή προσθήκης ενός νέου αρχείου, τύπου ADO.NET Entity Data Model. Μετά εμφανίζεται ένα παράθυρο με τέσσερις επιλογές, η πρώτη εκ των οποίων έχει το όνομα EF Designer from database.

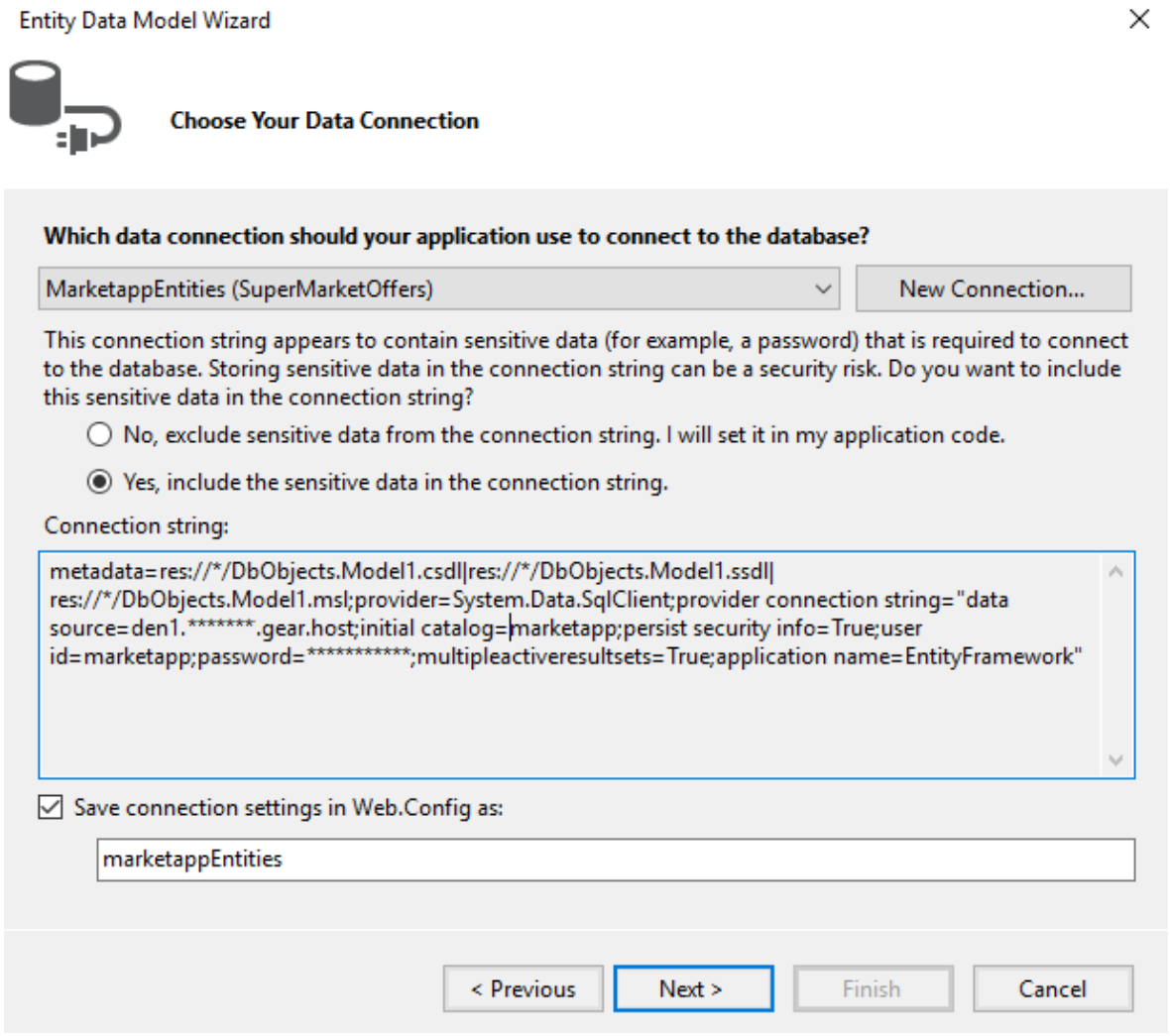


Εικόνα 4.3 Δημιουργία Database First

Όπως φαίνεται και στην περιγραφή, αυτό το εργαλείο δημιουργεί ένα μοντέλο στο Entity Framework Designer βάσει της υπάρχουσας βάσης δεδομένων και οι

κλάσεις που θα έχει η εφαρμογή θα δημιουργηθούν αυτόματα από αυτό το μοντέλο.

Προχωρώντας στο επόμενο βήμα, πρέπει να γίνει η σύνδεση με την επιθυμητή βάση δεδομένων. Η σύνδεση, όμως έχει γίνει στο κεφάλαιο 4.2 επομένως δεν χρειάζεται ξανά η ίδια διαδικασία αλλά αρκεί η επιλογή της συγκεκριμένης σύνδεσης από το κατάλληλο πεδίο όπως φαίνεται και στην επόμενη εικόνα.

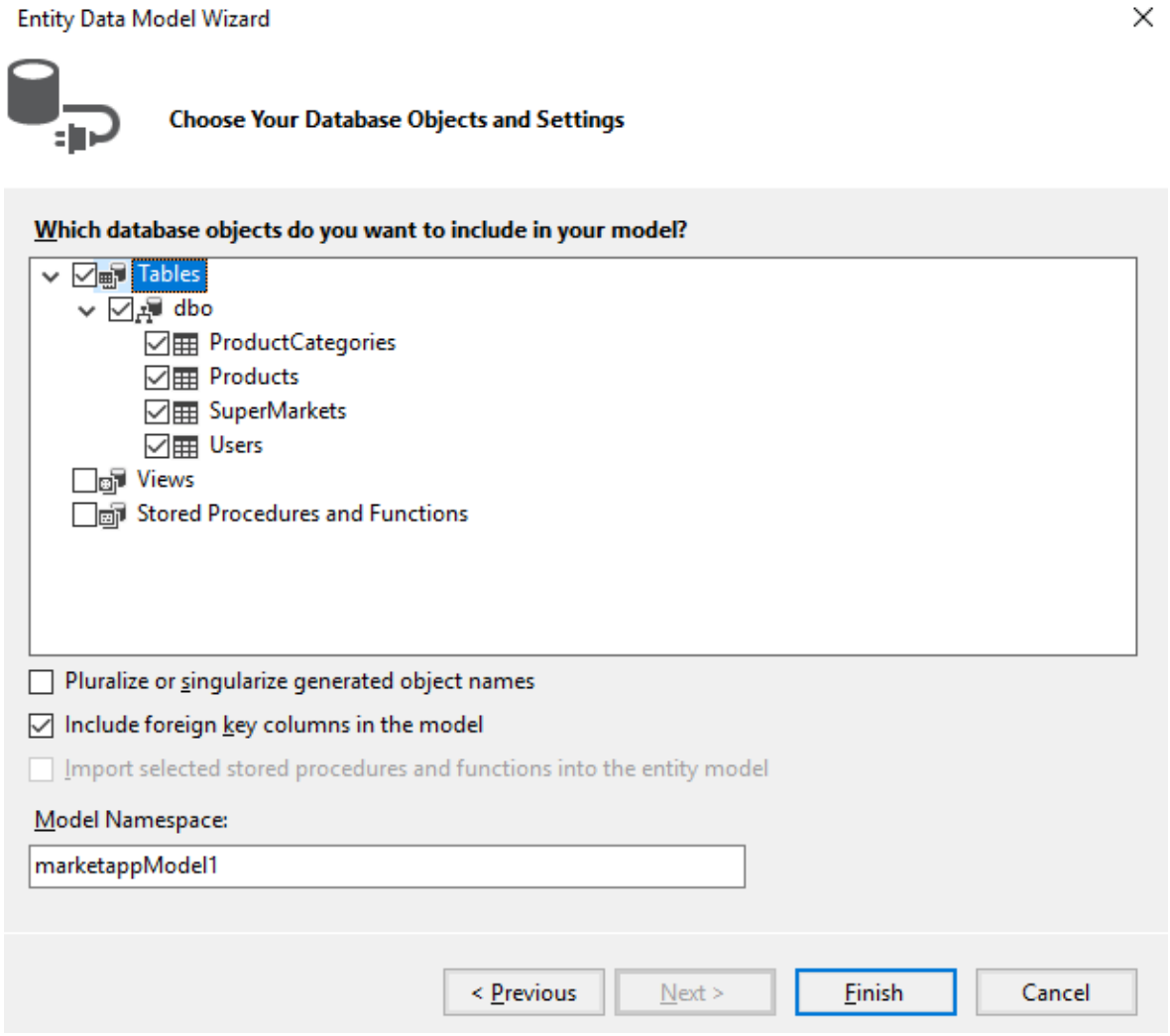


Εικόνα 4.4 Database First Connection

Η συγκεκριμένη διαδικασία δημιουργεί και το connection string στο Web.Config της εφαρμογής, το οποίο περιέχει χρήσιμες πληροφορίες που χρειάζεται η εφαρμογή. Το connection string είναι μια σειρά πληροφοριών που χρειάζεται η εφαρμογή για να μπορέσει να κάνει τη σύνδεση με τη βάση δεδομένων. Μερικές

από τις πληροφορίες που περιέχει είναι το όνομα του server, το όνομα της βάσης δεδομένων, ο κωδικός πρόσβασης και ένα μοναδικό όνομα ώστε να ξεχωρίζει από άλλα connection strings που θα μπορούσε να έχει η εφαρμογή.

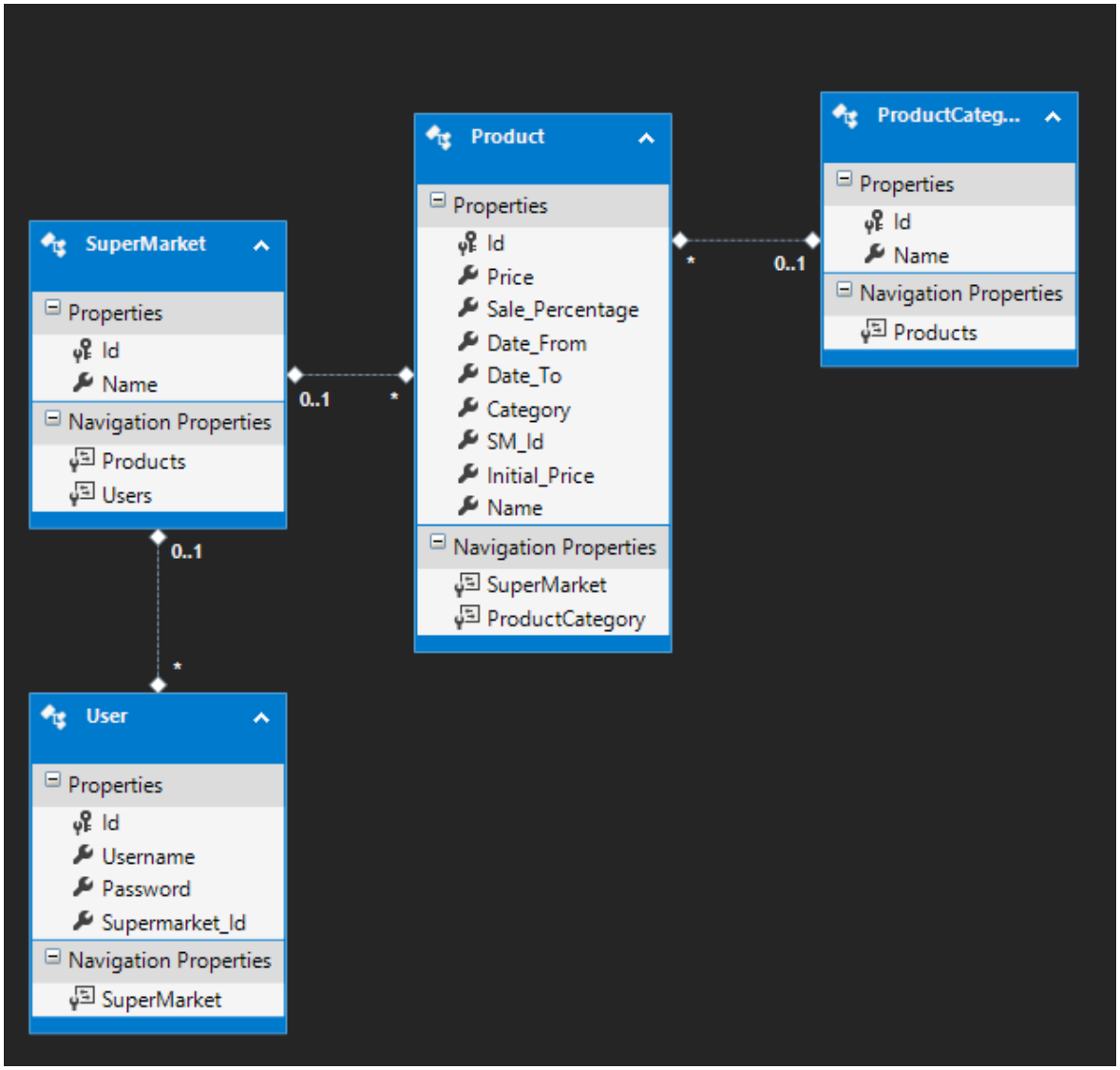
Στη συνέχεια, εμφανίζεται ένα παράθυρο από το οποίο πρέπει να γίνει η επιλογή των πινάκων που θα χρησιμοποιηθούν και να δοθεί και ένα όνομα στο μοντέλο που θα δημιουργηθεί.



Εικόνα 4.5 Επιλογή πινάκων Database First

Όπως φαίνεται και από την εικόνα, έχει γίνει επιλογή όλων των πινάκων που έχουν δημιουργηθεί για την πτυχιακή εργασία. Σε περίπτωση που η βάση είχε περισσότερους πίνακες από όσους χρειάζεται η web εφαρμογή τότε θα γινόταν επιλογή μόνο των αναγκαίων.

Μετά από το τελευταίο βήμα γίνεται η δημιουργία του μοντέλου, το οποίο εμφανίζεται και γραφικά στο Visual Studio.

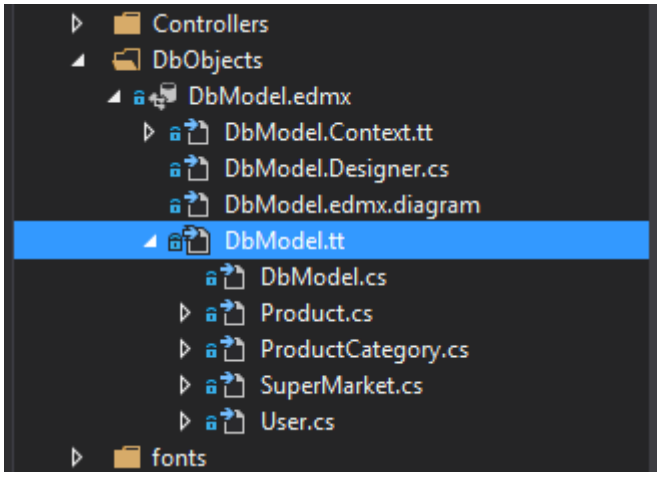


Εικόνα 4.6 Γραφική απεικόνιση κλάσεων

Όλοι οι πίνακες της βάσης έχουν πάρει μια γραφική απεικόνιση όπου φαίνονται όλες τους οι πληροφορίες όπως τα Primary Keys, τα Foreign Keys και τα πεδία τους.

Σημείωση: Σε περίπτωση αλλαγών της βάσης δεδομένων, η διαδικασία ενημέρωσης του μοντέλου γίνεται μέσα από το γραφικό περιβάλλον της παραπάνω εικόνας, πατώντας δεξί κλικ και “Update Model from Database”.

Πλέον οι κλάσεις έχουν δημιουργηθεί και μπορούν να βρεθούν μέσα στο μοντέλο, όπως φαίνεται και στην εικόνα που ακολουθεί.

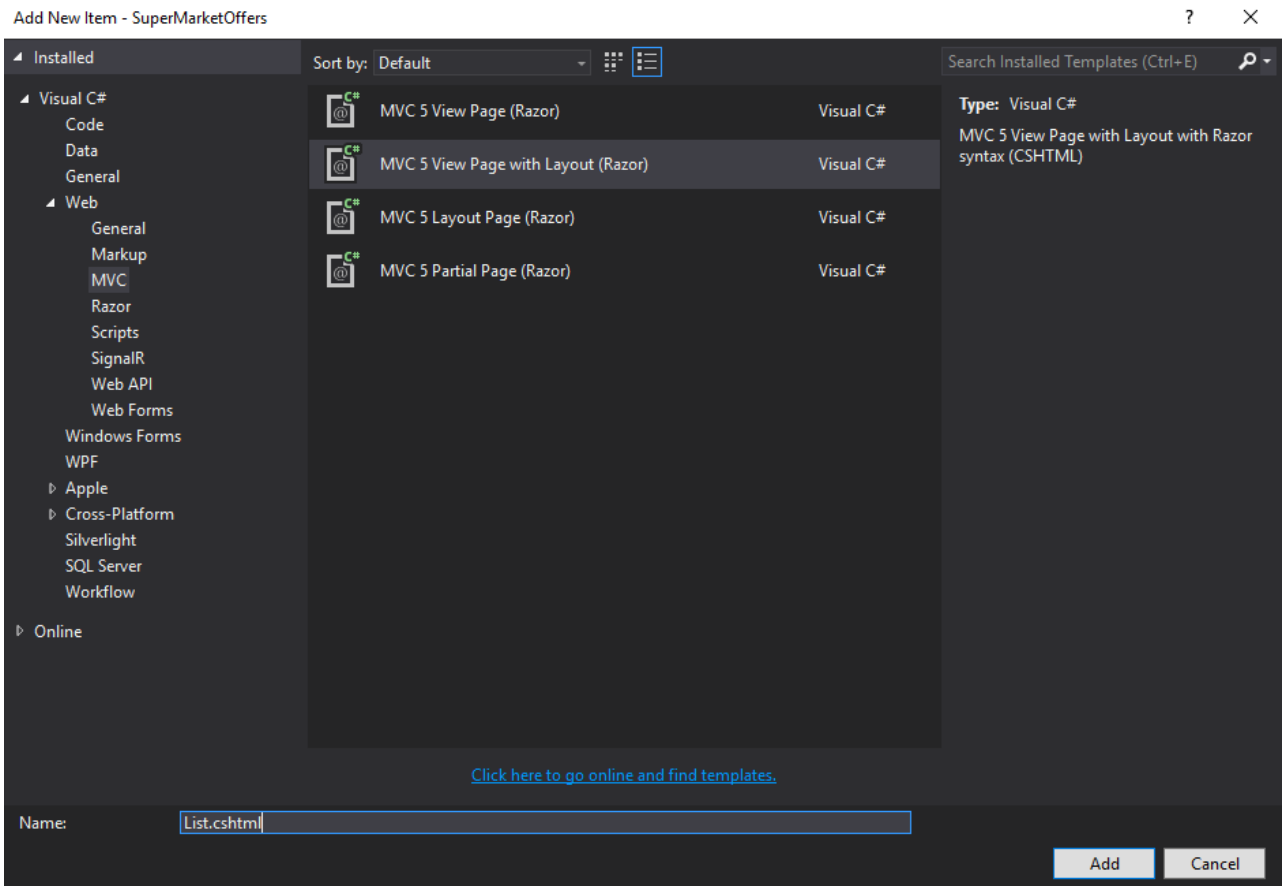


Εικόνα 4.7 Entity Model

4.4 Δημιουργία μίας ολοκληρωμένης οθόνης

4.4.1 Εισαγωγή ενός view

Οι οθόνες, ή αλλιώς τα views, που θα έχει το project αποθηκεύονται στον αντίστοιχο φάκελο Views. Μέσα σε αυτόν τον φάκελο δημιουργούνται υπό-φάκελοι για κάθε διαφορετική κλάση. Με αυτόν τον τρόπο επιτυγχάνεται μια οργάνωση των views που θα έχει η κάθε κλάση. Τα πρώτα views που θα δημιουργηθούν θα είναι για τα supermarkets. Αυτά που θα χρειαστούν είναι δύο views, το ένα που θα εμφανίζει μια λίστα με όλα τα supermarkets που έχουν περαστεί στην web εφαρμογή και το άλλο view που θα έχει την επεξεργασία ή την εισαγωγή ενός νέου. Η δημιουργία ενός view γίνεται από τον αντίστοιχο φάκελο, στην συγκεκριμένη περίπτωση είναι ο supermarkets, όπου γίνεται προσθήκη ενός MVC View Page (Razor) με το επιθυμητό όνομα. Η κατάληξη των αρχείων αυτών είναι .cshtml και το όνομα που δόθηκε είναι List.cshtml καθώς το view αυτό θα εμφανίζει τη λίστα με τα supermarkets.



Εικόνα 4.8 MVC Page

Αφού δημιουργηθεί το view αυτό θα πρέπει να οριστεί ένα model και ένας controller. Το model θα περιέχει τις πληροφορίες του αντικειμένου που θα εμφανιστεί στο view και ο controller θα κάνει την διαχείριση αυτών των πληροφοριών καθώς και την μεταφορά από ένα view σε ένα άλλο. Οπότε, προέχει πλέον η δημιουργία ενός model αρχείου.

4.4.2 Εισαγωγή ενός model

Η δημιουργία ενός model γίνεται με την επιλογή δημιουργίας ενός αρχείου class (.cs) στον φάκελο των models. Το όνομα του αρχείου έχει την κατάληξη Model.cs, ώστε να είναι εύκολα αναγνωρίσιμη η λειτουργεία του, οπότε για την περίπτωση των supermarkets ονομάστηκε SuperMarketModel.cs. Μέσα στο αρχείο θα μπουν τα πεδία που θα εμφανιστούν ή θα χρειαστούν στο view, δηλαδή το Id και το Name. Η μορφή που θα έχει το αρχείο είναι η εξής:

```
1   using System.ComponentModel;
2
3   namespace SuperMarketOffers.Models
4   {
5       public class SuperMarketModel
6       {
7           public int Id { get; set; }
8
9           [DisplayName("Όνομα")]
10          public string Name { get; set; }
11      }
12  }
```

Εικόνα 4.9 SuperMarketModel

4.4.3 Δημιουργία του view

Αφού δημιουργήθηκε το model, πλέον μπορεί το view να αρχίσει να παίρνει μορφή. Πρώτα από όλα γίνεται η δήλωση του μοντέλου που θα χρησιμοποιηθεί σε αυτό το view. Αυτό γίνεται με τον όρο @model στην κορυφή του αρχείου. Στο συγκεκριμένο view θα εμφανιστούν πολλά supermarkets ή αλλιώς μια λίστα από supermarkets, άρα αυτό που θα δηλωθεί είναι @model List<SuperMarketsModel>. Για να εμφανιστεί αυτή η λίστα στην οθόνη, θα χρησιμοποιηθεί ένας πίνακας σε html. Το body του πίνακα γεμίζει με γραμμές, κάθε μια από τις οποίες έχει πληροφορίες για ένα supermarket. Αυτό επιτυγχάνεται με τη χρήση C# μέσα στην html και συγκεκριμένα με μία foreach στο model. Για να μπορέσει να γίνει χρήση της C# μέσα στο αρχείο πρέπει να υπάρχει το σύμβολο ‘@’, όπως και έγινε και με την δήλωση του μοντέλου. Στην παρακάτω εικόνα φαίνεται όλος ο κώδικας που χρησιμοποιήθηκε για την δημιουργία του view.

```
@using SuperMarketOffers.Models
@model List<SuperMarketModel>
@{
    ViewBag.Title = "Supermarkets";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@if ((string)Session["Username"] == "admin")
{
    @Html.ActionLink("Προσθήκη", "Edit", null, new { @class = "btn btn-primary" })
}
<div class="tab-content">
    <div class="tab-pane fade in active" id="supermarketListView">
        <div class="table-responsive">
            <table id="supermarketTable" class="table">
                <thead>
                    <tr>
                        <th>AA</th>
                        <th>Ονομα supermarket</th>
                        @if ((string)Session["Username"] == "admin")
                        {
                            <th>Ενέργειες</th>
                        }
                    </tr>
                </thead>
                <tbody>
                    @foreach (SuperMarketModel supermarket in Model)
                    {
                        <tr class="form-group">
                            <td>@supermarket.Id</td>
                            <td>@supermarket.Name</td>

                            @if ((string)Session["Username"] == "admin")
                            {
                                <td>
                                    @Html.ActionLink("Επεξεργασία", "Edit", new { id = supermarket.Id }) |
                                    @Html.ActionLink("Διαγραφή", "Delete", new { id = supermarket.Id })
                                </td>
                            }
                        </tr>
                    }
                </tbody>
            </table>
        </div>
    </div>
</div>

```

Εικόνα 4.10 Supermarket List

Στον παραπάνω κώδικα γίνεται μια αναφορά στο Session. Στο κεφάλαιο 5.7 γίνεται αναλυτική περιγραφή της χρήσης και λειτουργίας του Session. Αξίζει να αναφερθεί όμως, ότι ο συγκεκριμένος κώδικας του session χρησιμοποιείται για να κόψει κάποιες δυνατότητες από τους χρήστες που δεν έχουν τον ρόλο του admin, δηλαδή του διαχειριστή. Επιπλέον, υπάρχουν δύο κουμπιά τύπου Html.ActionLink. Αυτά επιτρέπουν την επικοινωνία του view με οποιονδήποτε controller, και συγκεκριμένα καλούν την επιθυμητή μέθοδο που βρίσκεται στον κατάλληλο controller.

4.4.4 Δημιουργία του Controller

Μετά την δημιουργία του view πρέπει να υλοποιηθεί και ο controller, όπου το model θα γεμίζει με τις κατάλληλες πληροφορίες οι οποίες θα εμφανιστούν στο view.

Οι controllers συνήθως έχουν τον ακόλουθο τρόπο ονοματολογίας "…Controller.cs". Στην συγκεκριμένη περίπτωση το όνομα που δόθηκε είναι "SuperMarketController.cs" καθώς θα κάνει όλη την διαχείριση που αφορά τα supermarkets. Μετά την δημιουργία του αρχείου .cs πρέπει να του δοθεί και η λειτουργικότητα του controller. Αυτό γίνεται με το να κληρονομήσει από τον 'βασικό' controller του MVC.

Σε αυτό το σημείο πρέπει να αναφερθεί ότι το όνομα του controller, πριν από την λέξη 'controller', πρέπει να είναι ίδιο με το όνομα με τον φάκελο που έχουν τοποθετηθεί τα views τα οποία διαχειρίζεται. Αυτό βοηθάει τον controller, αλλά και τον προγραμματιστή, να βρίσκει πιο εύκολα το view που πρέπει. Για να καλεστεί ένα view χρειάζεται μέσα στον controller να δημιουργηθεί ένα ActionResult, δηλαδή τι θα γίνει όταν μια ενέργεια καταλήξει σε αυτή τη μέθοδο. Το συγκεκριμένο ActionResult θα είναι μια μέθοδος που θα αρχικά θα καλεί το view που δημιουργήθηκε προηγουμένως. Αυτό γίνεται με τον ακόλουθο κώδικα.

```
public class SupermarketController: Controller
{
    public ActionResult List()
    {
        List<SuperMarket> superMarkets;
        using (MarketappEntities db = new MarketappEntities())
        {
            superMarkets = db.SuperMarkets.ToList();
        }
        List<SuperMarketModel> model = superMarkets.Select(x => new SuperMarketModel
        {
            Id = x.Id,
            Name = x.Name
        }).ToList();

        return View("List", model);
    }
```

Εικόνα 4.11 SupermarketController

Η μέθοδος ονομάστηκε List() για να συμβαδίζει με το view που καλεί. Μέσα στην μέθοδο έχει γίνει αρχικοποίηση του MarketappEntities το οποίο προσφέρει πρόσβαση στην βάση δεδομένων μέσω του Entity Framework. Στη συνέχεια αφού περαστούν όλες οι πληροφορίες που χρειάζονται μέσα σε μία λίστα από supermarkets, γίνεται μεταφορά των πληροφοριών στο κατάλληλο μοντέλο, και πιο συγκεκριμένα, λίστα μοντέλων καθώς το view χρειάζεται μια λίστα από SuperMarketModels.

Παρατήρηση: Η χρήση του using γίνεται επειδή προσφέρει τη δυνατότητα να σβηστεί από τη μνήμη το MarketappEntities, μόλις τελειώσει η χρήση του. Με αυτόν τον τρόπο αποδεσμεύεται χώρος στην RAM αξιοποιώντας καλύτερα τις δυνατότητές της.

4.4.5 Πρόσβαση της σελίδας των supermarket από το κεντρικό μενού

Για να μπορέσει να υπάρξει πρόσβαση σε αυτή τη σελίδα θα πρέπει να μπει το κατάλληλο κουμπί στο κεντρικό μενού. Έτσι θα υπάρχει και εύκολη πρόσβαση σε όποια σελίδα κι αν βρίσκεται ο χρήστης. Το κεντρικό μενού είναι κάτι που φαίνεται σε όλη την web εφαρμογή, σε όλες τις σελίδες της. Κατά την δημιουργία του project, το Visual Studio δημιούργησε ένα τέτοιο μενού και το αποθήκευσε στα Views/Shared/_Layout.cshtml.

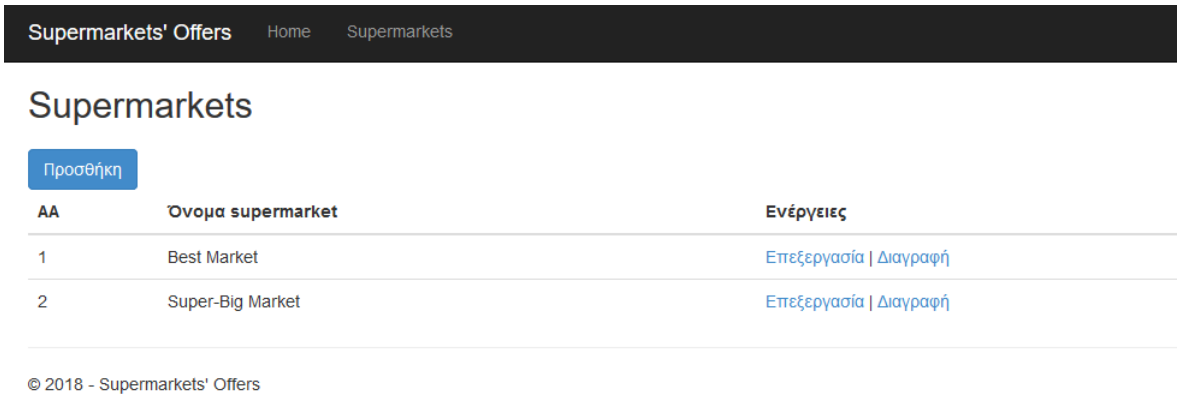
```
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Supermarkets' Offers", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Supermarkets", "List", "Supermarket")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Supermarkets' Offers</p>
        </footer>
    </div>
```

Εικόνα 4.12 Layout

Στην παραπάνω εικόνα φαίνεται να γίνεται ξανά χρήση του Html.ActionLink αλλά με διαφορετικούς παραμέτρους. Η διαφορά στους δύο τρόπους χρήσης είναι η αναφορά του controller. Όταν ο controller δεν αναφέρεται τότε αυτόματα ψάχνει τον controller που κάλεσε αυτή τη σελίδα. Αν ο controller αναφέρεται τότε θα ψάξει για την συγκεκριμένη συνάρτηση σε εκείνον τον controller. Στην παρούσα περίπτωση, δηλαδή @Html.ActionLink("Supermarkets", "List", "Supermarket"), αυτό που θα συμβεί είναι να ψάξει το σύστημα μέσα στον “Supermarket”, που είναι ο controller, την μέθοδο List, και να την καλέσει.

4.4.6 Εμφάνιση της σελίδας.

Στην παρακάτω εικόνα φαίνεται η μορφή που έχει πάρει η σελίδα και το μενού. Έχουν περαστεί και μερικά δεδομένα από τη βάση ώστε να υπάρχει μια πιο ολοκληρωμένη εικόνα της σελίδας.



Εικόνα 4.13 Supermakets

4.5 Οθόνη εισαγωγής και επεξεργασίας

Το επόμενο βήμα είναι η προσθήκη και η επεξεργασία των supermarkets μέσα από την web εφαρμογή και όχι μόνο μέσα από τη βάση όπως έγινε πιο πάνω.

4.5.1 Σελίδα εισαγωγής

Μέσα στον φάκελο supermarkets, που δημιουργήθηκε πριν, θα τοποθετηθεί και η νέα σελίδα, που σκοπό θα έχει την εισαγωγή αλλά και την επεξεργασία ενός supermarket. Το όνομα που δόθηκε σε αυτή τη σελίδα είναι “Edit” και ο κώδικάς της είναι ο ακόλουθος.

```
@using SuperMarketOffers.Models
@model SuperMarketModel
@{
    ViewBag.Title = !string.IsNullOrEmpty(Model.Name) ? $"{Model.Name}" : "Νέο Supermarket";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@using (Html.BeginForm("Save", "Supermarket", FormMethod.Post))
{
    @Html.Hidden("Id", Model.Id)

    <div class="form-group">
        @Html.LabelFor(model => model.Name, new {@class = "control-label col-md-2 "})
        <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new {htmlAttributes = new {@class = "form-control"}})
            @Html.ValidationMessageFor(model => model.Name, "", new {@class = "text-danger"})
        </div>
    </div>

    <input type="submit" value="Αποθήκευση" class="btn btn-primary"/>
}
```

Εικόνα 4.14 Supermarket Edit View

Σε αυτή τη σελίδα το model που χρειάζεται είναι το SuperMarketModel, καθώς μόνο ένα supermarket θα αποθηκεύεται κάθε φορά. Για να περαστούν οι πληροφορίες εύκολα στον controller και να υποθηκευθούν, γίνεται χρήση του Html.BeginForm. Με αυτό τον τρόπο, όλα τα δεδομένα που υπάρχουν μέσα στα άγκιστρα «{ }» θα περαστούν στο model και θα πάνε στην “Save” του “Supermarket” controller, η οποία είναι η συνάρτηση που έχει δηλωθεί ότι θα καλεστεί κατά την υποβολή. Για την υποβολή υπάρχει ένα απλό κουμπί στο τέλος της σελίδας.

4.5.2 Μέθοδος πρόσβασης στη σελίδα

Στον controller πλέον πρέπει να προστεθεί μία συνάρτηση “Edit()” που θα καλεί την σελίδα της εικόνας 5.14.

```
public ActionResult Edit(int? id)
{
    ProductCategoryModel model = new ProductCategoryModel();
    if (id.HasValue)
    {
        ProductCategory category;
        using (MarketappEntities db = new MarketappEntities())
        {
            category = db.ProductCategories.Single(x => x.Id == id.Value);
        }

        model.Id = category.Id;
        model.Name = category.Name;
    }

    return View("Edit", model);
}
```

Εικόνα 4.15 Supermarket Controller Edit

Η συνάρτηση “Edit()” έχει δομηθεί με τέτοιο τρόπο που μπορεί να επεξεργαστεί και την εισαγωγή ενός νέου supermarket καθώς και την επεξεργασία ενός υπάρχοντος. Αυτό καθορίζεται από το id που υπάρχει ως παράμετρος της συνάρτησης. Αν το id που έχει περαστεί είναι null τότε σημαίνει ότι πρέπει να εμφανιστεί άδεια, από δεδομένα, η σελίδα καθώς σκοπός είναι να δημιουργηθεί μία νέα εγγραφή στα supermarkets. Αν δεν είναι null τότε θα βρει μέσα στη βάση την εγγραφή με το αντίστοιχο id και θα την επιστρέψει στην οθόνη προς επεξεργασία. Η παράμετρος id γεμίζει με δεδομένα κατά την κλήση της. Στην σελίδα List των supermarkets, η Edit συνάρτηση καλείται δύο φορές με διαφορετικό τρόπο. Η μία φορά είναι από το κουμπί «Προσθήκη» και καλείται χωρίς παραμέτρους ενώ την άλλη φορά καλείται από το κουμπί «Επεξεργασία» και περνάει ως παράμετρος το id του supermarket.

4.5.3 Αποθήκευση ενός supermarket

Το επόμενο βήμα είναι να μπορεί να γίνει αποθήκευση ενός νέου supermarket από την web εφαρμογή. Όπως αναφέρθηκε και προηγουμένως στο view, θα υπάρχει μια μέθοδος με όνομα "Save" η οποία θα παίρνει τις πληροφορίες που

υπάρχουν μέσα στην φόρμα του view και θα τις αποθηκεύει. Αυτό επιτυγχάνεται με τον παρακάτω κώδικα.

```
public ActionResult Save(SuperMarketModel model)
{
    using (MarketappEntities db = new MarketappEntities())
    {
        if (model.Id == 0)
        {
            SuperMarket superMarket = new SuperMarket
            {
                Name = model.Name
            };

            db.SuperMarkets.Add(superMarket);
        }
        else
        {
            SuperMarket superMarket = db.SuperMarkets.Single(x => x.Id == model.Id);
            superMarket.Name = model.Name;
        }
        db.SaveChanges();
    }

    return RedirectToAction("List");
}
```

Εικόνα 4.16 Αποθήκευση Supermarket

Η Save χρησιμοποιείται και για την αποθήκευση ενός νέου supermarket αλλά και για την επεξεργασία ενός ήδη υπάρχοντος. Για τον λόγο αυτό γίνεται ένας διαχωρισμός βάσει του model.Id, το οποίο κρίνει αν θα το μοντέλο ανήκει σε νέα εγγραφή ή όχι. Πρέπει να αναφερθεί ότι για να μπορέσουν να γίνουν αλλαγές σε ένα υπάρχον supermarket τότε πρέπει πρώτα να ανακτηθεί από τη βάση η εγγραφή του και να αλλάξουν οι τιμές της σύμφωνα με τις νέες και στη συνέχεια να αποθηκευτεί. Αυτή η διαδικασία πρέπει να γίνε μέσα στο ίδιο using του MarketappEntities αλλιώς θα αναγνωριστεί ως διαφορετική εγγραφή και θα πάρει άλλο id.

4.6 Δημιουργία της σελίδας αναζήτησης των προϊόντων

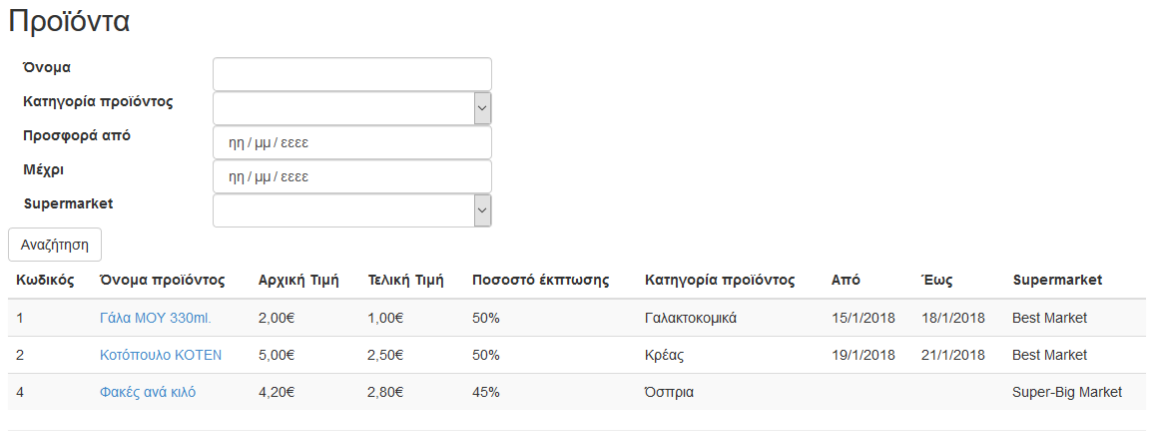
Η σελίδα που θα εμφανίζει όλα τα προϊόντα είναι λίγο διαφορετική από εκείνη που είχε όλα τα supermarkets. Αυτό οφείλεται στο ότι ο χρήστης θα μπορεί σε αυτή τη σελίδα να κάνει αναζήτηση σε όλα τα διαθέσιμα προϊόντα με κάποια κριτήρια αναζήτησης.

4.6.1 Κριτήρια αναζήτησης προϊόντων

Τα κριτήρια αναζήτησης που επιλέχθηκαν είναι αυτά που ο χρήστης θα χρειάζεται περισσότερο για να βρει τα επιθυμητά προϊόντα. Αυτά τα κριτήρια είναι τα παρακάτω:

- Το όνομα του προϊόντος ή μέρος του ονόματος, για διευκόλυνση του χρήστη.
- Η κατηγορία που ανήκει το προϊόν (από λίστα με όλες τις κατηγορίες).
- Ημερομηνίες από και έως ισχύος της προσφοράς.
- Επιλογή supermarket (από τη λίστα όλων των διαθέσιμων).

Η συμπλήρωση των παραπάνω κριτηρίων είναι προαιρετική και μπορεί να γίνει οποιοσδήποτε συνδυασμός κριτηρίων.



Εικόνα 4.17 Κριτήρια αναζήτησης

Στην εικόνα 5.17 φαίνεται η σελίδα με τα κριτήρια αναζήτησης καθώς και μερικά από τα αποτελέσματα βάσει των δεδομένων που υπάρχουν στο σύστημα. Για την σελίδα αυτή ο κώδικας είναι ο ακόλουθος:

```
@using SuperMarketOffers.Models
@model ProductListModel
@{
    ViewBag.Title = "Προϊόντα";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@using (Html.BeginForm("Search", "Product", FormMethod.Get))
{
    <div class="form-group">
        @Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2 " })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.CategoryId, new { @class = "control-label col-md-2 " })
        <div class="col-md-10">
            @Html.DropDownListFor(model => model.CategoryId, Model.CategoriesList, "", new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateFrom, new { @class = "control-label col-md-2 " })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateFrom, new { htmlAttributes = new { @class = "form-control", @type = "date" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateTo, new { @class = "control-label col-md-2 " })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateTo, new { htmlAttributes = new { @class = "form-control", @type = "date" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.SupermarketId, new {@class = "control-label col-md-2 "})
        <div class="col-md-10">
            @Html.DropDownListFor(model => model.SupermarketId, Model.SupermarketsList, "", new {@class = "form-control"})
        </div>
    </div>
    <input type="submit" value="Αναζήτηση" class="btn btn-default" />
}
```

Εικόνα 4.18 Κριτήρια αναζήτησης View

```
<div class="tab-content">
    <div class="tab-pane fade in active" id="productListView">
        <div class="table-responsive">
            <table id="productTable" class="table table-striped">
                <thead>
                    <tr>
                        <th class="hidden-xs hidden-sm">Κωδικός</th>
                        <th>Όνομα προϊόντος</th>
                        <th class="hidden-xs hidden-sm">Αρχική Τιμή</th>
                        <th>Τελική Τιμή</th>
                        <th class="hidden-xs hidden-sm">Ποσοστό έκπτωσης</th>
                        <th>Κατηγορία προϊόντος</th>
                        <th>Από</th>
                        <th>Έως</th>
                        <th>Supermarket</th>
                        @if (Session["Username"] != null)
                        {
                            <th>Ενέργειες</th>
                        }
                    </tr>
                </thead>
                <tbody>
                    @foreach (ProductModel product in Model.Products)
                    {
                        <tr class="form-group">
                            <td class="hidden-xs hidden-sm">@product.Id</td>
                            <td>@Html.ActionLink(@product.Name, "Edit", new { product.Id, viewOnly = true }) </td>
                            <td class="hidden-xs hidden-sm">@product.InitialPrice€</td>
                            <td>@product.Price€</td>
                            <td class="hidden-xs hidden-sm">@(product.SalePercentage?.ToString("####,##"))%</td>
                            <td>@product.Category.Name</td>
                            <td>@(product.DateFrom?.ToShortDateString())</td>
                            <td>@(product.DateTo?.ToShortDateString())</td>
                            <td>@product.SuperMarketName</td>
                            @if (Session["Username"] != null)
                            {
                                <td>@Html.ActionLink("Επεξεργασία", "Edit", new {product.Id, viewOnly = false}) |
                                    @Html.ActionLink("Διαγραφή", "Delete", new {product.Id})</td>
                            }
                        </tr>
                    }
                </tbody>
            </table>
        </div>
    </div>
</div>
```

Εικόνα 4.19 Πίνακας αποτελεσμάτων

4.6.2 Διαχείριση αναζήτησης προϊόντων

Με το πάτημα του κουμπιού «Αναζήτηση» στην σελίδα των προϊόντων, το σύστημα πηγαίνει στην συνάρτηση “Search” στον controller των προϊόντων, η οποία έχει ως παράμετρο ένα ProductListModel, που περιέχει όλα τα κριτήρια αναζήτησης. Αυτά τα κριτήρια αναζήτησης ελέγχονται ένα προς ένα για το ποια έχουν συμπληρωθεί και με τη χρήση της LINQ γίνεται η αναζήτηση μέσα στα προϊόντα. Ο κώδικας που κάνει αυτή την εργασία είναι ο ακόλουθος:

```
if (!string.IsNullOrEmpty(model.Name))
    dbProducts = dbProducts.Where(x => x.Name.Contains(model.Name)).ToList();

if (model.CategoryId != null)
    dbProducts = dbProducts.Where(x => x.ProductCategory.Id == model.CategoryId).ToList();

if (model.SupermarketId.HasValue)
    dbProducts = dbProducts.Where(x => x.SuperMarket.Id == model.SupermarketId.Value).ToList();

if (model.DateFrom.HasValue || model.DateTo.HasValue)
{
    if (model.DateFrom.HasValue && model.DateTo.HasValue)
        dbProducts = dbProducts.Where(x => x.Date_From <= model.DateFrom && x.Date_To >= model.DateTo).ToList();
    else if (model.DateFrom.HasValue)
        dbProducts = dbProducts.Where(x => x.Date_From <= model.DateFrom).ToList();
    else
        dbProducts = dbProducts.Where(x => x.Date_To >= model.DateTo).ToList();
}
```

Εικόνα 4.20 Αναζήτηση προϊόντων LINQ

Μετά την εύρεση των αποτελεσμάτων πρέπει τα αποτελέσματα αυτά να περαστούν μέσα σε ένα μοντέλο το οποίο θα επιστραφεί στην οθόνη αναζήτησης. Αυτό γίνεται με τον εξής τρόπο.

```
    model = new ProductListModel
    {
        Products = dbProducts.Select(x => new ProductModel
        {
            Id = x.Id,
            Name = x.Name,
            Price = x.Price,
            InitialPrice = x.Initial_Price,
            SalePercentage = x.Sale_Percentage,
            Category = x.ProductCategory,
            DateFrom = x.Date_From,
            DateTo = x.Date_To,
            SuperMarketName = x.SuperMarket.Name
        }).ToList(),
        CategoriesList = db.ProductCategories.Select(x => new SelectListItem { Text = x.Name, Value = x.Id.ToString() }).ToList(),
        SupermarketsList = db.SuperMarkets.Select(x => new SelectListItem { Text = x.Name, Value = x.Id.ToString() }).ToList()
    };
}

return View("List", model);
```

Εικόνα 4.21 Δημιουργία μοντέλου αποτελεσμάτων αναζήτησης

4.7 Η χρήστη του Session

Το session είναι ένα object το οποίο έχει ο κάθε χρήστης που εισέρχεται στην εφαρμογή. Το session είναι σαν global μεταβλητή και είναι διαθέσιμο από οποιοδήποτε σημείο της εφαρμογής.

Ένα πολύ δυνατό χαρακτηριστικό του session είναι η αποθήκευση κάποιων δεδομένων από οποιαδήποτε σελίδα και η μεταφορά τους σε οποιαδήποτε άλλη σελίδα. Ένα παράδειγμα τέτοιας χρήσης είναι το UserId και του Username που αποθηκεύεται στο session κατά την είσοδο ενός χρήστη στην εφαρμογή. Έτσι είναι εύκολο να βρεθούν οι πληροφορίες για τον κάθε χρήστη και να οριστεί σε ποια πεδία και κουμπιά θα έχει πρόσβαση σε κάθε σημείο της εφαρμογής, απλώς ψάχνοντας μέσα στο session.

Για να γίνει αποθήκευση μιας πληροφορίας μέσα στο session πρέπει να οριστεί ένα όνομα για την μεταβλητή που θα περαστεί μέσα. Το όνομα που θα οριστεί έχει σημασία μόνο για τον προγραμματιστή καθώς με αυτό το όνομα θα μπορεί να πάρει την συγκεκριμένη πληροφορία από το session όποτε τη χρειαστεί. Το όνομα αυτό μπαίνει μέσα σε αγκύλες όπως στο παράδειγμα που ακολουθεί.

```
Session[“UserId”] = object
```

5. ΠΡΟΟΠΤΙΚΕΣ ΕΞΕΛΙΞΗΣ

5.1 Security

Μία από τις πιο σημαντικές διαδικασίες που πρέπει να γίνουν για την εξέλιξη της web εφαρμογής είναι το security. Αυτό γίνεται σε πολλά βήματα. Ένα από αυτά είναι τα passwords στην βάση δεδομένων πρέπει να είναι κωδικοποιημένα. Ένα ακόμη βήμα που σχετίζεται με την βάση είναι να μπει κώδικας που θα εμποδίζει τα SQL injections. Επιπλέον, θα πρέπει να γίνεται έλεγχος κατά την επεξεργασία και αποθήκευση των προϊόντων, ώστε να επιβεβαιώνεται ότι ο χρήστης που κάνει την αποθήκευση είναι όντως αυτός που το έχει της δικαίωμα επεξεργασίας του.

5.2 Δημιουργία λογαριασμού απλού χρήστη

Ένα στοιχείο που θα μπορούσε να μπει για να εξελιχθεί η web εφαρμογή είναι ο κάθε χρήστης που το επιθυμεί να έχει δικό του λογαριασμό. Με αυτόν τον τρόπο μπορούν να δοθούν στον χρήστη κάποιες επιπλέον δυνατότητες όπως οι παρακάτω:

- Επιλογή αγαπημένων προϊόντων / supermarket ώστε να μπορεί ο χρήστης να τα βρει πιο γρήγορα.
- Επιλογή ενεργοποίησης ειδοποιήσεων όταν ένα επιλεγμένο ή αγαπημένο προϊόν δεχτεί κάποια προσφορά.
- Επιλογή ενεργοποίησης ειδοποιήσεων όταν ένα επιλεγμένο ή αγαπημένο supermarket ανεβάσει κάποια νέα προσφορά.
- Σχόλια χρηστών για κάποιο προϊόν ή κάποιο supermarket.

5.3 Προγραμματισμός προσφορών

Ακόμα μία καλή λειτουργία θα ήταν ο προγραμματισμός των προσφορών. Ο χρήστης supermarket θα μπορεί να φτιάχνει μια προσφορά και να ορίζει μέσω ενός πεδίου πότε η προσφορά αυτή θα εμφανίζεται στον χρήστη. Με τον τρόπο αυτό θα μπορεί να δημιουργήσει πολλές προσφορές μαζί και να τις βάλει να εμφανίζονται η κάθε μία σε διαφορετική στιγμή χωρίς να χρειάζεται να ξαναμπεί στην web εφαρμογή για να την ενεργοποιήσει.

5.4 Σύγκριση προϊόντων

Ο χρήστης θα μπορεί να επιλέξει μερικά προϊόντα, τα οποία θα εμφανίζονται μαζί στην οθόνη του ώστε να μπορεί να τα συγκρίνει μεταξύ τους και τα επιλέξει ποιο από όλα θέλει να πάρει.

5.5 Reskinning

Επιπλέον ένα στοιχείο που θα μπορούσε να αλλάξει είναι η συνολική εικόνα της εφαρμογής, ώστε να είναι πιο προσιτή στο κοινό και θα την χειρίζεται πιο ευχάριστα.

5.6 Χρήση εικόνων στα προϊόντα

Ακόμη μία χρήσιμη αλλαγή για να γίνει πιο ευχάριστη η εφαρμογή στο κοινό είναι η χρήση εικόνων από τα προϊόντα που υπάρχουν σε αυτήν.

5.7 Disclaimer

Σε περίπτωση που η web εφαρμογή μπει σε διαδικασία παραγωγής ώστε να βγει στο κοινό τότε θα χρειαστεί ένα disclaimer που να αναφέρει ότι η εφαρμογή δεν ευθύνεται για τις τιμές των προϊόντων και απλώς είναι ένα μέσο άμεσης ενημέρωσης του ενδιαφερόμενου κοινού.

6. ΠΑΡΑΡΤΗΜΑ Α’

6.1 Login

6.1.1 LoginController

```
using System.Linq;
using System.Web.Mvc;
using SuperMarketOffers.DbObjects;
using SuperMarketOffers.Models;

namespace SuperMarketOffers.Controllers
{
    public class LoginController: Controller
    {
        public ActionResult Login()
        {
            UserModel model = new UserModel();
            return View("Login", model);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(UserModel model)
        {
            if (ModelState.IsValid)
            {
                using (MarketappEntities db = new MarketappEntities())
                {
                    User user = db.Users.FirstOrDefault(a =>
a.Username.Equals(model.Username) && a.Password.Equals(model.Password));
                    if (user != null)
                    {
                        Session["UserID"] = user.Id.ToString();
                        Session["Username"] = user.Username;
                        return View("~/Views/Home/Index.cshtml");
                    }
                    model.Message = "Υπήρξε σφάλμα με τα στοιχεία που δώσατε.
Δοκιμάστε ξανά.";
                }
            }
            return View(model);
        }

        public ActionResult Logout()
        {
            Session["UserID"] = null;
            Session["Username"] = null;
            return View("~/Views/Home/Index.cshtml");
        }
    }
}
```

6.1.2 Login (View)

```
@model SuperMarketOffers.Models.UserModel
```

```
@{
    ViewBag.Title = "Login";
}

@using (Html.BeginForm("Login", "Login", FormMethod.Post))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    if (@Model.Message != null)
    {
        <div class="form-group text-danger">
            @Model.Message
        </div>
    }
    <div class="form-group">
        @Html.LabelFor(model => model.Username, new {@class = "control-label col-md-
2 "})
        <div class="col-md-10">
            @Html.EditorFor(model => model.Username, new {htmlAttributes = new
{@class = "form-control"}})
            @Html.ValidationMessageFor(model => model.Username, "", new {@class =
"text-danger"})
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Password, new {@class = "control-label col-md-
2 "})
        <div class="col-md-10">
            @Html.PasswordFor(model => model.Password, new {@class = "form-
control"})
            @Html.ValidationMessageFor(model => model.Password, "", new {@class =
"text-danger"})
        </div>
    </div>
    <div class="form-group">
        <input type="submit" value="Login" class="btn btn-primary" />
    </div>
}
```

6.1.3 User Model

```
namespace SuperMarketOffers.Models
{
    public class UserModel
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string Message { get; set; }
    }
}
```

6.2 Layout (κεντρικό μενού)

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Supermarkets' Offers", "Index", "Home", new { area
= "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Supermarkets", "List", "Supermarket")</li>
                    <li>@Html.ActionLink("Προϊόντα", "List", "Product")</li>
                    @if ((string)Session["Username"] == "admin")
                    {
                        <li>@Html.ActionLink("Κατηγορίες Προϊόντων", "List",
"ProductCategory")</li>
                    }

                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    @if (Session["Username"] == null)
                    {
                        <li>@Html.ActionLink("Login", "Login", "Login")</li>
                    }
                    else
                    {
                        <li>@Html.ActionLink($"Logout ({Session["Username"]})",
"Logout", "Login")</li>
                    }

                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Supermarkets' Offers</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", false)
```

```
</body>
</html>
```

6.3 Home Page

6.3.1 HomeController

```
using System.Web.Mvc;

namespace SuperMarketOffers.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            return View();
        }

    }
}
```

6.3.2 Index (View)

```
@{
    ViewBag.Title = "Αρχική";
}

<div class="jumbotron">
    <h1>Προσφορές Supermarkets</h1>
    <p class="lead">Βρείτε εύκολα και γρήγορα τις προσφορές που υπάρχουν στα
supermerket για τα αγαπημένα σας προϊόντα.</p>
</div>

<div class="row">
    <div class="col-md-6">
        <h2>Προϊόντα</h2>
        <p>Αναζήτηση σε όλα τα προϊόντα της εφαρμογής για την εύρεση των καλύτερων
προσφορών.</p>
        <p>@Html.ActionLink("Μετάβαση στην αναζήτηση", "List", "Product", null, new
{ @class = "btn btn-primary" })</p>
    </div>
    <div class="col-md-6">
        <h2>Supermarkets</h2>
        <p>Δείτε ποιά είναι τα supermarket που έχουν επιλέξει να προσθέσουν τα
προϊόντα τους στην εφαρμογή.</p>
        <p>@Html.ActionLink("Μετάβαση στα supermarkets", "List", "Supermarket",
null, new { @class = "btn btn-primary" })</p>
    </div>
</div>
```

6.3.3 About (View)

```
@{
    ViewBag.Title = "Σχετικά";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

<p>Η συγκεκριμένη web εφαρμογή αποτελεί μέρος της πτυχιακής εργασίας των φοιτητών Λιούτα Χ. και Λαμπαδάκη Γ.</p>
```

6.4 Supermarkets

6.4.1 SupermarketController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Microsoft.Ajax.Utilities;
using SuperMarketOffers.DbObjects;
using SuperMarketOffers.Models;

namespace SuperMarketOffers.Controllers
{
    public class SupermarketController: Controller
    {
        public ActionResult List()
        {
            List<SuperMarket> superMarkets;
            using (MarketappEntities db = new MarketappEntities())
            {
                superMarkets = db.SuperMarkets.ToList();
            }
            List<SuperMarketModel> model = superMarkets.Select(x => new
SuperMarketModel
            {
                Id = x.Id,
                Name = x.Name
            }).ToList();

            return View("List", model);
        }

        public ActionResult Edit(int? id)
        {
            SuperMarketModel model = new SuperMarketModel();
            if (id.HasValue)
            {
                SuperMarket superMarket;
                using (MarketappEntities db = new MarketappEntities())
                {
                    superMarket = db.SuperMarkets.Single(x => x.Id == id.Value);
                }
```

```
                model.Id = superMarket.Id;
                model.Name = superMarket.Name;
            }

            return View("Edit", model);
        }

        public ActionResult Save(SuperMarketModel model)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                if (model.Id == 0)
                {
                    SuperMarket superMarket = new SuperMarket
                    {
                        Name = model.Name
                    };

                    db.SuperMarkets.Add(superMarket);
                }
                else
                {
                    SuperMarket superMarket = db.SuperMarkets.Single(x => x.Id ==
model.Id);
                    superMarket.Name = model.Name;
                }
                db.SaveChanges();
            }

            return RedirectToAction("List");
        }

        public ActionResult Delete(int id)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                db.SuperMarkets.Remove(db.SuperMarkets.Single(x => x.Id == id));
                db.SaveChanges();
            }
            return RedirectToAction("List");
        }
    }
  }
```

6.4.2 List (Supermarket View)

```
@using SuperMarketOffers.Models
@model List<SuperMarketModel>
@{
    ViewBag.Title = "Supermarkets";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@if ((string)Session["Username"] == "admin")
```

```
{
    @Html.ActionLink("Προσθήκη", "Edit", null, new { @class = "btn btn-primary" })
}
<div class="tab-content">
    <div class="tab-pane fade in active" id="supermarketListView">
        <div class="table-responsive">
            <table id="supermarketTable" class="table">
                <thead>
                    <tr>
                        <th>AA</th>
                        <th>Όνομα supermarket</th>
                        @if ((string)Session["Username"] == "admin")
                        {
                            <th>Ενέργειες</th>
                        }
                    </tr>
                </thead>
                <tbody>
                    @foreach (SuperMarketModel supermarket in Model)
                    {
                        <tr class="form-group">
                            <td>@supermarket.Id</td>
                            <td>@supermarket.Name</td>

                            @if ((string)Session["Username"] == "admin")
                            {
                                <td>
                                    @Html.ActionLink("Επεξεργασία", "Edit", new { id
= supermarket.Id }) |
                                    @Html.ActionLink("Διαγραφή", "Delete", new { id
= supermarket.Id })
                                </td>
                            }
                        </tr>
                    }
                </tbody>
            </table>
        </div>
    </div>
</div>
```

6.4.3 Edit (Supermarket View)

```
@using SuperMarketOffers.Models
@model ProductCategoryModel
@{
    ViewBag.Title = !string.IsNullOrEmpty(Model.Name) ? $"{Model.Name}" : "Νέα
κατηγορία προϊόντων";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@using (Html.BeginForm("Save", "ProductCategory", FormMethod.Post))
{
    @Html.Hidden("Id", Model.Id)
```

```
    <div class="form-group">
        @Html.LabelFor(model => model.Name, new {@class = "control-label col-md-2
"})
        <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new {htmlAttributes = new {@class =
"form-control"}})
            @Html.ValidationMessageFor(model => model.Name, "", new {@class = "text-
danger"})
        </div>
    </div>

    <input type="submit" value="Αποθήκευση" class="btn btn-primary"/>
}
```

6.4.4 SupermarketModel

```
using System.ComponentModel;

namespace SuperMarketOffers.Models
{
    public class SuperMarketModel
    {
        public int Id { get; set; }

        [DisplayName("Όνομα")]
        public string Name { get; set; }
    }
}
```

6.5 Product Categories

6.5.1 ProductCategoryController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SuperMarketOffers.DbObjects;
using SuperMarketOffers.Models;

namespace SuperMarketOffers.Controllers
{
    public class ProductCategoryController : Controller
    {
        public ActionResult List()
        {
            List<ProductCategory> categories;
            using (MarketappEntities db = new MarketappEntities())
            {
                categories = db.ProductCategories.ToList();
            }
            List<ProductCategoryModel> model = categories.Select(x => new
ProductCategoryModel
```

```
            {
                Id = x.Id,
                Name = x.Name
            }).ToList();

            return View("List", model);
        }

        public ActionResult Edit(int? id)
        {
            ProductCategoryModel model = new ProductCategoryModel();
            if (id.HasValue)
            {
                ProductCategory category;
                using (MarketappEntities db = new MarketappEntities())
                {
                    category = db.ProductCategories.Single(x => x.Id == id.Value);
                }

                model.Id = category.Id;
                model.Name = category.Name;
            }

            return View("Edit", model);
        }

        public ActionResult Save(ProductCategoryModel model)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                if (model.Id == 0)
                {
                    ProductCategory superMarket = new ProductCategory
                    {
                        Name = model.Name
                    };

                    db.ProductCategories.Add(superMarket);
                }
                else
                {
                    ProductCategory category = db.ProductCategories.Single(x => x.Id
== model.Id);
                    category.Name = model.Name;
                }
                db.SaveChanges();
            }

            return RedirectToAction("List");
        }

        public ActionResult Delete(int id)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                db.ProductCategories.Remove(db.ProductCategories.Single(x => x.Id ==
id));
                db.SaveChanges();
```

```
            }
            return RedirectToAction("List");
        }


    }
}
```

6.5.2 List (ProductCategory View)

```
@using SuperMarketOffers.Models
@model List<ProductCategoryModel>
@{
    ViewBag.Title = "Κατηγορίες Προϊόντων";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@if ((string)Session["Username"] == "admin")
{
    @Html.ActionLink("Προσθήκη", "Edit", null, new { @class = "btn btn-primary" })
}
<div class="tab-content">
    <div class="tab-pane fade in active" id="productCategoryListView">
        <div class="table-responsive">
            <table id="productCategoryTable" class="table">
                <thead>
                    <tr>
                        <th>AA</th>
                        <th>Όνομα κατηγορίας</th>
                        @if ((string)Session["Username"] == "admin")
                        {
                            <th>Ενέργειες</th>
                        }
                    </tr>
                </thead>
                <tbody>
                    @foreach (ProductCategoryModel category in Model)
                    {
                        <tr class="form-group">
                            <td>@category.Id</td>
                            <td>@category.Name</td>

                            @if ((string)Session["Username"] == "admin")
                            {
                                <td>
                                    @Html.ActionLink("Επεξεργασία", "Edit", new { id
= category.Id }) |

                                    @Html.ActionLink("Διαγραφή", "Delete", new { id
= category.Id })
                                </td>
                            }
                        </tr>
                    }
                </tbody>
            </table>
```

```
        </div>
    </div>
</div>
```

6.5.3 Edit (ProductCategory View)

```
@using SuperMarketOffers.Models
@model ProductCategoryModel
@{
    ViewBag.Title = !string.IsNullOrEmpty(Model.Name) ? $"{Model.Name}" : "Νέα
κατηγορία προϊόντων";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@using (Html.BeginForm("Save", "ProductCategory", FormMethod.Post))
{
    @Html.Hidden("Id", Model.Id)

    <div class="form-group">
        @Html.LabelFor(model => model.Name, new {@class = "control-label col-md-2
"})
        <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new {htmlAttributes = new {@class =
"form-control"}})
            @Html.ValidationMessageFor(model => model.Name, "", new {@class = "text-
danger"})
        </div>
    </div>

    <input type="submit" value="Αποθήκευση" class="btn btn-primary"/>
}
```

6.5.4 ProductCategoryModel

```
namespace SuperMarketOffers.Models
{
    public class ProductCategoryModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
                                        }
```

6.6 Products

6.6.1 ProductController

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using SuperMarketOffers.DbObjects;
using SuperMarketOffers.Models;
```

```
namespace SuperMarketOffers.Controllers
{
    public class ProductController: Controller
    {
        public ActionResult List()
        {
            ProductListModel model;
            using (MarketappEntities db = new MarketappEntities())
            {
                User user = new User();
                if (!string.IsNullOrEmpty((string) Session["UserId"]))
                {
                    int userId = int.Parse((string) Session["UserId"]);
                    user = db.Users.Single(x => x.Id == userId);
                }

                List<Product> dbProducts = user.SuperMarket != null ?
db.Products.Where(x => x.SuperMarket.Id == user.SuperMarket.Id).ToList() :
db.Products.ToList();

                model = new ProductListModel
                {
                    Products = dbProducts.Select(x => new ProductModel
                    {
                        Id = x.Id, Name = x.Name, Price = x.Price, InitialPrice =
x.Initial_Price, SalePercentage = x.Sale_Percentage, Category = x.ProductCategory,
DateFrom = x.Date_From, DateTo = x.Date_To, SuperMarketName = x.SuperMarket.Name
                    }).ToList(),
                    CategoriesList = db.ProductCategories.Select(x => new
SelectListItem {Text = x.Name, Value = x.Id.ToString()}).ToList(), SupermarketsList
= db.SuperMarkets.Select(x => new SelectListItem {Text = x.Name, Value =
x.Id.ToString()}).ToList()
                };
            }
            return View("List", model);
        }

        public ActionResult Edit(int? id, bool viewOnly)
        {
            ProductModel model;
            using (MarketappEntities db = new MarketappEntities())
            {
                model = new ProductModel
                {
                    CategoriesList = new List<SelectListItem>()
                };

                model.CategoriesList = db.ProductCategories.Select(x => new
SelectListItem { Text = x.Name, Value = x.Id.ToString() }).ToList();

                if (id.HasValue)
                {
                    Product product = db.Products.Single(x => x.Id == id.Value);

                    model.Id = product.Id;
                    model.Name = product.Name;
```

```
                    model.Name = product.Name;
                    model.Price = product.Price;
                    model.InitialPrice = product.Initial_Price;
                    model.SalePercentage = product.Sale_Percentage;
                    model.CategoryId = product.ProductCategory.Id;
                    model.Category = product.ProductCategory;
                    model.DateFrom = product.Date_From;
                    model.DateTo = product.Date_To;
                    model.SuperMarketName = product.SuperMarket.Name;
                }
            }

            return View(viewOnly ? "Index" : "Edit", model);
        }


        public ActionResult Save(ProductModel model)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                if (model.Id == 0)
                {
                    Product product = new Product
                    {
                        Name = model.Name,
                        Price = model.Price,
                        Initial_Price = model.InitialPrice,
                        Sale_Percentage = model.SalePercentage,
                        ProductCategory = db.ProductCategories.Single(x => x.Id ==
model.CategoryId),
                        Date_From = model.DateFrom,
                        Date_To = model.DateTo
                    };

                    int userId = int.Parse((string)Session["UserId"]);
                    User user = db.Users.Single(x => x.Id == userId);
                    product.SuperMarket = db.SuperMarkets.Single(y => y.Id ==
user.Supermarket_Id.Value);

                    db.Products.Add(product);
                }
                else
                {
                    Product product = db.Products.Single(x => x.Id == model.Id);
                    product.Name = model.Name;
                    product.Price = model.Price;
                    product.Initial_Price = model.InitialPrice;
                    product.Sale_Percentage = model.SalePercentage;
                    product.ProductCategory = db.ProductCategories.Single(x => x.Id
== model.CategoryId);
                    product.Date_From = model.DateFrom;
                    product.Date_To = model.DateTo;
                    //Το supermarket δεν χρειάζεται γιατί δεν μπορεί να αλλάξει
                }
                db.SaveChanges();
            }

            return RedirectToAction("List");
```

```
        }

        public ActionResult Delete(int id)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                db.Products.Remove(db.Products.Single(x => x.Id == id));
                db.SaveChanges();
            }
            return RedirectToAction("List");
        }

        public ActionResult Search(ProductListModel model)
        {
            using (MarketappEntities db = new MarketappEntities())
            {
                User user = new User();
                if (!string.IsNullOrEmpty((string)Session["UserId"]))
                {
                    int userId = int.Parse((string)Session["UserId"]);
                    user = db.Users.Single(x => x.Id == userId);
                }

                List<Product> dbProducts = user.SuperMarket != null ?
db.Products.Where(x => x.SuperMarket.Id == user.SuperMarket.Id).ToList() :
db.Products.ToList();

                if (!string.IsNullOrEmpty(model.Name))
                    dbProducts = dbProducts.Where(x =>
x.Name.Contains(model.Name)).ToList();

                if (model.CategoryId != null)
                    dbProducts = dbProducts.Where(x => x.ProductCategory.Id ==
model.CategoryId).ToList();

                if (model.SupermarketId.HasValue)
                    dbProducts = dbProducts.Where(x => x.SuperMarket.Id ==
model.SupermarketId.Value).ToList();

                if (model.DateFrom.HasValue || model.DateTo.HasValue)
                {
                    if (model.DateFrom.HasValue && model.DateTo.HasValue)
                        dbProducts = dbProducts.Where(x => x.Date_From <=
model.DateFrom && x.Date_To >= model.DateTo).ToList();
                    else if (model.DateFrom.HasValue)
                        dbProducts = dbProducts.Where(x => x.Date_From <=
model.DateFrom).ToList();
                    else
                        dbProducts = dbProducts.Where(x => x.Date_To >=
model.DateTo).ToList();
                }

                model = new ProductListModel
                {
                    Products = dbProducts.Select(x => new ProductModel
                    {
                        Id = x.Id,
                        Name = x.Name,
```

```
                        Price = x.Price,
                        InitialPrice = x.Initial_Price,
                        SalePercentage = x.Sale_Percentage,
                        Category = x.ProductCategory,
                        DateFrom = x.Date_From,
                        DateTo = x.Date_To,
                        SuperMarketName = x.SuperMarket.Name
                    }).ToList(),
                    CategoriesList = db.ProductCategories.Select(x => new
SelectListItem { Text = x.Name, Value = x.Id.ToString() }).ToList(),
                    SupermarketsList = db.SuperMarkets.Select(x => new
SelectListItem { Text = x.Name, Value = x.Id.ToString() }).ToList()
                };
            }

            return View("List", model);
        }
    }
}
```

6.6.2 List (Product View)

```
@using SuperMarketOffers.Models
@model ProductListModel
@{
    ViewBag.Title = "Προϊόντα";
}
<h2>@ViewBag.Title</h2>
<h3>@ViewBag.Message</h3>

@using (Html.BeginForm("Search", "Product", FormMethod.Get))
{
    <div class="form-group">
        @Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2 "
})
        <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class
= "form-control" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.CategoryId, new { @class = "control-label col-
md-2 " })
        <div class="col-md-10">
            @Html.DropDownListFor(model => model.CategoryId, Model.CategoriesList,
"", new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateFrom, new { @class = "control-label col-
md-2 " })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateFrom, new { htmlAttributes = new {
@class = "form-control", @type = "date" } })
        </div>
    </div>
```

```
    <div class="form-group">
        @Html.LabelFor(model => model.DateTo, new { @class = "control-label col-md-2
" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateTo, new { htmlAttributes = new {
@class = "form-control", @type = "date" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.SupermarketId, new {@class = "control-label
col-md-2 "})
        <div class="col-md-10">
            @Html.DropDownListFor(model => model.SupermarketId,
Model.SupermarketsList, "", new {@class = "form-control"})
        </div>
    </div>
    <input type="submit" value="Αναζήτηση" class="btn btn-default" />
}
@if (Session["Username"] != null)
{
    @Html.ActionLink("Προσθήκη", "Edit", new { viewOnly = false }, new { @class =
"btn btn-primary" })
}

<div class="tab-content">
    <div class="tab-pane fade in active" id="productListView">
        <div class="table-responsive">
            <table id="productTable" class="table table-striped">
                <thead>
                    <tr>
                        <th class="hidden-xs hidden-sm">Κωδικός</th>
                        <th>Όνομα προϊόντος</th>
                        <th class="hidden-xs hidden-sm">Αρχική Τιμή</th>
                        <th>Τελική Τιμή</th>
                        <th class="hidden-xs hidden-sm">Ποσοστό έκπτωσης</th>
                        <th>Κατηγορία προϊόντος</th>
                        <th>Από</th>
                        <th>Έως</th>
                        <th>Supermarket</th>
                        @if (Session["Username"] != null)
                        {
                            <th>Ενέργειες</th>
                        }
                    </tr>
                </thead>
                <tbody>
                    @foreach (ProductModel product in Model.Products)
                    {
                        <tr class="form-group">
                            <td class="hidden-xs hidden-sm">@product.Id</td>
                            <td>@Html.ActionLink(@product.Name, "Edit", new {
product.Id, viewOnly = true }) </td>
                            <td class="hidden-xs hidden-
sm">@product.InitialPrice€</td>
                            <td>@product.Price€</td>
                            <td class="hidden-xs hidden-
sm">@(product.SalePercentage?.ToString("####,##"))%</td>
                            <td>@product.Category.Name</td>
```

```
                            <td>@(product.DateFrom?.ToShortDateString())</td>
                            <td>@(product.DateTo?.ToShortDateString())</td>
                            <td>@product.SuperMarketName</td>
                            @if (Session["Username"] != null)
                            {
                                <td>@Html.ActionLink("Επεξεργασία", "Edit", new
{product.Id, viewOnly = false}) |
                                    @Html.ActionLink("Διαγραφή", "Delete", new
{product.Id})</td>
                            }
                        </tr>
                    }
                </tbody>
            </table>
        </div>
    </div>
</div>
```

6.6.3 Edit (Product View)

```
@model SuperMarketOffers.Models.ProductModel
@{
    ViewBag.Title = !string.IsNullOrEmpty(Model.Name) ? $"{Model.Name}" : "Νέο
Προϊόν";
}
<h2>@ViewBag.Title</h2>

@using (Html.BeginForm("Save", "Product", FormMethod.Post))
{
    @Html.Hidden("Id", Model.Id)

    <div class="form-group">
        @Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2 "
})
        <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class
= "form-control" } })
            @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.CategoryId, new { @class = "control-label col-
md-2 " })
        <div class="col-md-10">
            @Html.DropDownListFor(model => model.CategoryId, Model.CategoriesList,
new { @class = "form-control" })
            @Html.ValidationMessageFor(model => model.CategoryId, "", new { @class =
"text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.InitialPrice, new { @class = "control-label
col-md-2 " })
        <div class="col-md-10">
            <div class="input-group">
```

```
                <span class="input-group-addon">€</span>
                @Html.EditorFor(model => model.InitialPrice, new { htmlAttributes =
new { @class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.InitialPrice, "", new { @class
= "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.SalePercentage, new { @class = "control-label
col-md-2 " })
        <div class="col-md-10">
            <div class="input-group">
                <span class="input-group-addon">%</span>
                @Html.EditorFor(model => model.SalePercentage, new { htmlAttributes
= new { @class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.SalePercentage, "", new {
@class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Price, new { @class = "control-label col-md-2
" })
        <div class="col-md-10">
            <div class="input-group">
                <span class="input-group-addon">€</span>
                @Html.EditorFor(model => model.Price, new { htmlAttributes = new {
@class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.Price, "", new { @class =
"text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateFrom, new { @class = "control-label col-
md-2 " })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateFrom, new { htmlAttributes = new {
@class = "form-control", @type = "date" } })
            @Html.ValidationMessageFor(model => model.DateFrom, "", new { @class =
"text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateTo, new { @class = "control-label col-md-2
" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateTo, new { htmlAttributes = new {
@class = "form-control", @type="date"} })
            @Html.ValidationMessageFor(model => model.DateTo, "", new { @class =
"text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.SuperMarketName, new { @class = "control-label
col-md-2 " })
        <div class="col-md-10">
```

```
            @Html.EditorFor(model => model.SuperMarketName, new { htmlAttributes =
new { @class = "form-control", @readonly = "readonly" } })
            @Html.ValidationMessageFor(model => model.SuperMarketName, "", new {
@class = "text-danger" })
        </div>
    </div>
    <input type="submit" value="Αποθήκευση" class="btn btn-primary" />
}
```

6.6.4 Index (Product View)

```
@model SuperMarketOffers.Models.ProductModel
@{
    ViewBag.Title = $"{Model.Name}";
}
<h2>@ViewBag.Title</h2>

<div>
    <div class="form-group">
        @Html.LabelFor(model => model.Category, new { @class = "control-label col-
md-2 " })
        @Model.Category.Name
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.InitialPrice, new { @class = "control-label
col-md-2 " })
        @Model.InitialPrice€
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.SalePercentage, new { @class = "control-label
col-md-2 " })
        @(Model.SalePercentage?.ToString("####,##"))%
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Price, new { @class = "control-label col-md-2
" })
        @Model.Price€
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateFrom, new { @class = "control-label col-
md-2 " })
        @(Model.DateFrom?.ToShortDateString())
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateTo, new { @class = "control-label col-md-2
" })
        @(Model.DateTo?.ToShortDateString())
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.SuperMarketName, new { @class = "control-label
col-md-2 " })
        @Model.SuperMarketName
    </div>
</div>
```

6.6.5 ProductModel

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SuperMarketOffers.DbObjects;

namespace SuperMarketOffers.Models
{
    public class ProductModel
    {
        public int Id { get; set; }
        [DisplayName("Όνομα")]
        public string Name { get; set; }
        [DisplayName("Τελική Τιμή")]
        public decimal? Price { get; set; }
        [DisplayName("Ποσοστό προσφοράς")]
        public decimal? SalePercentage { get; set; }
        [DisplayName("Προσφορά από")]
        public DateTime? DateFrom { get; set; }
        [DisplayName("Μέχρι")]
        public DateTime? DateTo { get; set; }
        [DisplayName("Κατηγορία προϊόντος")]
        public ProductCategory Category { get; set; }
        [DisplayName("Κατηγορία προϊόντος")]
        public int CategoryId { get; set; }
        public List<SelectListItem> CategoriesList { get; set; }
        [DisplayName("Αρχική τιμή")]
        public decimal? InitialPrice { get; set; }
        [DisplayName("Supermarket")]
        public string SuperMarketName { get; set; }
    }

    public class ProductListModel
    {
        public List<ProductModel> Products { get; set; }
        [DisplayName("Όνομα")]
        public string Name { get; set; }
        [DisplayName("Προσφορά από")]
        public DateTime? DateFrom { get; set; }
        [DisplayName("Μέχρι")]
        public DateTime? DateTo { get; set; }
        [DisplayName("Κατηγορία προϊόντος")]
        public ProductCategory Category { get; set; }
        [DisplayName("Κατηγορία προϊόντος")]
        public int? CategoryId { get; set; }
        public List<SelectListItem> CategoriesList { get; set; }
        [DisplayName("Supermarket")]
        public int? SupermarketId { get; set; }
        public List<SelectListItem> SupermarketsList { get; set; }
    }

}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

ADO.NET Entity Data Model Tools. (2011, Septempber 1). Ανάκτηση από Microsoft Developer Network: https://msdn.microsoft.com/library/91076853-0881-421b-837a-f582f36be527

C Sharp (programming language). (2018, February 4). Ανάκτηση από Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Entity Framework. (2017, November 29). Ανάκτηση από Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Entity_Framework

Git. (2018, January 26). Ανάκτηση από Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Git

Integrated development environment. (2018, January 16). Ανάκτηση από Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Integrated_development_environment

Microsoft Visual Studio. (2018, February 4). Ανάκτηση από Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Model-view-controller. (2018, January 29). Ανάκτηση από Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

RoMiller. (2016, July 26). *Introduction to Entity Framework*. Ανάκτηση από Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx#What%20is%20Entity%20Framework

Rouse, M. (2017, February). *database (DB)* . Ανάκτηση από Techtarget Network: http://searchsqlserver.techtarget.com/definition/database