

## Θέμα 1 - Refactor

Στο σύστημα παραγγελιών ενός ERP, η εφαρμογή των εκπτώσεων γίνεται αποκλειστικά από ένα sql query μέσα σε μια stored procedure. Η ανάγκη είναι να εξάγουμε τη λογική εφαρμογής εκπτώσεων σε κάποιο middle layer, με σκοπό να δομήσουμε επεκτάσιμο και testable κώδικα.

Δύο από τα είδη εκπτώσεων που διαθέτει το σύστημα αυτήν την στιγμή είναι εκπτώσεις Τιμοκαταλόγου (Price Lists) και Promotions. Θα θέλαμε όμως να έχουμε την δυνατότητα να δημιουργήσουμε ένα νέο είδος το οποίο θα ονομάζεται Κουπόνια (Coupons) για παράδειγμα.

Όλα τα είδη των εκπτώσεων έχουν ένα όνομα (Discount Name) και τύπο έκπτωσης (Discount Type) ανάλογα με τον οποίο αφαιρούν ποσοστό ή σταθερό ποσό από την αρχική τιμή. Επίσης, οι εκπτώσεις που δικαιούται ένας πελάτης εφαρμόζονται κατά σειρά προτεραιότητας με πιο σημαντική την Price List, έπειτα το Promotion και τέλος τα Κουπόνια.

## Απαιτήσεις

Ζητούμενο είναι να δομηθεί ένα υποσύστημα το οποίο θα είναι Unit Testable και δεδομένης μιας παραγγελίας θα εφαρμόζει όλες τις εκπτώσεις που δικαιούται, πάντα κατά σειρά προτεραιότητας.

Το κάθε είδος έκπτωσης (ένα είδος ή πολλαπλά) που μπορεί να εφαρμοστεί σε κάθε παραγγελία, μπορεί να είναι αποτέλεσμα ενός πολύπλοκου μηχανισμού, κατά τον οποίο απαιτούνται ερωτήματα και κλήσεις σε άλλες υπηρεσίες του συστήματος για να παραχθεί τελικά η έκπτωση.

Το αποτέλεσμα του υποσυστήματος αυτού θα είναι μια λίστα με αποτελέσματα εκπτώσεων και τελική τιμή, δηλαδή η ανάλυση των εκπτώσεων.

Παράδειγμα: Έστω ότι ένας πελάτης δικαιούται έκπτωση Price List 5%, Promotion 10% και ένα Κουπόνι -10€. Η συνδρομή για την οποία δικαιούται τις εκπτώσεις κοστίζει 340€/6μηνο. Το αποτέλεσμα (σε μια αφηρημένη μορφή) που θα περιμέναμε από το υποσύστημα εκπτώσεων θα ήταν

1. Price List Name: -17€ (5%)
2. Promotion Name: -32.3€ (10%)
3. Coupon Name: -10€

## Deliverable

Για το θέμα αυτό μπορεί να παραδοθεί ένα class diagram ή οποιοδήποτε άλλο διάγραμμα που θα απεικονίζει την αρχιτεκτονική του υποσυστήματος ή/και η παράδοση ενός εκτελέσιμου project.

## Answer

Το class diagram βρισκεται στο repository μαζι με το api που υλοποίει την λογική των εκπτώσεων.

Το end point api/Order/SetFinalPriceOfOrder που καλει την SetFinalPriceOfOrder υλοποίει την λογική της παραγωγής των εκπτώσεων σε μια παραγγελία(Order). Επιλεγεί τυχαία μια από τις τρεις κατηγορίες εκπτώσεων (για να προσομοιώσει τον πολύπλοκο μηχανισμό που επιλέγονται οι εκπτώσεις που θα εφαρμοστούν)- DiscountTypeTemplate που είναι αποθηκευμένα στην βάση.Στην συνέχεια δημιουργεί μια καινούργια έκπτωση ειδικά για αυτή την παραγγελία και την κατηγορία έκπτωσης που επιλέχθηκε. Το ποσοστό έκπτωσης επιλέγεται και αυτό τυχαία. Και μετα εφαρμόζονται οι εκπτώσεις με την σειρά(δεν έχει δημιουργηθεί μηχανισμός προτεραιότητας)

Το τελικό αποτέλεσμα είναι η τελική τιμή να εφαρμόζεται στην παραγγελία και δημιουργούνται όλες οι εκπτώσεις με τις τιμές έκπτωσης που αφαιρέθηκαν από την αρχική τιμή.

Το endpoint api/Discount/GetDiscountsByOrderId φερνει ολες τις εκπτώσεις που εφαρμόστηκαν στην εκάστοτε παραγγελίας μέσω του id της παραγγελίας

Επισης εχουν υπαρχουν end points για τις CRUD διαδικασίες του κάθε entity

Οι κλάσεις που αφορούν αυτή την λογική είναι (Χωρίς τις base κολάσεις και τα interfaces που κληρονομούν):

-Customer.cs

-Discount.cs

-DiscountType.cs

-DiscountTypeTemplate.cs

-Order.cs

-NewOrderDto

-CustomerRepository.cs

-DiscountRepository.cs

-DiscountTypeRepository.cs

-DiscountTypeTemplateRepository.cs

-OrderRepository.cs

-CustomerController.cs

-DiscountController.cs

-DiscountTypeController.cs

-DiscountTypeTemplateController.cs

-OrderController.cs

## Θέμα 2 – Δυναμικά πεδία σε καρτέλα πελάτη

Στο ίδιο ERP σύστημα, απαιτείται ο συνδεδεμένος χρήστης, να μπορεί να δημιουργεί κατά απαίτηση πεδία για την οντότητα του πελάτη. Τα πεδία αυτά θα πρέπει να έχουν τη δυνατότητα να δημιουργηθούν σαν απλά textboxes (μια απλή τιμή) ή σαν dropdown controls με πολλαπλές τιμές. Αφού δημιουργηθούν τα πεδία θα πρέπει να είναι διαθέσιμα σε όλες τις καρτέλες πελατών αλλά οι επιλογές των τιμών στα πεδία να αποθηκεύονται ανά πελάτη.

Επιπλέον, για κάθε αλλαγή που θα γίνεται στις τιμές οποιουδήποτε πεδίου ενός πελάτη θα πρέπει να διατηρούμε ιστορικό αλλαγών το οποίο θα μας εμφανίζει απαραιτήτως τις μεταβάσεις ανάμεσα στις διαδοχικές αλλαγές καθώς και τη χρονική στιγμή που συνέβη η εκάστοτε αλλαγή.

## Απαιτήσεις - Deliverable

Ζητείται ένα μοντέλο για την απεικόνιση των σχέσεων των αντικειμένων (ER) ή/και υλοποίηση σε κάποιο εκτελέσιμο project με τη χρήση data access τεχνολογίας της επιλογής σας.

## Answer

Το endpoint [GET]api/CustomerField/getHistory φέρνει όλα τα πεδία που έχουν το ίδιο guid και στην ουσία έτσι κρατάμε ιστορικό των αλλαγών για ένα πεδίο. Η οντοτητα CustomerField συνδέει πελάτη με πεδίο. Οποιαδήποτε αλλαγή (update-[HttpPost]-api/CustomerField) σε αυτή την οντότητα δημιουργεί μια καινούργια εγγραφή αντί να κάνει update ,αλλάζοντας την version του object CustomerField..

Οι κλάσεις που αφορούν αυτή την λογικη είναι (Χωρίς τις base κολάσεις και τα interfaces που κληρονομούν):

-Customer.cs

-Field.cs

-CustomerField.cs

-NewCustomerFieldDto.cs

-UpdateCustomerFieldDto.cs

-CustomerRepository.cs

-FieldRepository.cs

-CustomerFieldRepository.cs

-CustomerController.cs

-FieldController.cs

-CustomerFieldController.cs

Λοιπές πληροφορίες;

* Το project ακολουθεί το repository pattern μαζί με dependency injection με τις κοινές λειτουργικότητες των entities να ομαδοποιούνται σε γενικού σκοπού repositories.
* Για την αποθήκευση δεδομένων χρησιμοποιείτε first code approach με την χρήση του entity framework και για βαση Microsoft SQL server.
* Για το authentication/authorization γίνεται χρήση του jwt (επιστροφή ενός token στον χρήστη που αποθηκεύεται στα cookies)