Electron

Analysis and Design Document

Student: Czako Zoltan

**Group: 30235**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <18/04/2016> | <1.0> | Detailed description of Domain Model, Architecture, Components, Data model and Deployment | Czako Zoltan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 5

2. Architectural Design 5

2.1 Conceptual Architecture 5

2.2 Package Design 6

2.3 Component and Deployment Diagrams 7

III. Elaboration – Iteration 1.2 8

1. Design Model 8

1.1 Dynamic Behavior 8

1.2 Class Design 10

2. Data Model 11

3. Unit Testing 12

IV. Elaboration – Iteration 2 13

1. Architectural Design Refinement 13

2. Design Model Refinement 14

V. Construction and Transition 14

1. System Testing 14

2. Future improvements 15

VI. Bibliography 15

# Project Specification

In proiectul acesta voi realiza un e-commerce site, care are ca tema vanzari online de calculatoare, laptopuri, tablete, telefoane si alte dispozitive electronice.

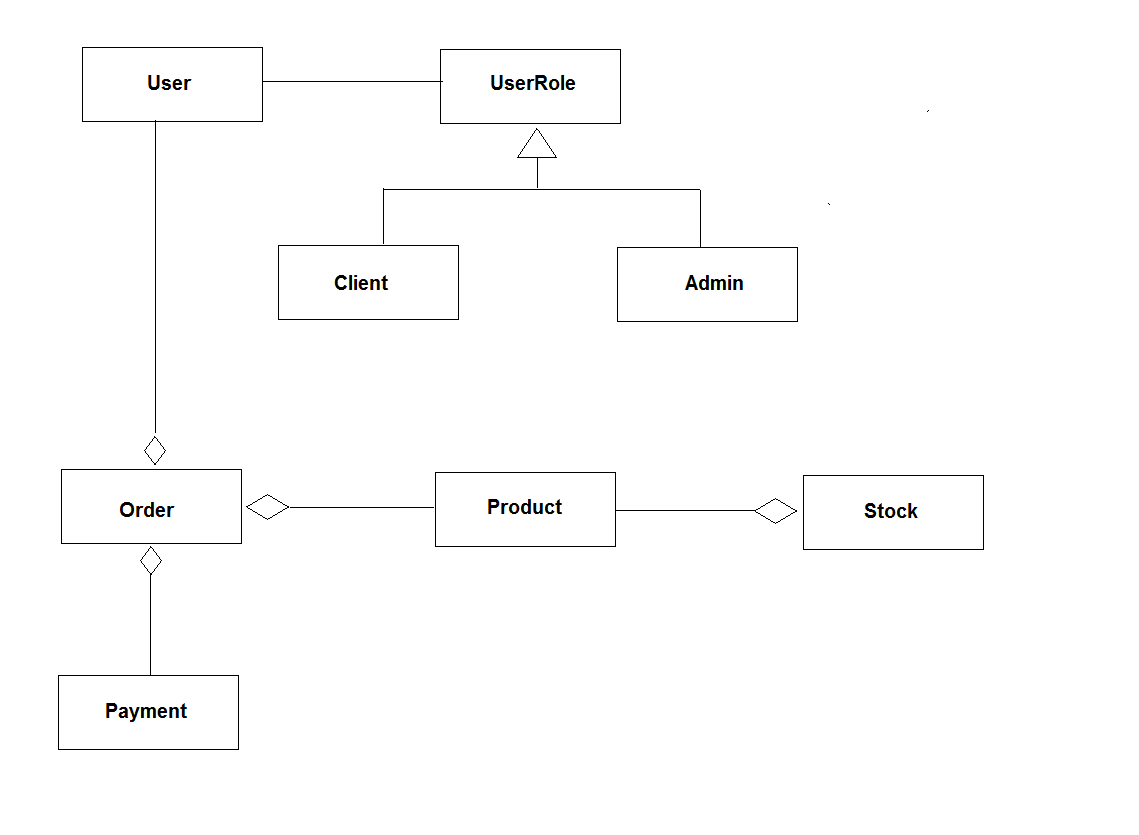
Va avea doua tipuri de utilizatori, utilizatorul normal, adica clientul, care poate sa vizualizeze produsele, poate sa caute produse dupa nume, dupa pret, dupa tehnologie etc, poate sa faca filtrare si poate sa-si cumpere produse. Pentru a cumpara produse, clientul trebuie sa se autentifice, iar daca nu are un cont activ, atunci poate sa-si creeze un cont gratis. Dupa logare, fiecare utilizator va avea un cos de vanzari pentru a realiza cumparaturi. Dupa adaugarea produselor in cosul de cumparaturi, pentru finalizarea cumparaturii, clientul trebuie sa aleaga o modalitate de plata, care poate fi prin credit card sau utilizand PayPal.

Al doilea tip de utilizator este adminul, care are mai multe drepturi decat un client normal. Adminul poate sa actualizeze datele produselor, poate sa stearga produse si sa accepte cumpararea produselor, se ocupa de introducerea produselor in stoc, actualizarea stocului, organizarea produselor si livrarea produselor cumparate.

In privinta implementarii acestui proiect vom folosi ca limbaj de baza C#, deoarece acest limbaj sta la baza proiectelor ASP.NET. Prin urmare backendul, adica logica de baza a proiectului va fi implementat in C#. In ceea ce priveste stocarea datelor, vom utiliza limbajul SQL pentru creearea bazei de date. Aceasta baza de date va fi stocat pe un cloud foarte renumit si anume Microsoft Azure. Pentru realizarea legaturii intre baza de date si proiectul nostru vom folosi Entity Framework-ul, care e un framework nou si foarte raspandit. Frontend-ul proiectului, adica interfata grafica intre utilizator si sistemul nostru va fi implementat utilizand tehnologii precum HTML5, CSS, Twitter Bootstrap si JQuery.

# Elaboration – Iteration 1.1

# Domain Model



# Architectural Design

## Conceptual Architecture

Arhitectura acestui site va fi MVC, deoarece acest sablon architectural are o serie de avantaje. Principalele avantaje sunt:

1. Separation of concerns (SoC): organizarea codului e foarte curat, granular, ceea ce usureaza implementarea cerintelor functionale
2. Permite o integrare usoara ale intrumentelor client side: utilizarea usoara a toolkit-urilor ca si JQuery, Bootstrap etc.
3. Search Engine Optimization (SEO) Friendly / Stateless
4. Grad de control foarte ridicat: developerul poate sa controleze cu usurinta randarea UI-ului, are un control mare asupra HTML.
5. Test driven development (TDD)

## DeploymentElectron.pngPackage Design

## Component and Deployment Diagrams

# DeploymentDiagram.png

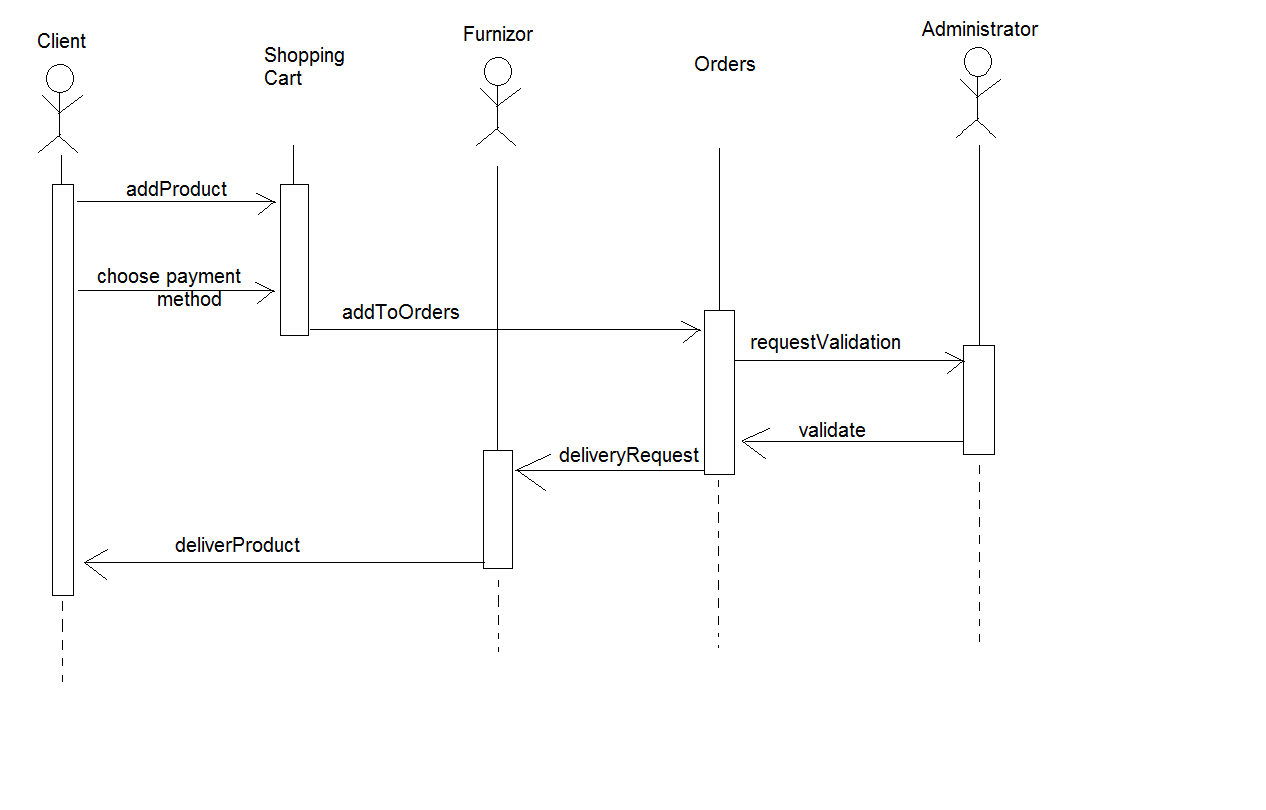
# Elaboration – Iteration 1.2

# Design Model

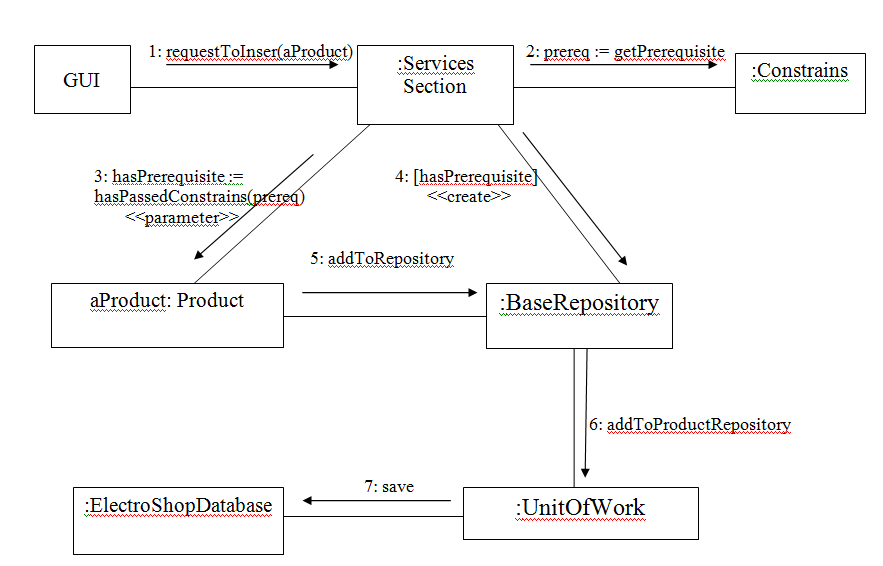
## Dynamic Behavior

## RegisterClientActivity.pngDiagrama de activitate pentru crearea unui produs nou

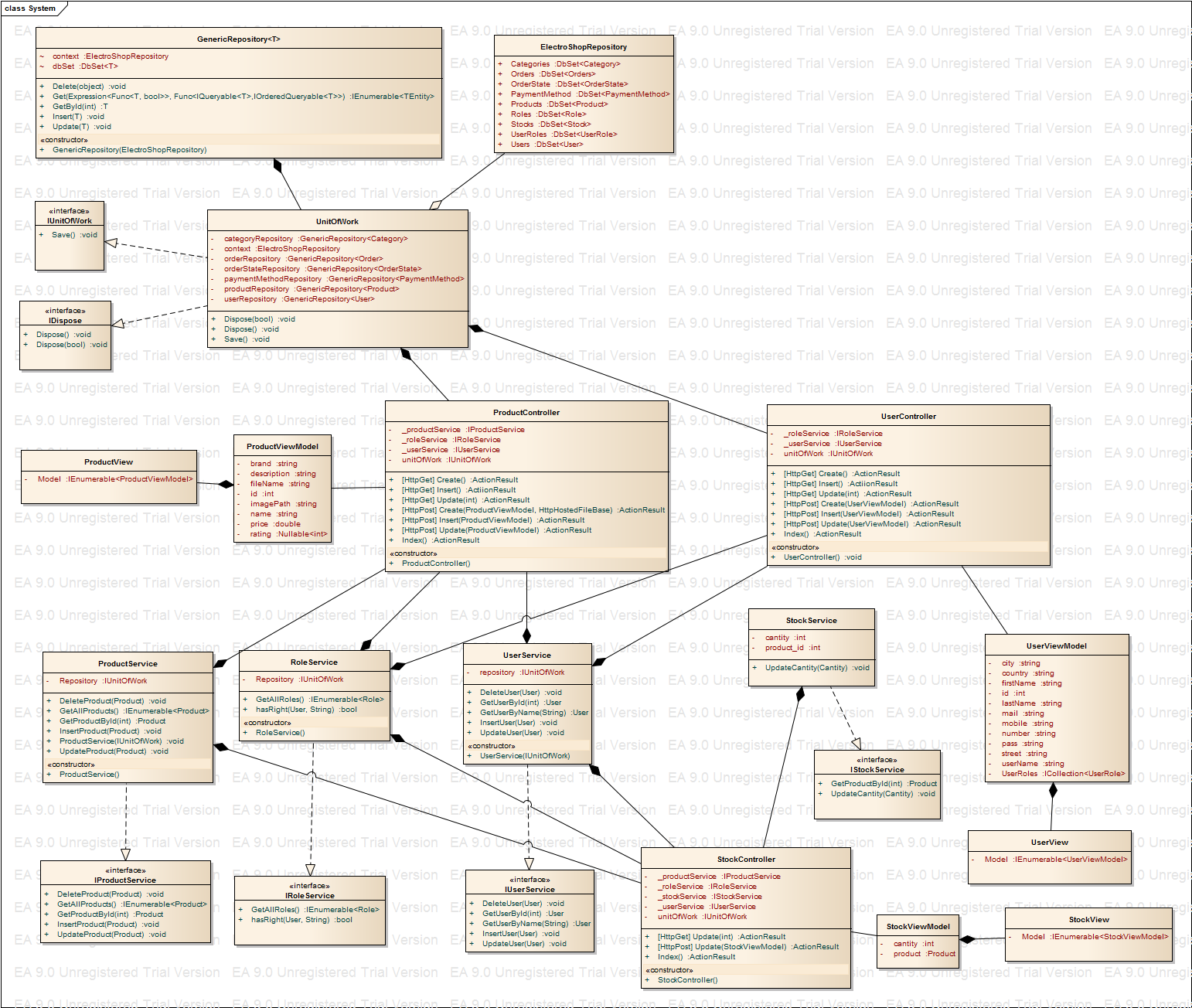
## Diagrama de secvență pentru cumpărarea unui produs



## Diagrama de comunicare pentru introducearea unui produs nou



## Class Design



# Data Model

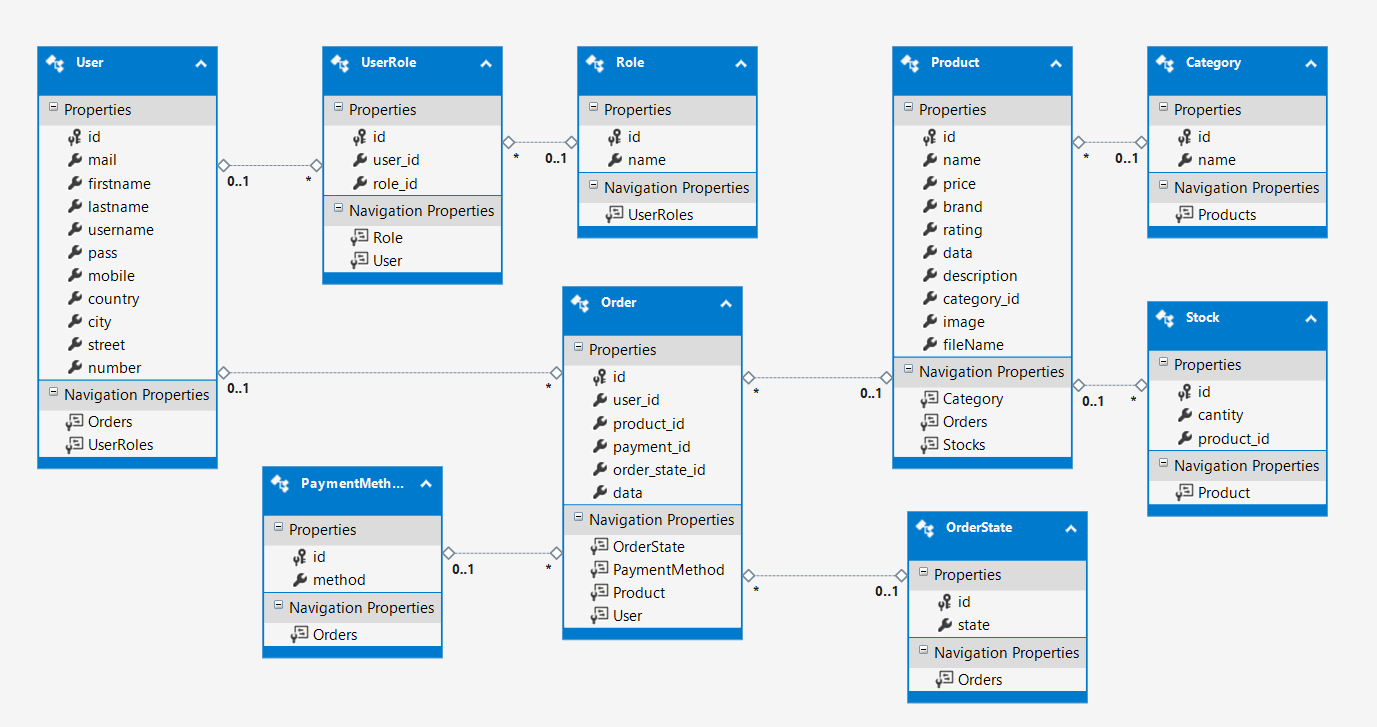
Pentru a realize un design cât mai curat și reutilizabil, la nivelul inferior, aproape de baza da date vom folosi Patternul de Generic Repository și Unit Of Work, astfel modelând cât mai corect modelul de date, iar fiecărei table îi corespunde o clasă și în aplicația noastră vom folosi doar o singură instanță pentru fiecare tabel, adică vom folosi patternul numit Sigletone.

Următoarea figură arată structura pentru un Generic Repository și un Unit of Work:



Așaa cum se vede și în imaginea de mai sus, fără acest Pattern Controllerul nostru ar fi fost strâns legat de baza de date, deci orice modificare a bazei de date poate strica toată aplicația. Cu un Unit Of Work separăm controllerul și logica de baza de date, astfel putem modifica baza noastră de date, fără a avea nevoie să modificăm și logica din Servicii și din Controller.

Pentru crearea bazei de date am folosit SQL-ul si conexiunea este facuta de catre Entity Framework, un framework foarte raspandit si creat de firma Microsoft.

 Diagrama bazei de date:

# Unit Testing

Pentru testarea sistemului nostrum vom folosi Unit Teste și vom creea date mock, folosind frameworkul de Mock. Pentru aceasta vom avea nevoie de interfețe pentru fiecare Serviciu, Controller, adică pentru fiecare funcționalitate. Toate componentele aplicației vor fi legate împreună utilizând interfețe, astfel vom avea un proiect loosely coupled și ușor de modificat, adaptabil, reutilizabil și ușor de menținut.

Pentru ca toate Seviciile și Controllerele să fie independente unul de celălalt, vom folosi Dependency Injection, iar pentru aceasta vom folosi Unity.

Vom încerca o acoperire cât mai largă, deci vom scrie teste atât pentru happy-flow cât și pentru corner cases. Vom testa și validările datelor introduse, astfel testând logica de business.

# Elaboration – Iteration 2

# Architectural Design Refinement

# SistemDiagramm.png

La baza acestei aplicatii sta un SQL Database. Legatura intre baza de date si aplicatia noastra (ASP.NET MVC Web Application) se face cu ajutorul lui Entity Framework, dezvoltat de catre Microsoft. Urmatorul nivel este o legatura prin interfete intre contextual bazei de date si aplicatia noastra. Aceasta legatura cu interfete asigura un nivel mare de reutilizabilitate, deoarece urmatoarele nivele pot fi inlocuite cu alte tehnologii atat timp cat tehnologiile implementeaza interfelete. In centrul aplicatiei putem observa un alt design pattern, si anume Unit of Work care implica folosirea Patternului de Generic Repository si fiecare deposit concret(Specific Repository) este unic, deci implica folosirea Singletone-ului. Astfel la acest nivel avem trei tipuri de Design Pattern, contribuind la o arhitectura foarte curata si reutilizabila. Controllerul, unde se proceseaza toate evenimentele, toate cererile utilizatorilor foloseste Servicii pentru a accesa nivelele mai inferioare. Pentru legatura intre Servicii si nivelul inferior introducem interfete pentru ierarhizare si pentru reutilizare a proiectului. La nivelul cel mai inalt se afla interfata grafica, care ofera o interfata pentru comunicare intre utilizatori si sistemul nostru.

Motivul principal pentru care am ales această arhitectură este mentabilitatea, deaorece arhitectura MVC este perfectă pentru site-uri și alte aplicații de business, în care fiecare zi există schimbări, MVC-ul este deschis la schimbări, e ușor de modificat, reutilizat etc.

Un design patter, care va fi folosit în aplicația noastră e Command Pattern, deaorece clienții vor avea posibilitatea de Undo la o cumpărare nedorită timp de 30 de zile de la cumpărare.

Administratorul va genera rapoarte despre produsele vândute sau în stoc, deci vom folosi Factory Patternul pentru a creea mai multe tipuri de repoarte (.xml, .doc, .xls etc.)

# Design Model Refinement

# Construction and Transition

# System Testing

Testarea software reprezintă o investigație empirică realizată cu scopul de a oferi părților interesate informații referitoare la calitatea produsului sau serviciului supus testăriiluând în considerație contextul operațional în care acesta din urma va fi folosit. Testarea software pune la dispoziție o viziune obiectivă și independentă asupra produsului în dezvoltare, oferind astfel businessului posibilitatea de a înțelege și evalua riscurile asociate cu implementarea produsului soft.

Tehnicile de testare includ, dar nu sunt limitate la, procesul de execuție a programului sau aplicației în scopul identificării defectelor/erorilor de software. Testarea software mai poate fi definită ca un proces de validare și verificare a faptului că un program/aplicație/produs software corespunde business cerințelor și cerințelor tehnice care au ghidat proiectarea și implementarea lui; și rulează și se comportă corespunzător așteptărilor.

În dependență de metodologia de testare aleasa, testarea software poate fi implementată la orice etapă în cadrul procesului de dezvoltare, deși partea considerabilă a efortului de testare este aplicată de obicei la etapa de după cizelarea/formalizarea cerințelor și finisarea implementării/codării propriu-zise.

# Deci pentru testarea sistemului nostru, vom urmări și vom realize toate acțiunile posibile unui utilizator și pentru taote tipurile de utilizatori. Astfel pentru clienți vom testa cumpărarea produselor, anularea cumpărării, plata cu card bancar și apoi cu PayPal și căutarea avansată, filtrearea produselor și acordarea calificativelor produselor.

Ca și admin, vom testa ștergerea, editarea, introducerea produselor, creare raporturilor, acceptarea vânzării.

Furnizorul nu are acces, doar la stocul firmei, deci vom testa modificarea stocului.

# Future improvements

Acest proiect se poate înbunătății cu o interfață grafică mult mai interactivă, cu animații și cu imagini de calitate mai bună, deaorece în business, mai ales în vânzări e foarte important designul, cât e mai frumos cu atât mai bine pentru firmă, pentru că va avea mai mulți clienți.

# Bibliography

[1] <http://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application/>

[2] <https://en.wikipedia.org/wiki/System_testing>

[3] <https://msdn.microsoft.com/en-us/library/ff649643.aspx>

[4] <https://msdn.microsoft.com/en-us/data/ef.aspx>

[5] <https://msdn.microsoft.com/en-us/library/hh323705(v=vs.100).aspx>