### UNIVERSITATEA DIN BUCUREȘTI

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

**Lucrare de licență**

**Aplicație web tip magazin online**

### Absolvent

**Paraschiv**

**Alexandru-Andrei**

**Coordonator științific**

**Lect.dr. Sipoș Andrei-Valentin**

**București, iunie 2022**

#### Rezumat

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce vitae eros sit amet sem ornare varius. Duis eget felis eget risus posuere luctus. Integer odio metus, eleifend at nunc vitae, rutrum fermentum leo. Quisque rutrum vitae risus nec porta. Nunc eu orci euismod, ornare risus at, accumsan augue. Ut tincidunt pharetra convallis. Maecenas ut pretium ex. Morbi tellus dui, viverra quis augue at, tincidunt hendrerit orci. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam quis sollicitudin nunc. Sed sollicitudin purus dapibus mi fringilla, nec tincidunt nunc eleifend. Nam ut molestie erat. Integer eros dolor, viverra quis massa at, auctor.

#### Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce vitae eros sit amet sem ornare varius. Duis eget felis eget risus posuere luctus. Integer odio metus, eleifend at nunc vitae, rutrum fermentum leo. Quisque rutrum vitae risus nec porta. Nunc eu orci euismod, ornare risus at, accumsan augue. Ut tincidunt pharetra convallis. Maecenas ut pretium ex. Morbi tellus dui, viverra quis augue at, tincidunt hendrerit orci. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam quis sollicitudin nunc. Sed sollicitudin purus dapibus mi fringilla, nec tincidunt nunc eleifend. Nam ut molestie erat. Integer eros dolor, viverra quis massa at, auctor

# Cuprins

1. [Introducere](#_bookmark0) 4
   1. [Motivație](#_bookmark1) 4
   2. [Dolor sit amet](#_bookmark2) 5
      1. [Quis tellus vitae](#_bookmark3) 5
2. [Preliminarii](#_bookmark4)
3. Contribuție
4. [Concluzii](#_bookmark5) 7

[Bibliografie](#_bookmark6) 8

**Capitolul 1**

# Introducere

## 

* 1. **Motivație**

Pentru realizarea proiectului de licență am ales să dezvolt o aplicație web de tip magazin online, care va ajuta o firmă ce se ocupă de comerțul cu odorizante, de diferite tipuri. Această firmă a avut în trecut un *site* care nu avea o interfață ușor de înțeles pentru orice tip de utilizator, nu avea detalii relevante în legătură cu produsele și nici funcționalități. Consider că proiectul creat poate ajuta în proporții foarte mari firma, atât printr-un *design* mult mai plăcut pentru potențialii clienți, cât și prin organizarea mult mai ușoară a comenzilor efectuate.

**1.2 Scopul temei**

Tema abordată are ca scop realizarea unei aplicații web în ASP.NET MVC, folosind noțiunile învățate în cei trei ani de facultate, atât din materiile predate, cât și din lectura individuală. Prin dezvoltarea programului, am pus în practică cunoștințele acumulate în legătură cu planificarea unui proiect: am purtat discuții cu firma respectivă pentru a definitiva cerințele în funcție de importanța acestora, am ales tehnologiile, am schițat arhitectura, am implementat *backendul*, *frontendul* și am populat baza de date cu itemi relevanți.

**Capitolul 2**

# Preliminarii

**2.1 ASP.NET MVC**

ASP.NET MVC este un *framework* construit pentru crearea de aplicații web, dar și a API-urilor. Arhitectura MVC se folosește de trei componente principale, bine legate între ele: *Modelul, View-ul* și *Controllerul.* Ca și conexiune, *Modelul* nu depinde de nimeni, *View-ul* depinde de *Controller*, iar, în final, atât *Controllerul* cât și *View-ul* depind de *Model.* Interacțiunea cu utilizatorul este realizată de către *Controller* care preia date din *Model* pentru a le afișa în *View-ul* corespunzător.[1]

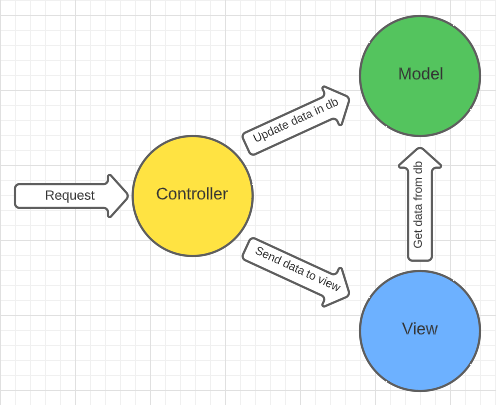


Figura 2.1 – Arhitectura MVC

**2.1.1 Model**

*Modelul* reprezintă ansamblul tuturor claselor ce depind de scopul aplicației.[2] Datele sunt stocate într-o bază de date, gestionate și modificate în urma cererilor formulate de *View-uri* și *Controllere*.

**2.1.2 View**

*View-ul* afișează datele din *Model*, pe ecran, prin intermediul HTML. Acesta poate conține și cod JavaScript, iar stilizarea este construită prin CSS. Se mai pot alătura și diferite *framework-uri* precum Bootstrap, Angular sau librării specifice de JavaScript ca și React. Pe lângă datele din *Model*, *View-ul* poate afișa orice informație prezentă în cadrul aplicației, deci acesta face legătura dintre utilizator și program. Există *View-uri* legate de către un anumit *request* sau acțiune dintr-un *Controller* și *View-uri* de tip *Partial*. Cele de tip *Partial* definesc bucăți dintr-un *View* ce pot fi refolosite în una sau mai multe pagini.

**2.1.3 Controller**

*Controllerul* preia *requesturi* de tip HTTP, scrise în cod C#. Anumite instrucțiuni sunt date de către utilizator, *Controllerul* pasează aceste date către *Model*, iar în final este selectat *View-ul* corespunzător cu informațiile cerute. În ASP.NET MVC, fiecare *Controller* are în spatele său o clasă, iar regula de numire vizează ideea *ClasăController*, fiind necesară uneori forma de plural al numelui clasei. De exemplu, pentru aplicația dezvoltată, am creat în urma *Modelului Freshener*, *Controllerul FreshenersController.*

**2.2 .NET Framework și C#**

**2.2.1 .NET Framework**

.NET Framework este creat pentru Windows de către Microsoft. Acesta folosește diferite limbaje de programare precum C#, F# sau Visual Basic. .NET Framework este alcătuit din două părți: *Common Language Runtime* (CLR) și *Base Class Library.* CLR execută programul, iar *Base Class Library* conține clasele din .NET. [3]

**2.2.2 C#**

Limbajul C# este unul de tip compilat, codul sursă scris fiind transformat de către compilator în cod mașină. Dacă nu există erori de compilare după această transformare, din codul mașină se produce cod executabil și se rulează programul. C# este o derivare a limbajului C++, deci este și orientat pe obiect. [3]

**2.3 NuGet, Entity Framework și Migrații**

**2.3.1 NuGet**

Microsoft a dezvoltat o platformă numită *NuGet* prin care pot fi descărcate și instalate, actualizate sau șterse diferite librării, fie scrise de utilizatori, fie de companii. Cele mai populare librării din NuGet sunt: *Newtonsoft.Json, Microsoft.Extensions.DependencyInjection, Serilog, EntityFramework*.

**2.3.2 Entity Framework**

Entity Framework facilitează legarea obiectelor din *Modele* cu tabelele unei baze de date relaționale. Entity Framework permite programatorilor să folosească date ale unor obiecte din clase definite, fără să își pună problema schimbării bazei de date.[4] În aplicația dezvoltată am folosit versiunea EF6, mai exact 6.2.0.

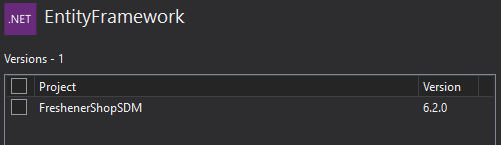
****

Figura 2.2 – Versiune Entity Framework

**2.3.3 Migrații**

Migrațiile sunt de două tipuri: *Code First Migration* (Migrații Automate) și *Code Based Migration*.

Migrațiile de tip *Code First* dau voie Entity Framework-ului să realizeze migrații automate, atunci când apar modificări asupra bazei de date. Atunci când aplicația este rulată, Entity Framework observă acele schimbări și face un *updatează* baza de date la ultima versiune.

Cel de-al doilea tip de migrații, *Code Based*, nu realizează acele schimbări în mod automat. Este necesar un cod adițional explicit, această variantă fiind des întâlnită atunci când se dorește un *upgrade* sau *downgrade* la o anumită versiune.[3]

**2.4 Componente și facilități ale unui View**

**2.4.1 HTML, CSS, JavaScript**

Cele trei sunt componentele de bază pentru *frontendul* unei aplicații web. În ASP.NET MVC, fișierele de CSS și JavaScript pot fi grupate în *bundle-uri* și încărcate ușor în pagina de HTML prin metoda *Render()*. În continuare voi atașa o secvență de cod ce reprezintă unul din *bundle-urile­* definite în aplicație. Acest *bundle* cuprinde două fișiere de bootstrap, un fișier numit *site.css* ce conține cod CSS și un fișier pentru font-awesome. Elementele prezentate de bootstrap și font-awesome sunt descărcate automat prin platforma NuGet.

bundles.Add(new StyleBundle("~/Content/css").Include(

"~/Content/bootstrap.css",

"~/Content/bootstrap.min.css",

"~/Content/site.css",

"~/Content/font-awesome.css"));

**2.4.2 Bootstrap**

Bootstrap este unul dintre cele mai populare și importante *framework-uri* de CSS. Utilizarea acestuia conferă aplicației un aspect modern și totodată atinge și partea de *responsive*. În proiectul meu am folosit versiunea de Bootstrap 4.6.1, ultima versiune de Bootstrap 4, dar nu și cea mai nouă, existând Bootstrap 5.

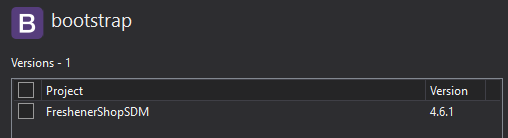


Figura 2.3 – Versiune Bootstrap

**2.4.3 ASP.NET Razor**

Razor oferă posibilitatea utilizatorilor de a introduce cod C# printre *tagurile* de HTML. Prin utilizarea simbolului *@* se începe o secvență de cod *server-side*.

**2.4.4 Helpere**

*ViewBag* este un *helper* care, împreună cu simbolul *@* specific ASP.NET Razor, ajută *Controllerul* să trimită date către *View*, în special obiecte. Un aspect important este acela că datele trimise nu se regăsesc in *Model*.

*ViewData* este un alt *helper*, asemănător cu *ViewBag*. Diferența este dată de faptul că *ViewData* este reprezentat de un dicționar.

*Helperul model* facilitează transmiterea datelor dintr-un anumit *Model* la un anumit *View*.

*TempData* poate redirecționa o valoarea dintr-o anumită acțiune într-alta, fiind disponibilă doar la prima accesare.

**2.5 Lucidchart**

Lucidchart este o platformă online care ajută utilizatorii în crearea de diagrame, în funcție de tipul dorit. Astfel, Lucidchart oferă posibilitatea de a modela diagrame entitate-relație, pentru scheme logice de Kubernetes, AWS, Cisco și multe altele. Serviciile platformei pot fi folosite atât în varianta gratuită, cât și contra cost. Pentru aplicația mea am folosit varianta gratuită, fiind suficientă pentru îndeplinirea cerințelor mele.

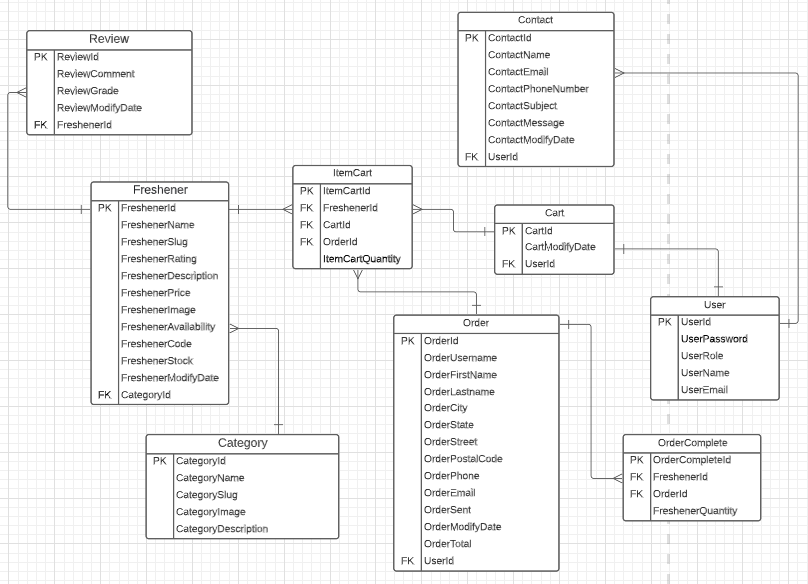


Figura 2.4 – Diagrama aplicației în Lucidchart

**Capitolul 3**

# Contribuție

**3.1 Abordări existente**

Aplicația este realizată în ASP.NET MVC, o tehnologie ce facilitează dezvoltarea programelor complete, ușurează *debugging-ul*, are o comunitate bine definită, iar numeroasele librării ce pot fi accesate de către dezvoltatori sunt însoțite de *update-uri* regulate. În prezent, domeniul actual pune la dispoziție si alte tehnologii, precum WordPress, Laravel sau Django, însă acestea au diferite minusuri importante. Spre exemplu, aplicațiile de tip WordPress au numeroase *template-uri*, dar flexibilitatea în schițarea propriului *design* lipsește. Laravel, la fel ca și alte *framework-uri* de PHP, scad în popularitate, librăriile lor nemaiprimind la fel de mult suport tehnic. Iar Django poate avea probleme în viteza limbajului, având la bază Python, un limbaj interpretat (există un interpretor care schimbă fiecare linie de cod în cod mașină și îl execută).

**3.2 Fluxuri de utilizare**

În prezent, aplicația dispune de trei fluxuri de utilizare, fiecare definind un mod de lucru des întâlnit pe *site-urile* de comercializare produse: utilizator neînregistrat, utilizator înregistrat și admin.

**3.2.1 Fluxul pentru utilizatorul neînregistrat**

Acest flux este destinat persoanelor care descoperă recent *site-ul* și doresc să vadă produsele și categoriile din care fac parte. *Review-urile* articolelor din aplicație pot fi și acestea văzute, alături de paginile de *Home, About* și *Contact*. Lista de produse poate fi modificată în funcție de sortarea aleasă, mai exact crescător sau descrescător după preț, *rating* sau data adăugării. De asemenea, poate fi folosită și funcționalitatea de *search* ce filtrează rezultate în funcție de nume sau descriere. Atunci când un utilizator neînregistrat dorește să lase un *review*, să trimită un formular de contact sau să adauge în coșul de cumpărături un articol, este redirecționat către pagina de *Login*, care duce către următorul flux de utilizare.

**3.2.2 Fluxul pentru utilizatorul înregistrat**

Odată cu înregistrarea utilizatorului pe platforma web, acesta se poate bucura de un număr mult mai mare de funcționalități. Pot fi trimise formulare de contact și pot fi lăsate și editate *review-uri*. Utilizatorul are dreptul de a adăuga produse în coșul de cumpărături, de a le modifica numărul sau de a le șterge din acesta. După stabilirea finală a listei din coșul de cumpărături, *user-ul* este redirecționat către un formular pentru completarea adresei de livrare, în final afișându-i-se un mesaj de mulțumire, urmat de numărul comenzii. Totodată, pentru acest flux de utilizare este disponibil și un meniu special care afișează lista de comenzi, alături de detaliile fiecăreia. Aceste comenzi pot fi anulate doar dacă administratorul aplicației nu a trimis deja comanda, în caz contrat apărând un text de informare.

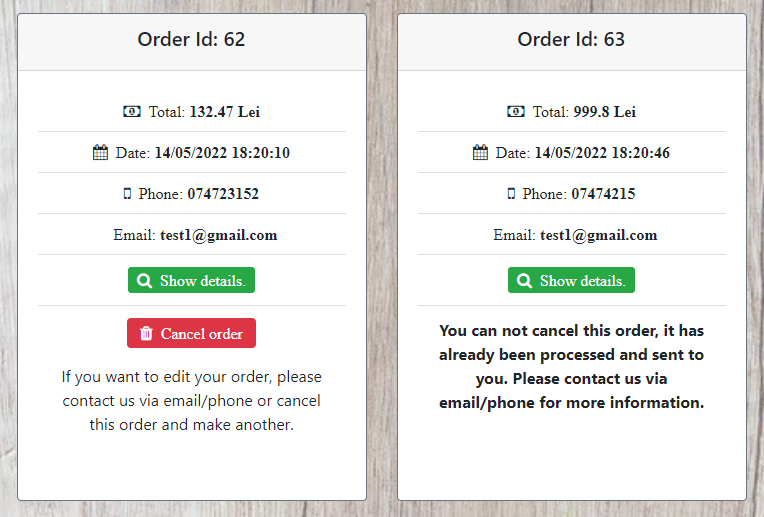


Figura 3.1 – Două comenzi plasate de un utilizator înregistrat

**3.2.3 Fluxul pentru admin**

Fluxul pentru admin conține toate funcționalitățile utilizatorului înregistrat, la care se alătură și altele noi. Dreptul de a adăuga, edita sau șterge atât produse cât și categorii, dreptul de a șterge sau edita *review-urile* oricărui utilizator (utilizatorul înregistrat poate edita doar *review-urile* proprii) sunt idei create doar pentru admin. Acesta dispune și de un meniu special, de unde poate vedea lista tuturor comenzilor plasate, cu anumite detalii relevante, și le poate șterge, edita sau marca ca și *trimis*. Tot în meniul descris se mai găsește și o listă a tuturor utilizatorilor, adminul putând să îi șteargă, să le editeze unele date și să îi promoveze. Lista tuturor formularelor de contact este și ea prezentă și oferă posibilitatea adminului de a vedea toate detaliile trimise și de a le șterge. În final, fiind o funcționalitate des folosită, adminul dispune și de un buton pentru adăugarea unui produs nou pe platforma web.

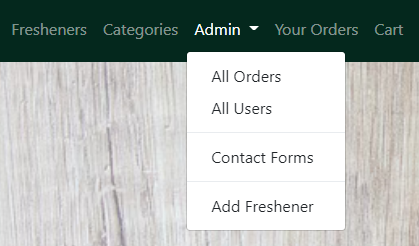


Figura 3.2 – Meniu special pentru admin

**3.3 Funcționalitățile aplicației**

**3.2.1 Modelele create**

Modelul principal al aplicației se numește *Freshener* și conține date relevante pentru fiecare produs afișat pe *site* precum: nume, *slug* (ajută partea de URL a unui *site*, de obicei este numele produsului, iar între cuvinte se află cratime), descriere, preț, *rating*, adresa către o imagine, disponibilitate, cod, stoc, data adăugării în baza de date, categoria din care face parte și o listă de *review-uri*. Datele cele mai importante ale unui produs au validări, nepermițând necompletarea acestora atunci când sunt adăugate în baza de date. Lista de *review-uri* este definită de tip *ICollection*, care pe lângă tipul *IEnumerable* ce conține doar metodele de *get* și *loop*, adaugă funcționalități de ștergere, numărare, verificare și adăugare. Pe lângă cheia primară incrementată automat *FreshenerId*, modelul definește și o cheie externă pentru categoria din care face parte și una pentru utilizatorul ce a adăugat produsul (*CategoryId* și *UserId*). Mai sunt prezente liste de legătură cu tabelele *ItemCart* și *OrderComplete* ce vor fi prezentate ulterior.

Modelul pentru categorii este format din id, nume, *slug*, adresa imaginii, descriere și o listă de produse, existând o relație *one-to-many* între categorie și produs, iar fiecare câmp descris are și validări.

În continuare este prezentat modelul *Contact*, construit pentru a stoca anumite formulare cu întrebări sau opinii adresate de către utilizatori. Fiecare formular este alcătuit dintr-un id incrementat automat și care nu este vizibil în partea de *frontend*, numele utilizatorului, o adresă de email pe care dorește să fie contactat, putând fi diferită de cea folosită la înregistrarea în aplicație. Tot aici se mai poate completa și un număr de telefon, subiectul mesajului cât și conținutul acestuia, iar data completării formularului este preluată automat. Pentru a evita *spam­ul*, am ales să permit trimiterea formularului de contact doar utilizatorilor înregistrați, iar prin folosirea validărilor, toți itemii prezentați trebuie completați.

Pentru a strânge în baza de date *review-urile* unui produs, am dezvoltat un model ce reține comentariul, nota într-un interval de la unu la cinci și data adăugării pe *site*. Fiecare *review* este legat la un singur obiect de tip *Freshener* și un utilizator, iar un produs poate avea mai multe *review-uri*.

Funcționalitatea de plasare a comenzilor este inițiată în modelul *Cart*, unde fiecărui utilizator îi este atribuit câte un coș de cumpărături și o listă de produse. Această listă este ulterior definită în modelul *ItemCart*, fiind alcătuită din id, produs, cantitatea acelui tip de produs și o referință către modelul *Order*. În tabela menționată se găsesc datele pentru livrarea comenzii: *username-ul*, numele, prenumele, orașul, județul, strada, codul poștal, numărul de telefon, emailul, totalul comenzii și statusul de comandă (dacă a fost sau nu trimisă). Pentru validarea numărului de telefon, cât și a emailului, am folosit *regex-uri*[5]. Mai jos este afișat *regex-ul* pentru numărul de telefon și explicația lui, sub formă de tabel.

[RegularExpression(@"^(\+4|)?[0-9]{6,}", ErrorMessage = "This phone number is not valid.")]

public string OrderPhone { get; set; }

|  |  |
| --- | --- |
| Element din regex | Explicație |
| *^* | anunță *regex-ul* să pornească de la începutul liniei |
| *(\+4|)* | potrivește combinația *+4* la începutul unui număr de telefon |
| *?* | stabilește ca partea de *+4* să apară cel mult o dată |
| *[0-9]* | ia un singur caracter din intervalul dat, adică un număr între zero și nouă |
| *{6,}* | repetă alegerea de caracter anterioară pentru cel puțin șase ori |

Pentru a putea stoca permanent atât adresele de livrare, cât și produsele achiziționate, am implementat un nou model numit *OrderComplete* ce reține pentru fiecare OrderId, pe câte un rând separat, id-ul produsului și cantitatea setată.

În *IdentityModels* este definit fiecare model creat, un vector de tip *IEnumerable* pentru tipurile de roluri, cât și inițializarea bazei de date la migrații. Mai jos este codul pentru fiecare model.

public DbSet<Freshener> Fresheners { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Review> Reviews { get; set; }

public DbSet<Cart> Carts { get; set; }

public DbSet<Order> Orders { get; set; }

public DbSet<ItemCart> ItemCarts { get; set; }

public DbSet<Contact> Contacts { get; set; }

public DbSet<OrderComplete> OrderCompletes { get; set; }

**3.2.2 Controllerele create**

**Capitolul 4**

# Concluzii

**Capitolul 5**

**Bibliografie**

[1] Steve Smith, *Overview of ASP.NET Core MVC*, <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0>, accesat pe 15 mai 2022.

[2] Sipoș

[3] Cezara

[4] *Entity Framework 6*, <https://docs.microsoft.com/en-us/ef/ef6/>, accesat pe 15 mai 2022.

[5] Jon Galloway, *Part 9: Registration and Checkout*, <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/mvc-music-store/mvc-music-store-part-9>, accesat pe 24 mai 2022.