### 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. [Lorem ipsum](#_bookmark1) 4
   2. [Dolor sit amet](#_bookmark2) 5
      1. [Quis tellus vitae](#_bookmark3) 5
2. [Preliminarii](#_bookmark4) 6
3. [Concluzii](#_bookmark5) 7

[Bibliografie](#_bookmark6) 8

**Capitolul 1**

# Introducere

## 

**1.1 Motivație**

Tema abordată are ca scop realizarea unei aplicații web pentru o firmă de odorizante, folosind noțiunile învățate în cei trei ani de facultate, atât din materiile predate, cât și din lectură 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.

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 ne mai primind la fel de mult suport tehnic. Iar Django poate avea probleme în viteza limbajului, având la bază Python, un limbaj interpretat.

**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]

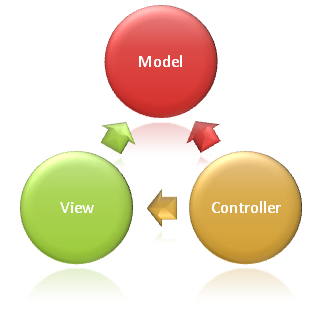


Figura 2.1 – Arhitectura MVC

**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 1 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.

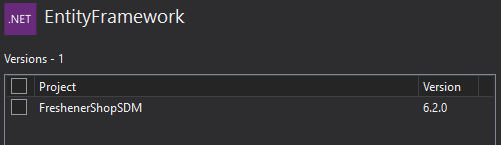
****

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()*.

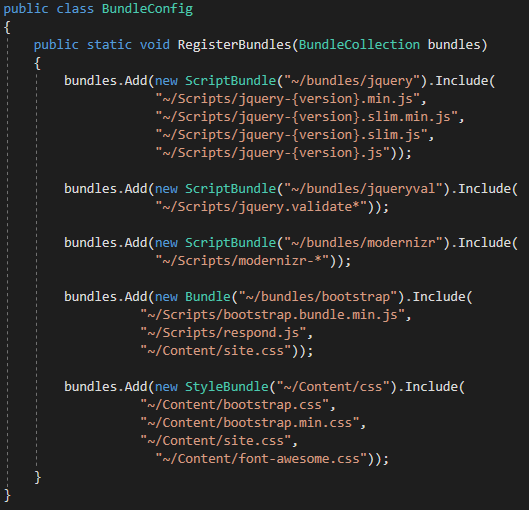


Figura 2.3 – *Bundle-urile* definite în aplicație

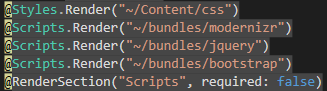


Figura 2.4 – Încărcarea *bundle-urilor* în *View*

**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.

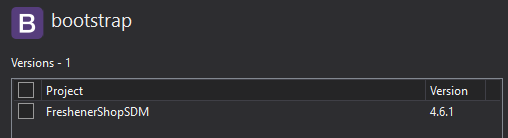


Figura 2.5 – Versiune Bootstrap

**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.

**Capitolul 3**

# Contribuție

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