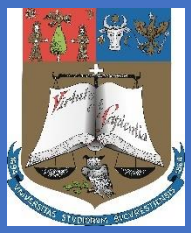


UNIVERISTATEA BUCUREȘTI

FACULTATEA DE

MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de Licență

Aplicație Web

M.C.Recipes

Absolvent:

Miu Cătălin-Stefan

Coordonator Științific:

Andrei Păun

București, România

Septembrie 2021

Rezumat

Ținta acestei aplicații a fost facilitarea introducerii persoanelor neștiutoare în magia care poate fi gătitul. Pentru mulți acest lucru poate părea descurajator, mai ales dacă nu au avut tangențe în viața de zi cu zi.

Am implementat acestă idee printr-o aplicatie web special creată special pentru cei la început de drum dar totodată folositoare și celor mai experimentați. Am reușit să realizez acest lucru prin creearea unei aplicații care conține tot ce mi s-a părut mie necesar în bucătărie. Un dicționar de condimente și ingrediente, unde poți găsi nu doar o descriere a tuturor condimentelor folosite în rețetele de pe site dar și posibili înlocuitori pentru acestea în cazul în care nu le ai la îndemână. Totodată, și paginile cu rețete conțin niște elemente care mi se par absolut necesare, adică niște Tag-uri pentru a te ajuta să îți creezi rapid o imagine a rezultatului final și un timer pentru fiecare pas, pentru a nu mai trebui să părăsești aplicația pentru a mai pune unul.

Concomitent, principalul mod prin care utilizatorul intercaționează cu aplicația este prin front-end, ceea ce îl face foarte important și inciodată nu ar trebui neglijat. Aplicația este gândită în primul rând mobile first, proiectată să fie cu un design responsive [1]iar UI-ul și modul intuitiv de utilizare a aplicației facilitează usurința utilizării aplicației.

[1] Responsive Web Design – este abordarea care sugerează că proiectul și dezvoltarea acestuia trebuie să răspundă comportamentului și mediului utilizatorului pe baza dimensiunii ecranului, platformei și orientării.

Abstract

The goal of this application was to facilitate the introduction of people that don't know into the magic that can be cooking. For many, this may seem daunting, especially if you didn't have any tangents with it in your daily life.

I implemented this ideea through a web application specially for those at the beginning of the road but also useful for those most experienced. I managed to acheve this by creating an application that contains everything I thought was necessary in the kitchen. A dictionary of spices and ingredients, where you can find not only a description of all spices used in the recipes on the site but also possible substitutes for them, if you don't have them at hand. At the same time, the recipe pages also contain some elements that I think are absolutely necessary, Tags to help you quickly create an image of the final result and a timer for each step, so you don't have to leave the application to set one.

Nontheless, the main way in which the user interacts with the application is through the front-end, which makes it makes it very important and should not be neglected. This application is designed primarily mobile first, made to have a responsive design, and the UI and intuitive way of using the application facilitates the ease of use of the application.

Cuprins

1.Introducere

În lumea noastră modernă, mereu în mișcare găsim din ce în ce mai puțin timp pentru a face foarte multe activități, una dintre care este gătitul. Într-adevăr multora nici nu le place iar alții o văd ca o pierdere de timp, mai ales când sute de feluri diferite de mancare, din aproape toate culturile de pe planetă, sunt la câteva atingeri ale ecranului smartphone-ului nostru distanță. Dar mai există și un grup de oameni care vor să își prepare singuri mâncarea, din diferite motive, poate au o alergie sau nu pot consuma anumite alimente din punct de vedere religios, dar totuși nu știu de unde să înceapă, din acest motiv am creeat această aplicație.

A găti este ceva strict uman. Noi, oamenii suntem singurele animale care își gătesc mâncarea înainte de a o mânca. Și este trist că din ce în ce mai puțină lume nu știe sau îi displace să gătească. Aproape 1/3 din tinerii cu vârste între 25 și 34 de ani spun că le displace să gătească. (1) Semipreparatele sau mâncarea pe care o comandăm de la numeroasele restaurante și fast-fooduri conțin cantități imense de sare, zahăr sau grăsime, iar faptul că de obicei porțiile lor sunt mai mari decât cantitatea de mâncare pe care noi trebuie să o consumăm la o masă, nu ajută. De aceea ne putem lua în gretutate fără a ne da seama. (2)

Putem rezolva această problemă gătindu-ne singuri mâncarea acasă, putem avea grijă la ce ingrediente folosim și la calitatea lor, consumăm mai puține calorii, pentru că putem folosi mai puțin unlei sau alte grăsim decât restaurantele. În plus mulți psihologi chiar recomandă să gătim, fiindcă este o metodă să reducem stresul acumulat pe timpul săptămânii. În plus gătitul acasă poate fi un factor benefic în viața socială, putem să chemăm prietenii la cină, fiecare să aducă un fel de mâncare, astfel reducem izolarea socială, de care ne lovim în special anii aceștia, care poate duce la depresie, anxietate sau chiar boli autoimune. (2) Gătitul ne ajută și partea creativă, ¼ din tinerii de pe internet spun că cea mai importantă parte a gătitului este să aduci un element personal în rețetă, să o faci unică, sa o faci a ta. (1)

"A recipe has no soul. You as the cook must bring soul to the recipe"

-Thomas Keller

În aceelași timp aproape 60% din tineri folosesc smartphone-ul sau tableta în timp ce gătesc. (1) De aceea, am încercat să creez o aplicație care să faciliteze gătitul prin urmatoarele funcționalități:

1. Un dex de condimente („Condiment-o-dex”) în care va fi o listă lungă cu toate condimentele și ingredientele mai exotice din rețetele de pe site împreună cu o imagine a lor. În descrierea fiecărui condiment pe lângă atributele sale, miros, gust, va fi și un mic istoric al acestora si o listă de substitute în cazul în care utilizatorul nu îl are la îndemână.
2. Fiecare rețetă va avea o listă de tag-uri (atribute), precum: sănătos, mic dejun, cină, fără gluten, vegan, etc. Pentru a ajuta utilizatorul să își creeze o imagine a rețetei finazliate. Precum să poată sorta și în funcție de acestea.
3. Fiecare pas al rețetei, unde e nevoie, are integrat un timer, pentru a facilita experiența de a găti și a înlătura nevoia de a părăsi aplicația.
4. Lista rețetelor unde se poate vedea o poză cu acestea, numele și o scurtă descriere a acestora.
5. Un design modern și atrăgător la pachet cu un mod de utilizare intuitiv.

2. Prezentarea Aplicației

2.1. Framework și Mediu de Dezvoltare

Aplicația web a fost dezvoltată în Microsoft Visual Studio Code, care este un IDE[2] utilzat în implementarea a numeroase proiecte precum website-uri, aplicații mobile, aplicații web, programe într-o multitudine de limbi de programare. Versiune folosită în creearea programului: Microsoft Visual Studio Code 1.59.

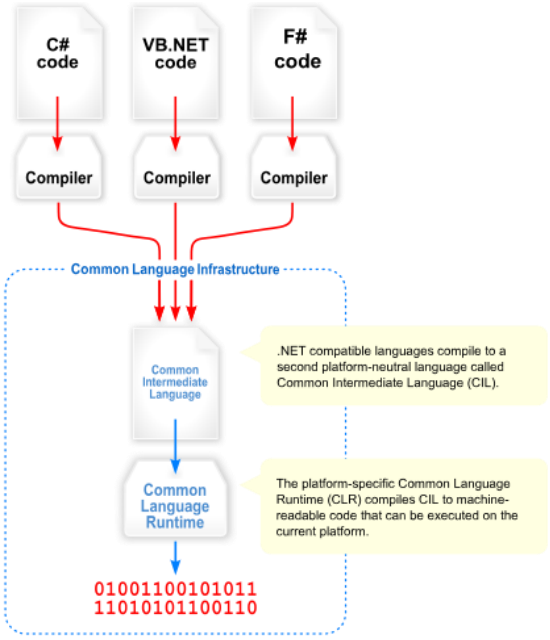
2.1.1. .NET Core

.NET Core este o platformă de dezvoltare gratis și open-source[3], succesor al .NET Framework. Acesta este cross-platform[4] disponibil pe sistemele de operare Windows, Linux și macOS. Limbajele suportate de .NET Core sunt C# și F#. (3)

Aplicațiile care pot fi construite pe .NET Core mobile, aplicații desktop, aplicații web, machine learning ș.a. .NET Core are o arhitectură modulară, folosind pachete NuGet. Aceste pachete oferă numeroase beneficii. Ele pot fi adăugate după nevoile aplicației și pot ușura munca depusă, considerabil. (4)

Procesul de Rulare:

La fel ca și .NET, în .NET Core codul scris este întai compilat în limbajul Common Intermediate Language și este stocat în fișiere assembly cu extensile .dll sau .exe. Când aplicația .NET este rulată atunci CLR-ul folosește JIT (Just In-Time Compiler) pentru a transforma fișierul assembly în cod mașină, care în final poate fi executat de computer. (5)

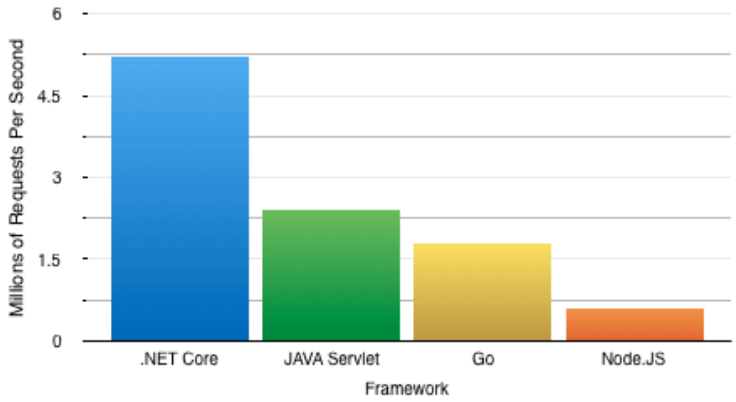


FIGURĂ 1: PROCESUL DE RULARE ÎN .NET (.NET, 2021)

[2] IDE – Mediu de dezvoltare integrat
[3] Open-Source – este un software care are codul sursă disponibil pentru oricine pentru orice scop
[4] Cross-Platform – este un software care funcționează pe mai multe sisteme de operare

ASP.NET Core este un framework moder, ușor care este făcut pentru nevoile programatorilor de astăzi fiind mobile friendly, scalabil, build once run everywhere și foarte performant. Câteva caracteristici ale framework-ului ar fi: (6)

1. Performanța – De obicei performanța depinde de codul programatorului dar nici arhitectura Framework-ului nu este de ignorat, ceea ce face .NET Core este unul dintre cele mai rapide framework-uri disponibile.



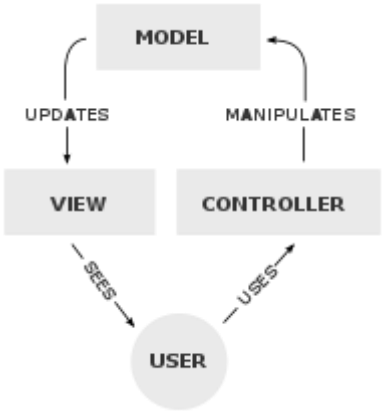
FIGURĂ 2: PERFORMANȚĂ .NET CORE (KHAN,2018)

2. Dependency Injection - .NET Core are Dependency Injection incorporat. Dependency Injection este o tehnică prin care un obiect (sau o metodă statică) furnizează dependențele unui alt obiect. O dependență este un obiect care poate fi folosit(un serviciu) (7) Acum Dependency Injection este principalul mod pentru a creea contexte pentru logging sau baza de date.

3. Output Caching – ASP.NET Core are și Output Caching, ceea ce înseamnă că aplicația salvează cele mai frecvent folosite resurse și fișiere, iar următoarea dată când utilizatorul deschide aplicația primește întâi ceea ce a fost salvat în cache, acest lucru imbunătățește vizibil viteza și experiența utilizatorului, a aplicației.

2.1.2 MVC

Model-View-Controller (cunoscut și sub numele de MVC) este un model de proiectare software folosit de obicei pentru dezvoltarea de UI-uri(interfețe ale utilizatorului). Logica programului este împărțită în trei părți interconectate: modelul de date, informații de prezentare și infomații de control. Acest lucru se face pentru a separa reprezentările interne ale informațiilor de modurile în care informațiile sunt prezentate și acceptate de utilizator. (8)



FIGURĂ 3: MODEL MVC (8)

Inițial folosit în dezvoltarea GUI-urilor (graphical user interface[5]), MVC a devenit foarte popular în dezvoltarea aplicaților web.

Arhitectura modelului:

- **Model** - Aceasta este componenta centrala a aplicației. Această parte include datele din baza de date și se ocupă de manipularea lor datele.

- **View** - Corespunde reprezentării grafice ale aplicației, practic UI-ul aplicației și aceasta interacționează cu utilizatorul final prin prezentarea datelor și lasându-l să le modifice. În proiect această componentă este realizată în React.

- **Controller** – Această componentă administrează cererile transmise de către utilizator aplicației. Controller-ul procesează cererile HTTP făcute de client și transmite înapoi răspunsul.

[5] GUI – Permite utilizatorului să interacționeze cu un dispozitiv electronic prin folosirea simbolurilor, pictogramelor, metaforelor vizuale, dispozitivelor de indicare și indicatoarelor audio

2.1.3. React

React este o librărie JavaScript pentru front-end, bazată pe componente, open source și modernă, creată în special pentru construirea rapidă și interactivă a UI-urilor pentru aplicații web și mobile. Este responsabilă numai pentru componenta View a aplicației. (9)

Câteva caracteristici ale React-ului:

- **JSX** - JavaScript Syntax Extension este o extensie de sintaxă pentru Javascript, este folosită împreună cu React pentru a descriere cum ar trebui să arate UI-ul. Folosinf JSX putem scrie structuri HTML în același fișier care conține cod JavaScript. JSX face mai facil de înțeles și depanat codul pentru că evită utilizarea structutrilor complexe JavaScript DOM.

- **DOM Virtual** – React păstează o reprezentare a DOM[6]-ului ”real” și acesta este cunoscut sub numele de DOM ”virtual” (VDOM). Manipularea VDOM-ului este mai rapid decât DOM-ul real, fiindcă nimic nu este desenat pe ecran, iar când starea unui obiect se schimbă, numai acel obiect este actualizat, față de DOM-ul real care ar fi actualizat toate obiectele.

- **Legarea de date Unidirecțională** – Legarea Unidirecțională a datelor în React păstrează totul modular și rapid. Un flux de date unidirecțional înseamnă că atunci când un dezvoltator proiectează o aplicație în React, deseori îmbrică componentele copil în componentele de tip părinte. În acest mod, dezvoltatorul știe când și unde o eroare se produce, dându-le un mai bun control asupra aplicației.

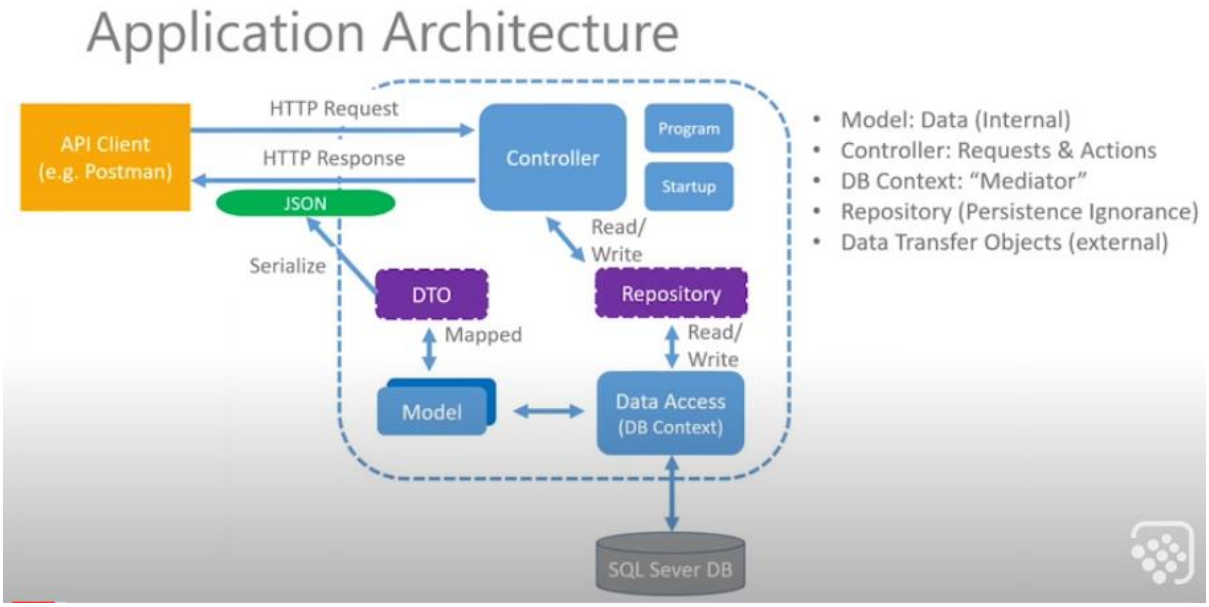
[6] DOM – Document Object Model este reprezentarea datelor care cuprind structura și conșinutul unui document de pe web.

2.2. Back-End

2.2.1. API

Un API (Application Programming Interface) este o conexiune între computere sau între programe. Este un tip de interfață software care oferă servicii unui alt software, față de un User Interface care conectează utilizatorul la computer. În proiect API-ul este folosit pentru a lega baza de date la Front-End (aplicația folosită de utilizator) și pentru a separa și ascunde detaliile interne ale aplicației.

Front-End-ul face request-uri de tip Http precum Get, Post, Put sau Patch și unde e nevoie un mesaj JSON (JavaScript Object Notation). Back-End primește request-ul, iar dacă acesta este formatat corect returnează informația cerută.

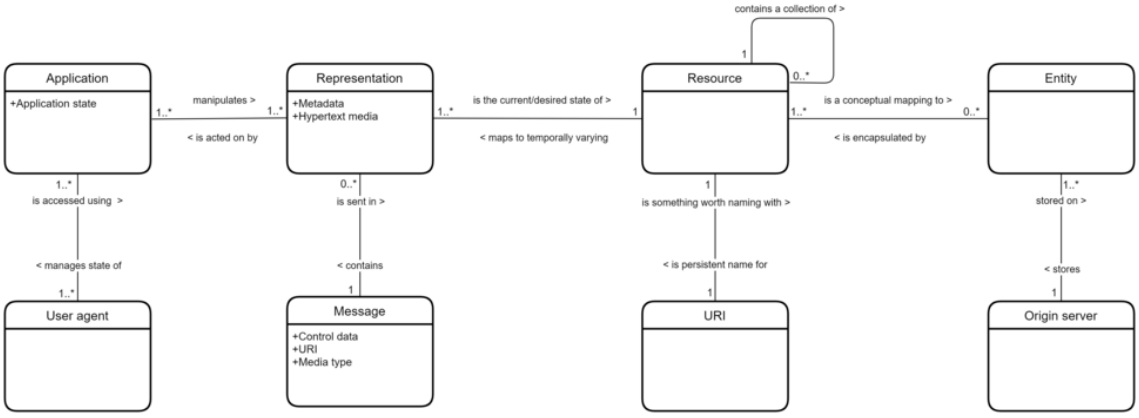


FIGURĂ 4: ARHITECTURA API-ULUI (JACKSON, 2020)

API-ul este de tip REST(Representational State Transfer), el trebuie să fie să aibă interacții scalabile între componente, interfețe uniforme, componentele să aibă o desfășurare independentă și să existe o arhitectură stratificată pentru a facilita stocarea componentelor în cache care reduce latența percepută de utilizator. Ținta REST-ului este să crească performața,

scalabilitatea, spimplicitatea, modificabilitatea, vizibilitatea, portabilitatea și fiabilitatea API-ului. Acest obiectiv este realizată prin urmarea principiilor REST printre care sunt: o arhitectură client-server, folosirea unui sistem stratificat, folosirea unei interfețe uniforme, ș.a. toate principiile trebuie sa fie urmărite pentru a fi declarat API-ul ca REST. (10)

Stilul arhitectural REST este conceput pentru aplicații bazate pe rețea, în special pentru aplicații client-server. Mai mult decât atât, este conceput pentru a fi utilizat la scală largă, pe internet, astfel încât cuplarea între client și server trebuie să fie cât mai ușoară pentru a facilita adoptarea la scară largă. Acest lucru se realizează prin crearea unui strat de abstractizare prin definirea resurselor ce incapsulează entități precum fișiere astfel ascunzând detalile implementării subiacente. Clienții pot solicita resurse numai utilizând un URI[7], iar serverul răspunde cu o reprezentare a resursei. Reprezentarea resursei este alt concept foarte important în REST; o resursă este reprezentată ca hipertext, pentru a fi ușor interpretată de cel mai mare număr de aplicații client. Această decuplare puternică a clientului și a serverului împreună cu transferul de infomații pe bază de text a construit baza pentru îndeplinirea cerințelor WebȘ robustețe, implementare independentă a componentelor, transferul de date large-grain și o ușurință de intrare pentru cititorii de conținut, autori cât și pentru dezvoltatori. (10)



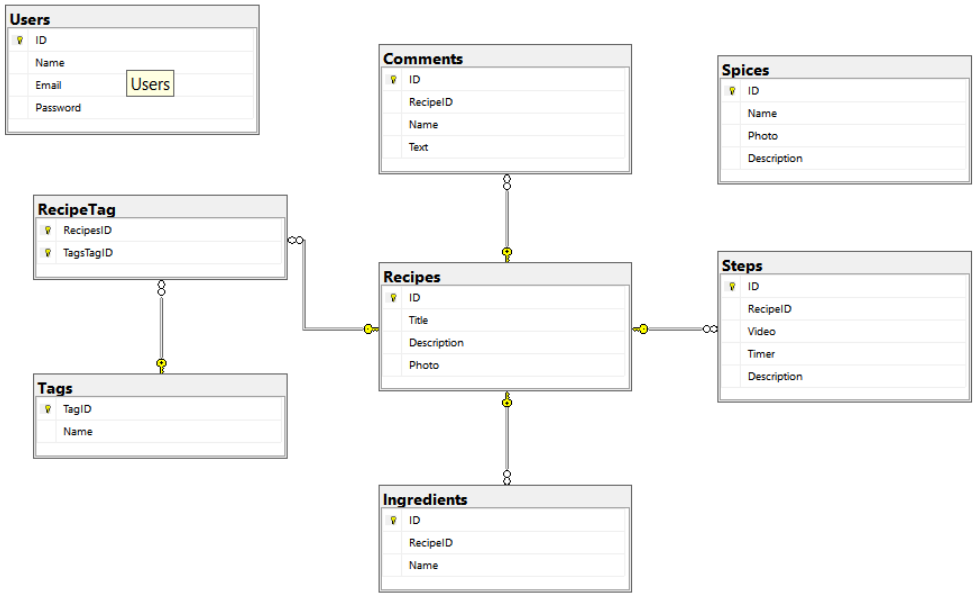
FIGURĂ 5: UN MODEL ENTITATE-RELAȚIE AL CONCEPTELOR EXPRIMATE ÎN STILUL ARHITECTURAL REST. (REPRESENTATIONAL STATE TRANSFER, 2021)

[7] URI – Uniform Resource Identifier este un șir de caractere unic care identifică o resursă logică sau fizică utilizată de către o tehnologie web.

2.2.2. Baza de Date, Entity Framework

Stocarea Datelor a fost realizată cu ajustorul a Microsoft SQL Server, care este un sistem de gestionare a bazelor de date și de Microsoft SQL Server Management Studio 18, care este un mediu integrat pentru gestionarea oricărei infrastructuri SQL.

Relațiile claselor sunt următoarele:



FIGURĂ 6: DATABASE RELATIONSHIPS - EXTRASĂ DIN MICROSOFT SQL SERVER MANAGEMENT STUDIO

Interacțiunea cu baza de date a fost realizată prin Entity Framework Core, care este un O/RM[8] (object-relational mapping) open-source, ușor, extensibil și cross platform utilizat în aplicațiile .NET Core. Acesta are două abordări de dezvoltare:

- Una Code-First, în care EF Core API creează baza de date și tabelele folosind migrări be baza configurărilor în clasele date. Această abordare permite dezvoltatorului să lucreze întâi la structura claselor domeniu primordial, față de workflow-ul tradițional acest mod este mai eficient, crescând viteza de implementare.

[8]O/RM - este o tehnică de porgramare pentru conversia datelor între sisteme de tip incompatibil folosind limbaje de oop. Aceasta practice creează "O bază de date de obiecte virtuale" care poate fi folosită din limbajul de programare.

- Alta Database-First, în care EF Core API ceează domeniul și clasele context în imaginea bazei de date deja existente, dar totuși are suport limitat în EF Core.

Entity Framework ia locul dintre baza de date și clasele domeniu, făcând modificările rezultate în urma cererilor către și dinspre baza de date automat. Un pas cheie în a stabili legătura dintre baza de date și API a fost să adaug în fișierul appsettings.json unde se adaugă un ConnectionString:

```
"ConnectionStrings":
{
  "MC_ApiConnection":"Server=localhost; Initial Catalog= RecipeDB; User ID=*******; Password=********;"
}
```

FIGURĂ 7: CONNECTION-STRING - EXTRAS DIN CODUL APLICAȚIEI

Următorul pas este crearea clasei DbContext, prin aceasta Entity Framework Core recunoaște și poate transpune clasele domeniu în tabelele din baza de date. Utlimul pas este declararea claselor domeniu (entităților) ca elemente care aparțin clasei DbSet<TEntity>.

```
12 references
public class MC_ApiContext : DbContext
{
    0 references
    public MC_ApiContext(DbContextOptions<MC_ApiContext> options) : base(options){ }

    6 references
    public DbSet<Recipe> Recipes { get; set; }

    5 references
    public DbSet<Ingredient> Ingredients { get; set; }

    0 references
    public DbSet<Tag> Tags {get; set;}

    2 references
    public DbSet<Step> Steps { get; set; }

    5 references
    public DbSet<Spice> Spices { get; set; }

    3 references
    public DbSet<Comment> Comments { get; set; }
```

FIGURĂ 8: EF-DBCONTEXT - EXTRAS DIN CODUL APLICAȚIEI

Modul descris mai sus, permite dezvoltaroilor să lucreze la un nivel înalt de abstractizare atunci când se ocupă cu date. Pe lângă asta, facilitează crearea și menținerea aplicaților fără a trebui ca dezvoltatorii să se concentreze pe tabelele și coloanele bazei de date, în aceeași timp scriind mai puțin cod sursă.

În Entity Framework Core crearea și actualizarea bazei de date se face prin migrări. Beneficiul principal în utilizarea migrărilor este accesul la istoricul lor dar și posibilitatea de a reveni la o migrare anterioară în cazul unei erori făcute. Acest lucru poate totuși veni la costul pierderii anumitor date din tabel, fiindcă revenirea la o versiune anterioară modifică tabelul iar uneori anumite date pot fi șterse. (11)

2.2.3. Interfețe

În lumea reală, un contract între doi sau mai mulți oameni îi obligă pe aceștia să îl respecte și să acționeze conform acestuia. În același fel, o interfață include declarările funțiilor, iar entitățile care implementează această interfață trebuie să furnizeze implementarea funțiilor declarate. În C#, interfața este definită folosind cuvântul cheie ”Interface”. (12)

```
5 references
public interface ISpiceRepo
{
    1 reference
    IEnumerable<Spice> GetAllSpices();

    1 reference
    Spice GetSpiceByName(string name);

    3 references
    Spice GetSpiceByID(int id);

    1 reference
    void CreateSpice(Spice spc);

    4 references
    bool SaveChanges();

    2 references
    void UpdateSpice(Spice spc);

    1 reference
    void DeleteSpice(Spice spc);
```

FIGURĂ 9: INTERFAȚĂ SPICES - EXTRAS DIN CODUL APLICAȚIEI

În proiect am folosit interfețele pentru a creea un intermediar între bază de date și controller, mai exact folosind interfețe, ofer un contract controllerului, iar acesta știe ce primește dar totuși nu îl interesează implementarea. Așa că inițial am testat funcțiile http, cu ajutorul unui repository hard-codat, iar mai apoi, folosind dependency injection am putut să trec la a folosi baza de date schimbând o singură linie de cod și știm că o să își păstreze același comportament. Dependency injection face ușoară și schimbarea bazei de date, unde trebuie doar să schimb implementarea Interfeței și schimbarea acestei linii de cod: services.AddScoped<ISpiceRepo[9], SqlSpiceRepo[10]>().

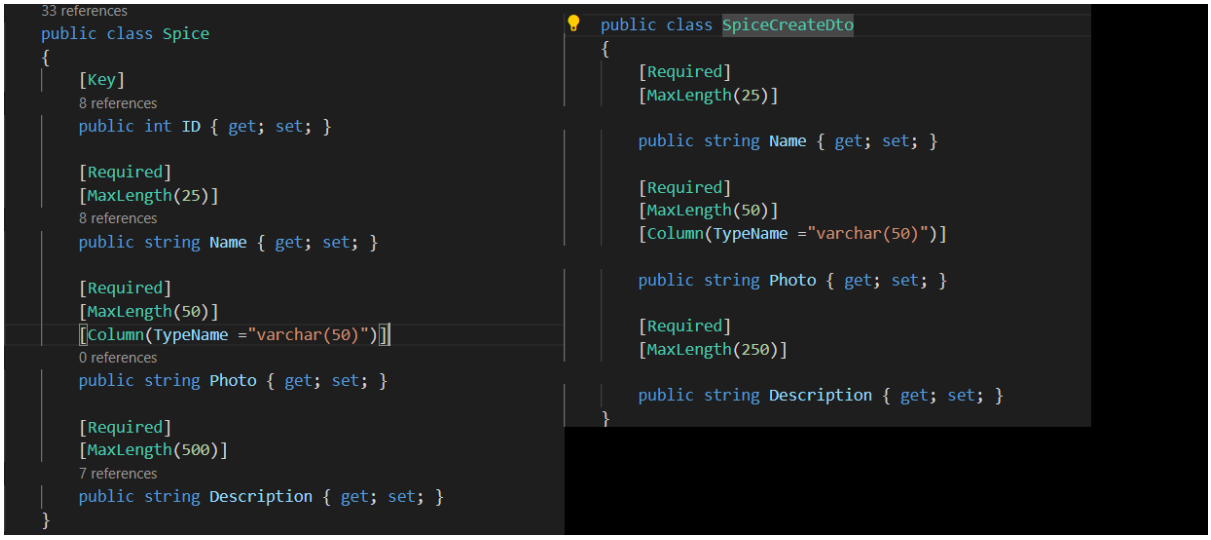
2.2.4. DTOs

Un DTO (data transfer object) este un obiect care transportă date între procese. Dacă nu includem aceste DTO-uri, API-ul expune clientului toate entitățile din baza de date, clientul primește date care se mapează direct la baza de date. Ceea ce este un lucru care uneori nu ni-l dorim. Implementăm DTO-uri dacă dorim să (13):

- Scăpăm de referențele circulare
- Să ascundem anumite proprietăți pe care clientul nu ar trebui să le vadă
- Să nu expunem detalii despre implementarea internă și să reducem riscul de a expune vulnerabilitățile.
- Să decuplezi stratul de servicii de stratul bazei de date.

[9] ISpiceRepo – este Interfața folosită de mine în proiect
[10] SqlSpiceRepo – este implementarea interfeței care comunică cu baza de date

Un loc unde în proiect am folosit DTO-uri a fost la transferul de date al funcției POST, unde nu am inclus atributul ID acesta fiind atribuit automat de către baza de date.



```
33 references
public class Spice
{
    [Key]
    8 references
    public int ID { get; set; }

    [Required]
    [MaxLength(25)]
    8 references
    public string Name { get; set; }

    [Required]
    [MaxLength(50)]
    [Column(TypeName ="varchar(50)")]
    0 references
    public string Photo { get; set; }

    [Required]
    [MaxLength(500)]
    7 references
    public string Description { get; set; }
}
```

```
public class SpiceCreateDto
{
    [Required]
    [MaxLength(25)]

    public string Name { get; set; }

    [Required]
    [MaxLength(50)]
    [Column(TypeName ="varchar(50)")]

    public string Photo { get; set; }

    [Required]
    [MaxLength(250)]

    public string Description { get; set; }
}
```

FIGURĂ 10: MODEL VS DTO - EXTRAS DIN CODUL APLICAȚIEI

2.2.5. Clasele Controller

Web API Controller este similar cu controllerul din ASP.NET MVC adică se ocupă cu solicitările HTTP primite și trimite răspunsul către apelant. El este o clasă care poate fi creată în folderul Controllers sau orice alt folder din rădăcina proiectului, dar numele clasei trebuie să se termine în ”Controller” și trebuie să fie derivat din clasa System.Web.Http.ApiController, iar toate modelele publice se numesc metode acțiune.

Web API-ul decide ce controller Web API și ce metodă să execute pe baza adresei URL și a verbului HTTP(GET/ POST/ PUT/ PATCH/ DELETE). De exemplu metoda Get() va gestiona cererile HTTP GET, iar metoda Delete() va getsiona solicitarea HTTP DELETE. (14)

Metode folosite în proiect:

- **Metoda Get**: Metoda Get este o metodă folosită pentru a cere date de la API. Depinzând de rută această funcție poate întoarce lista cu toate entitățile, sau poate întoarce o anumită entitate în funcție de nume sau id.

```
[HttpGet]
0 references
public ActionResult <IEnumerable<RecipeReadDto>> GetAllRecipes()
{
    var RecipeItems =  _repository.GetAllRecipes();
    return Ok(_mapper.Map<IEnumerable<RecipeReadDto>>(RecipeItems));
}

[HttpGet("{name}", Name = "getRecipeByName")]
1 reference
public ActionResult <Recipe> GetRecipeByName(string name)
{
    var RecipeItem = _repository.GetRecipeByName(name);
    if(RecipeItem == null)
        return NotFound();
    return Ok(_mapper.Map<RecipeReadDto>(RecipeItem));
}
```

FIGURĂ 11: EXEMPLU FUNCȚII GET - EXTRAS DIN COD

- **Metoda Post**: Metoda Post este folosită pentru a creea un obiect. Funcția are și un corp unde se introduc detaliile obiectului, scrise în Json[11], de exemplu: {"Name":"Alin", "Text":"A fost foarte bună rețeta, am să o mai fac"}.

[11] JSON – JSON vine de la JavaScript Object Notations, el este un format ușor pentru a salva și transporta date.

```
[HttpPost]

0 references
public ActionResult <RecipeCreateDto> CreateRecipe(RecipeCreateDto recipeCreateDto)
{
    var recipeModel = _mapper.Map<Recipe>(recipeCreateDto);

    if(recipeModel == null)
        return BadRequest("Null Recipe");

    _repository.CreateRecipe(recipeModel);
    _repository.SaveChanges();

    var recipeReadDto = _mapper.Map<RecipeReadDto>(recipeModel);

    return CreatedAtRoute(nameof(GetRecipeByName), new {Name = recipeReadDto.Title}, recipeReadDto);
}
```

FIGURĂ 12: EXEMPLU FUNCȚIE POST - EXTRAS DIN COD

- **Metoda PUT**: Metoda Put este folosită pentru a actualiza un obiect. În cerere este transmis ID-ul obiectului, pe lângă **toate** detaliile acestuia. El practic înlocuiește complet detaliile obiectului, deci nu pot fi actualizate doar bucăți din el.

```
[HttpPut("{id}")]
0 references
public ActionResult UpdateSpice(int id, SpiceUpdateDto spiceUpdateDto)
{
    var spiceModelFromRepo = _repository.GetSpiceByID(id);

    if(spiceModelFromRepo == null)
    {
        return NotFound();
    }

    _mapper.Map(spiceUpdateDto, spiceModelFromRepo);

    _repository.UpdateSpice(spiceModelFromRepo);

    _repository.SaveChanges();

    return NoContent();
}
```

FIGURĂ 13: EXEMPLU FUNCȚIE PUT - EXTRAS DIN COD

- **Metoda PATCH**: Metoda Patch, față de de metoda Put, poate actualiza doar o parte din detalile unui obiect. Funcția PATCH are un body, mai complex, fiind foarte

versatilă. Un exemplu de corp pentru această funcție ar putea fi: {"op": "replace", "path": "/name", "value": "Piper Sichuan"}

```
[HttpPatch("{id}")]
0 references
public ActionResult PartialSpiceUpdate(int id, JsonPatchDocument<SpiceUpdateDto> patchDoc)
{
    var spiceModelFromRepo = _repository.GetSpiceByID(id);

    if(spiceModelFromRepo == null)
    {
        return NotFound();
    }

    var spiceToPatch = _mapper.Map<SpiceUpdateDto>(spiceModelFromRepo);
    patchDoc.ApplyTo(spiceToPatch, ModelState);
    if(!TryValidateModel(spiceToPatch))
    {
        return ValidationProblem(ModelState);
    }

    _mapper.Map(spiceToPatch, spiceModelFromRepo);

    _repository.UpdateSpice(spiceModelFromRepo);

    _repository.SaveChanges();

    return NoContent();
}
```

FIGURĂ 14: EXEMPLU FUNCȚIE PATCH - EXTRAS DIN COD

- **Metodă DELETE**: Metoda DELETE primește ID-ul obiectului și îl șterge.

```
[HttpDelete("{id}")]

0 references
public ActionResult DeleteRecipe(int id)
{
        var recipeModelFromRepo = _repository.GetRecipeByID(id);

    if(recipeModelFromRepo == null)
    {
        return NotFound();
    }

    _repository.DeleteRecipe(recipeModelFromRepo);
    _repository.SaveChanges();

    return NoContent();
```

FIGURĂ 15: EXEMPLU FUNCȚIE DELETE - EXTRAS DIN COD

- **Metoda SaveFile**: Metoda SaveFile salvează o imagine în folderul Photos.

```
public JsonResult SaveFile()
{
    try
    {

        var httpRequest = Request.Form;

        var postedFile = httpRequest.Files[0];

        string filename = postedFile.FileName;

        var physicalPath= _env.ContentRootPath + "/Photos/" + filename;

        using(var stream = new FileStream(physicalPath, FileMode.Create))
        {
            postedFile.CopyTo(stream);
        }


        return new JsonResult(filename);
    }
    catch(Exception)
    {
        return new JsonResult("AnonSpice.png");
    }
}
```

FIGURĂ 16: FUNCȚIE SALVARE FOTOGRAFIE - EXTRAS DIN COD

- **Metoda AddComment**: Metoda AddComent, îi acordă utilizatorului dreptul să adauge un comentariu la rețeta respectivă.

```
[HttpPost]
[Route("AddComment/{id}")]
0 references
public ActionResult AddComment(int id, Comment comm)
{
    var recipeModelFromRepo = _repository.GetRecipeByID(id);

    if(recipeModelFromRepo == null)
    {
        return NotFound();
    }

    _repository.AddComment(id, comm);

    _repository.SaveChanges();

    return NoContent();

}
```

FIGURĂ 17: FUNCȚIE DE SALVARE COMENTARII - EXTRAS DIN COD

2.3. Front-End

2.3.1 Single Page Application

Un SPA (Single Page Application) este o aplicație web sau un site web care interacționează cu utilizatorul rescriind dinamic pagina curentă, față de metoda implicită a unui browser web care încarcă pagini noi întregi. Scopul unui SPA este de a avea tranziții mai rapide, care fac ca site-ul web să se simtă ca o aplicație nativă.

Într-un SPA, nu se produce niciodata un refresh iar toate codurile HTML, JavaScript și CSS necesare sunt recuperate de browser cu o singură încărcare a paginii, fie resursele sunt încărcate dinamic și adăugate la pagină după cum este necesar. (15)

Sunt câteva probleme cu modelul SPA, principala problemă este cu optimizarea motoarelor de căutare, din cauza lipsei executării JavaScript pe crawlerele celor mai populare motare de căutare. Aceste site-uri cu model SPA erau penalizate și apăreau mai rar la rezultatele căutărilor. Alte probleme mai includ:

- **Istoria browserului** - care nu funționează cum ne-am aștepta datorită faptului că este doar ”o singură pagină” , iar butoanele ”înainte” și ”înapoi” în loc să încarce starea anterioară a SPA-ului acestea acestea încarcă pagina web anterioară

- **Securitatea** – Aplicațiile SPA sunt în continuare predispuse acelorași riscuri de securitate ca paginile web tradiționale, cum ar fi Cross-Site Scripting[12] dar și o serie de vulnerabilități unice cum ar fi expunerea datelor prin logica de pe partea clientului și API-ului și executarea pe partea clientului a securității de pe partea severului.

[12] Cross-Site Scripting – cunoscut și sub numele de XSS este un tip de vulnerabilitate de Securitate întâlnită de obicei în aplicațiile web. Atacurile XSS permit atacatorilor să injecteze scripturi din partea clientrului în paginile web vizualizate de alți utilizatori.

- **Accelerarea înărcării paginii web** – Există câteva metode în accelerarea încărcării inițiale a SPA-ului, cum ar fi încărcarea leneșă modulelor numai când este nevoie de ele, dar este inevitabil faptul că trebuie descărcat framework-ul, și cel puțin o parte din codul aplicației și sigur vom întâlni un API pentru date înainte de a afișa ceva. Acesta este un scenariu ”plătește-mă acum sau plătește-mă mai tarziu”. Iar decizia dintre timpii de așteptare și performață rămâne în mâna dezvoltatorului.

Pentru a conecta Fornt-End-ul la API, trebuie mai întâi să ne asigurăm că avem CORS[13] activat, dacă nu trebuie să introducem în fișierul Startup.cs următoarea bucată de cod:

```
services.AddCors(c =>
{
    c.AddPolicy("AllowOrigin", options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
});
services.AddDbContext<MC_ApiContext>(opt => opt.UseSqlServer
(Configuration.GetConnectionString("MC_ApiConnection")));

services.AddControllers().AddNewtonsoftJson(s => {
    s.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
});
```

FIGURĂ 18: ALLOW CORS - EXTRAS DIN COD

2.3.2 Componentele Front-End

1. **Bara de navigație** - Bara de navigație ajută la traversatul site-ului, are un buton cu logoul site-ului care deservește și ca buton pentru acasă, atunci când lista de pagini este închisă.

FIGURĂ 19: NAVBAR EXTINS - EXTRAS DIN APLICAȚIE

[13] CORS – Cross-Origin Resource Sharing este un mechanism care permite ca resursele restricționate de o pagină web să fie solicitate de un alt domeniu în afara domeniului de la care a fost servită prima resursă

2. **Footer-ul site-ului** – pe langa Copyright conține și butoane cu link-uri către paginile de Facebook, Twitter și YouTube asociate site-ului web.

3. Home Page - Pagina principală a aplicației conține două bucăți de text care explică motivul pentru care am creeat acest site și principul de pe care este construit toată filozofia mea despre gătit. Cele trei div[14]-uri prezintă niște feluri de mâncare care au rol de prezentare.

FIGURĂ 20: HOME PAGE-UL APLICAȚIEI

[14] Div – etichetă utilizată în HTML pentru a face diviza conținutul paginii web

4. **Condiment-o-dex** – Această componentă are în titlu un joc de cuvinte, și conține dicționarul de condimente. Este parte foarte importantă a site-ului. Conține o listă cu toate condimentele din rețete, de la cele mai simple la cele mai exotice. Fiecare condiment are o imagine reprezentativă cu acesta împreună cu numele condimentului și o descriere a acestuia. Componenta mai are și un search bar pentru a cauta rapid condimentul dorit.

FIGURĂ 21: EXEMPLU CONDIMENT - EXTRAS DIN APLICAȚIE

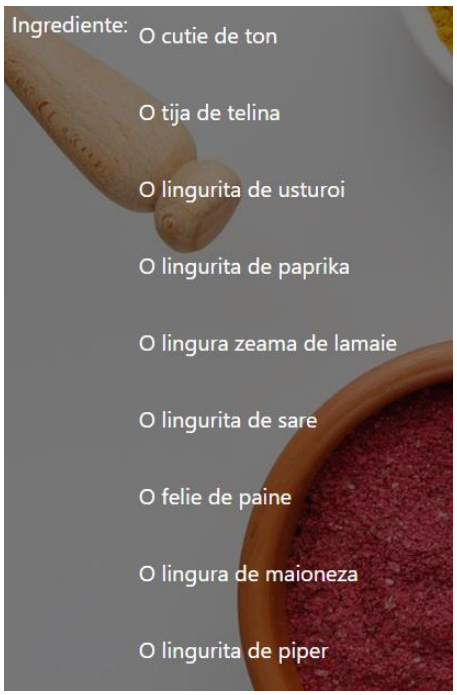
5. **Rețete** – Componența cu rețete are o listă cu toate rețetele și un search bar pentru a cauta rețeta dorită mai ușor. Aici putem vedea titlul fiecărei rețete împreună cu o mică descriere și o poză reprezentativă. Apăsând pe rețetă, ni se deschide o nouă componentă unde putem vedea toată rețeta.

FIGURĂ 22:EXEMPLU REȚETĂ - EXTRAS DIN APLICAȚIE

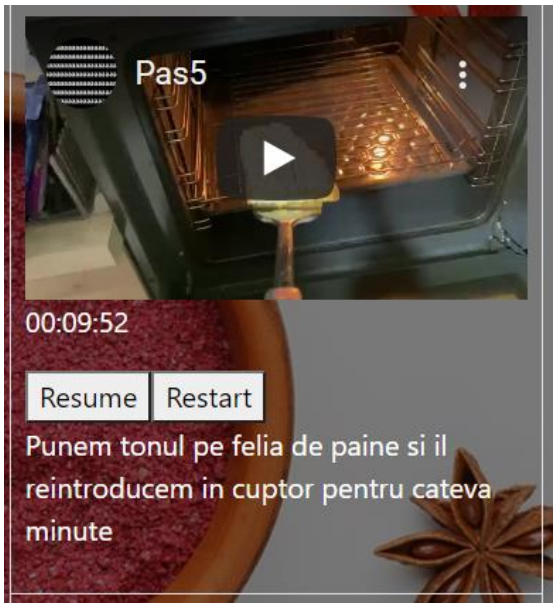
6. **Rețetă –** Aceasta componentă conține toate detaliile rețetei. Primele informații din pagină sunt titlul rețetei, o mică descriere, o poză reprezentativă dar și tagurile rețetei. Aceste tag-uri oferă rapid niște detalii despre rețetă, apăsând pe un tag ne duce pe o ”pagină” unde sunt toate rețetele cu tag-ul respectiv.

FIGURĂ 23: INFORMAȚII REȚETĂ - EXTRAS DIN APLICAȚIE

Mai jos există o listă cu toate ingredientele folosite în rețetă, iar pe partea opusă putem vedea pașii rețetei. Fiecare pas conține o descriere a ce se petrece în pasul respectiv, un timer integrat în aplicație unde este deja setat timpul exact de lăsat ,de exemplu la cuptor ,dar și un mic videoclip, cu mine în timp ce execut acel pas.

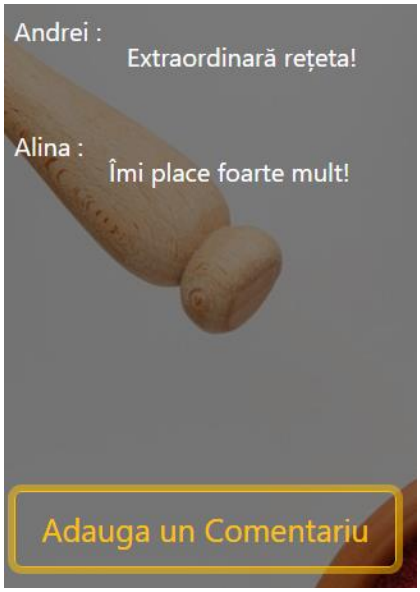


FIGURĂ 24:LISTĂ INGREDIENTE - EXTRAS DIN COD

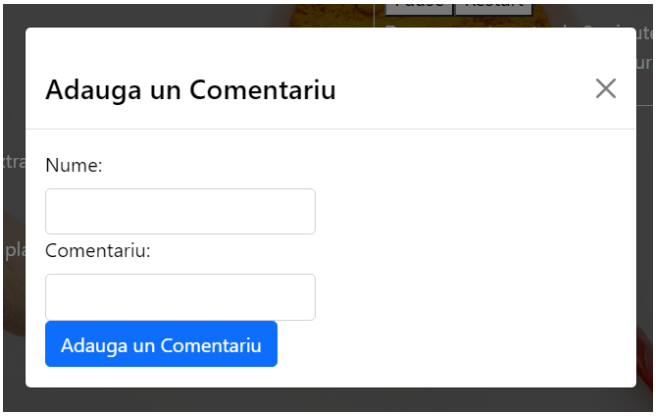


FIGURĂ 25: EXEMPLU PAS AL REȚETEI - EXTRAS DIN COD

În plus, mai este și un tabel de comentarii și o metodă de a adăuga unul. Se apasă pe butonul ”Adaugă un Comentariu” și atunci se deschide un modal care ne oferă două câmpuri, unul pentru nume iar altul pentru comentariu.



FIGURĂ 27: TABEL COMENTARII - EXTRAS DIN APLICAȚIE

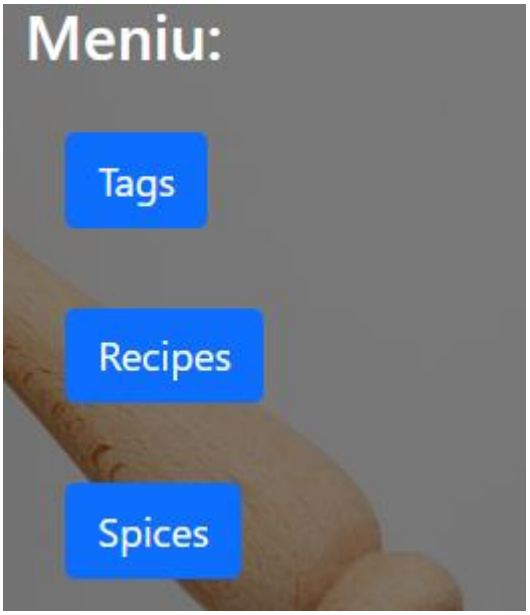


FIGURĂ 26: MODAL ADĂUGAT COMENTARIU

7. **TagSearch** - În rețetă, când utilizatorul apasă pe tag se deschide componenta cu toate rețetele cu proprietatea respectivă, iar când utilizatorul selectează o rețetă se deschide pagina cu ea, o ”pagină” ca cea de mai sus.

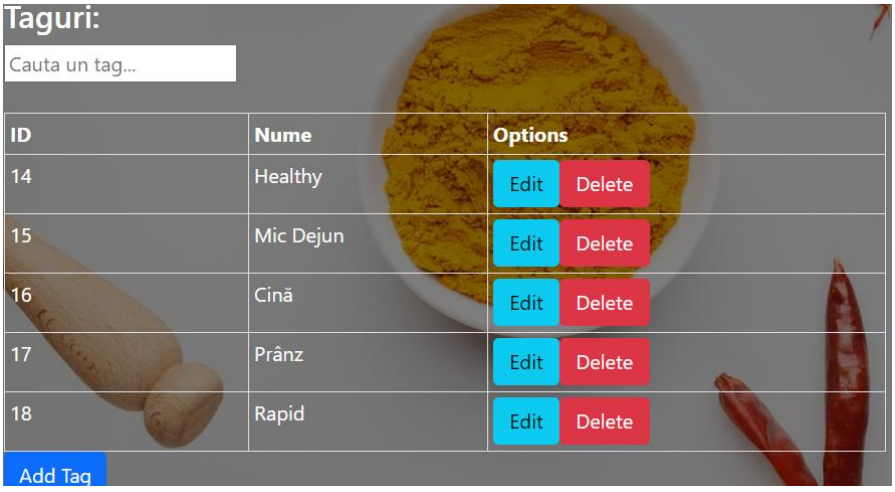
FIGURĂ 28: TAGSEARCH - EXTRAS DIN APLICAȚIE

8. **ControlPanel** – Este o componentă destinată administratorului. Aceasta conține trei butoane, unul pentru Tag-uri, unul prentu Rețete, iar ultimul pentru Condimente. Apăsând pe unul dintre cele trei butoane, ne redirecționează către paginile de administrator ale lor.



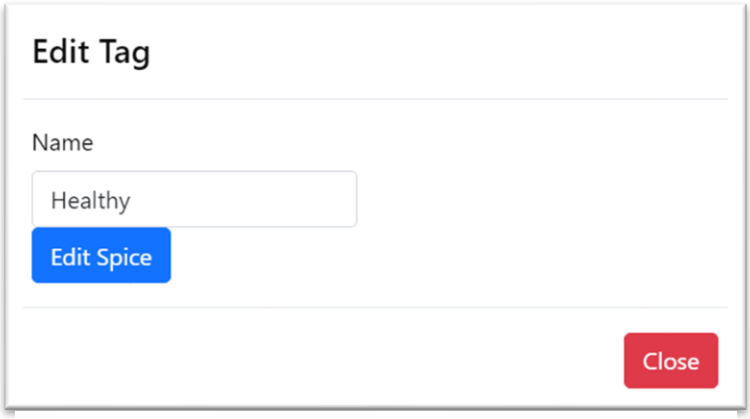
FIGURĂ 29: MENIU ADMINISTRATOR - EXTRAS DIN APLICAȚIE

9. **AdminTag** – Acest component conține un tabel cu toate tagurile, împreună cu două opțiuni, de delete și de edit dar și un buton de adăugat un tag. Există și un Search Bar pentru a găsi rapid tag-ul dorit. Butonul Delete, face exact ce spune, șterge acel tag. Butonul Add Tag, deschide un modal, unde poți adăuga un tag.



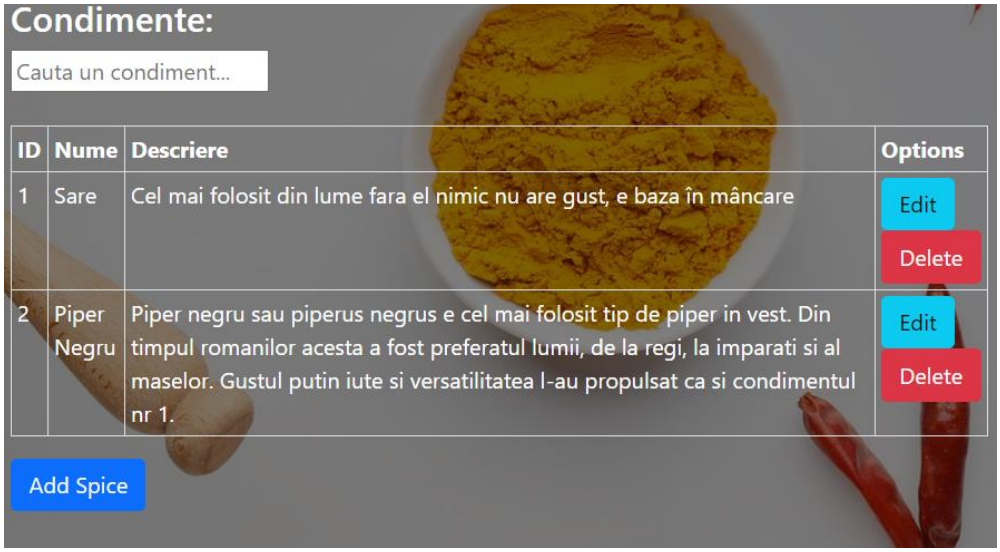
FIGURĂ 30: TABEL TAG-URI - EXTRAS DIN APLICAȚIE

Butonul Edit, deschide un modal care are deja câmpul nume completat, și poți să editezi acel câmp pentru a edita intrarea.



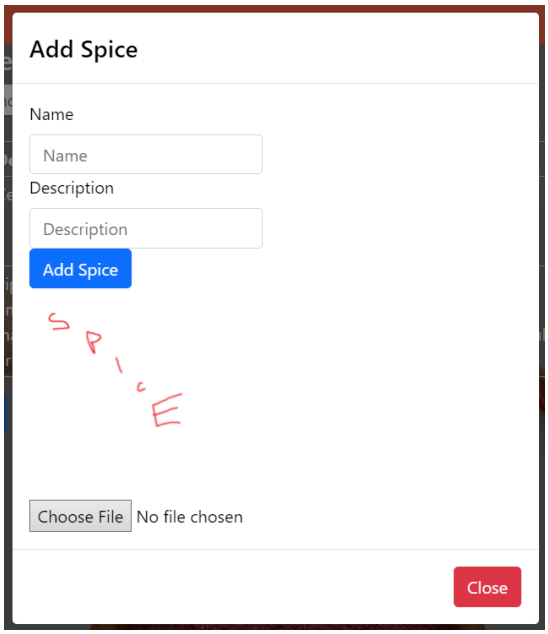
FIGURĂ 31: MODAL EDIT TAG- EXTRAS DIN APLICAȚIE

10. **SpiceAdmin** – La fel ca și TagAdmin, acestă componentă conține un tabel cu toate condimentele, un search bar, butoane de edit și delete pentru fiecare intrare în tabel, dar și un buton de ”Add Spice”.

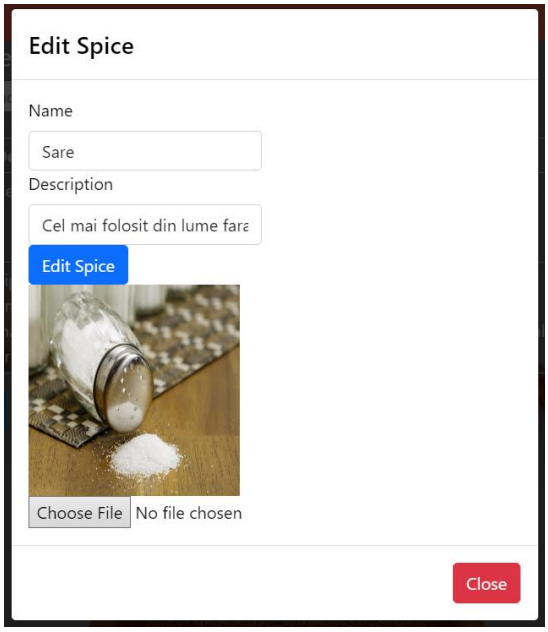


FIGURĂ 32: TABEL CONDIMENTE - EXTRAS DIN APLICAȚIE

Fiind un obiect mai complex, modalele de edit și add, au și o metodă de a încărca o poză în API, în rest acestea funcționează la fel ca cele din AdminTag.

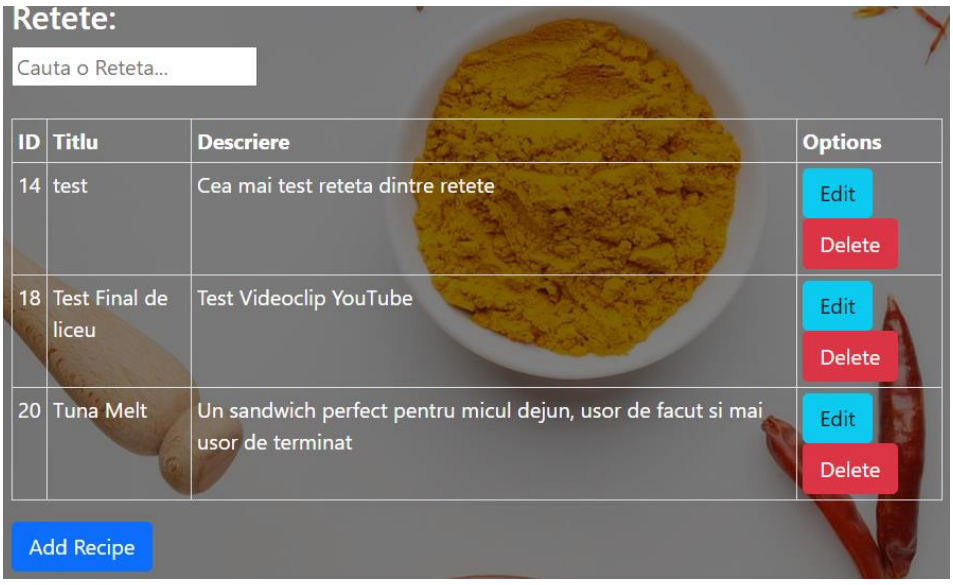


FIGURĂ 34: MODAL ADD SPICE - EXTRAS DIN APLICAȚIE



FIGURĂ 33: MODAL EDIT SPICE -EXTRAS DIN APLICAȚIE

11. **RecipesAdmin** – Acestă componentă conține o listă cu toate rețetele. La fel ca și celelalte pagini Admin, fiecare intrare poate fi editată sau ștearsă. Există și un buton de adăugat rețete, care creează baza la rețetă, doar câmpurile de titlu, descriere și încărcat poza.



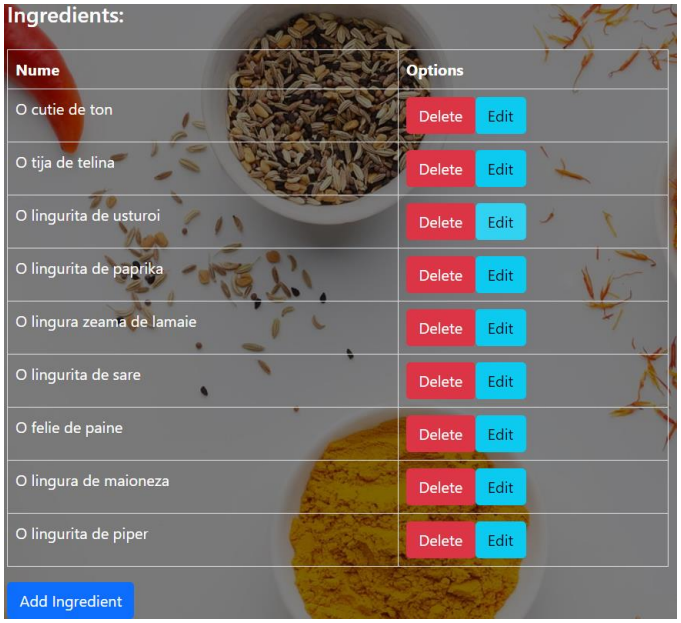
FIGURĂ 35: TABEL REȚETE

Atunci când administratorul apasă pe butonul Edit, atunci este redirecționat către o nouă ”pagină” care conține și mai multe detalii, și unde crearea rețetei poate fi continuată.

12. **AdminRecipe** – Este o componentă în care sunt toate detaliile rețetei. Are mai multe tabele, unul pentru fiecare element al rețetei. Un tabel pentru informațiile de bază, un tabel pentru ingrediente, unul pentru pași și încă unul pentru comentarii.

FIGURĂ 36: TABEL CU INFORMAȚIILE DE BAZĂ -EXTRAS DIN APLICAȚIE

Acest tabel are doar opțiunea edit, unde administratorul poate edita nu doar titlul rețetei cât și imaginea și descrierea. Rețeta se poate șterge din meniul principal RecipesAdmin.



FIGURĂ 37:TABEL INGREDIENTE - EXTRAS DIN COD

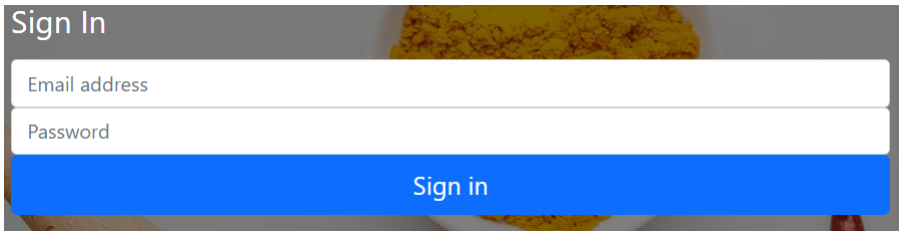
Tabelele de pași și ingrediente pot fi editate la fel ca și cele pentru tag-uri sau condimente. Apăsând pe butonul de edit se deschide pagina modală respectivă unde putem edita datele. Butonul de delete ne șterge intrarea respectivă, iar butonul de add, creează un nou obiect.

FIGURĂ 38: TABEL COMENTARII - EXTRAS DIN COD

Tabelul de comentarii are numai funcția de delete, unde administratorul poate șterge comentariile care sunt nedorite, de exemplu cele care au injurii sau cuvinte jicnitoare.

3. Autentificare

Pentru ca administratorul să poată intra în control panel, acesta trebuie să se logheze, în contul său. Informațiile despre username, parolă și adresă de email, sunt salvate în tabela user a bazei de date. Controller-ul de useri are niște funcții speciale, cele de Log-In, User și Log-Out. Funția de creeare user, ia la intrare un username, o parolă care este hash-uită cu ajutorul pachetului BCrypt. Funcția de Log-in caută user-ul cu email-ul respectiv iar dacă acesta există verifică dacă hash-urile parolelor este aceelași. Pasul următor este să genereze un Jason Web Token cu ajutorul id-ului utilizatorului respectiv, iar acel token este întors ca un cookie spre front end.



FIGURĂ 39: COMPONENTA SIGN-IN – EXTRAS DIN APLICAȚIE

Informațiile din account sunt trimise către front-end numai când funcția este apelată. În această funcție se verifică dacă token-ul este original și corect și este reconvertit în user_ID. Apoi user-ul cu ID-ul respectiv este întors. Funcția Log-out șterge cookie-ul.

```
[HttpPost("Login")]
0 references
public ActionResult Login(UserLoginDto userLogin)
{
    var user = _repository.GetUserByEmail(userLogin.Email);


    if (user == null)
        return BadRequest("Invalid Credentials");

    if(!BCrypt.Net.BCrypt.Verify(userLogin.Password, user.Password))
        return BadRequest("Invalid Credentials");

    var jwt = _jwtService.generate(user.ID);

    Response.Cookies.Append("Jwt", jwt, new CookieOptions{
        HttpOnly= true
    });

    return Ok(new {message = "success"});
```

FIGURĂ 40:FUNCȚIA LOGIN - EXTRAS DIN COD

4. Tehnologii și Librării Utilizate

4.1React-Boostrap

React boostrap înlocuiește Boostrap JavaScript, fiind un framework front-end, open-source proiectat pentru dezvoltarea ușoară și intuitivă a unei pagini web. Fiecare component a fost reconstruit de la zero pentru a fi un component React adevărat, fără a mai fi nevoie de dependențe ca jQuery (16).

Acesta este construit având în vedere compatibilitatea, și este foarte ușor de folosit. În proiect am folosit componentele: Table, Image, Button, Toolbar, Modalm Row, Col și Form pentru a creea un UI modern și ușor de folosit.

4.2React-Router-DOM

React-Router-DOM este utilizat în general în dezvoltarea SPA-urilor. El este folosit pentru a defini mai multe rute în aplicație, iar utilizatorul care introduce o anumită adresa URL în browser și dacă această cale URL se potrivește cu vreo rută, va fi redirecționat către aceasta (17).

În aplicație am folosit componentele:

- Navlink – Este folosit pentru a dăuga atribute de stil pentru rutele active.

- Link – Este folosit pentru a naviga între componentele site-ului.

- BrowserRouter – La baza fiecărei aplicații este o componentă router, BrowserRouter folosește căi URL normale.

- Route – Este cea mai importantă componentă pentru că aceasta are resposabilitatea de a randa UI-ul atunci când calea se potrivește cu adresa URL.

- Switch – Switch randează o anumită rută exclusiv.

```
<Switch>
  <Route path='/' component={Home} exact/>
  <Route path='/recipes' component={Recipes} exact/>
  <Route path='/spices' component={Spices} exact/>
  <Route path='/recipe' component={Recipe} exact/>
  <Route path='/tagsearch' component={TagSearch} exact/>
  <Route path='/adminspice' component={AdminSpice} exact/>
  <Route path='/admintag' component={AdminTag} exact />
  <Route path='/controlpanel' component={ControlPanel} exact />
</Switch>
```

FIGURĂ 41: EXEMPLU COMPONETĂ SWITCH - EXTRAS DIN COD

4.3. AutoMapper

Este o librărie ce automatizează procesul de mapare a unui obiect la altul. Această librărie în afară de faptul că ajută la dezvoltarea mai rapidă a API-ului, și face codul mai citibil și ușor de înțeles.

```
_mapper.Map(tagUpdateDto, tagModelFromRepo);
```

FIGURĂ 42:EXEMPLU FUNCȚIE MAP - EXTRAS DIN COD

4.4 Howler

Este o librărie care se ocupă în principal cu procesarea sunetelor în React, am folosit-o în proiect pentru a suna alarma atunci când expiră timer-ul pasului rețetei.

5. Concluzie și Dezvoltare Ulterioară

5.1. Concluzie

Dezvoltarea aplicației ma ajutat să îmi aprofundez cunoștințele în .NET Core și React. A trebuit să învăț cum funcționează un API și cum se conectează cele trei mari bucăți ale proiectului meu, baza de date, API-ul și Front-End-ul. În același timp am creeat elemente mai complexe ce îi permit administratorului să adauge, să editeze și să șteargă diferite rețete și ingrediente.

În API, a trebuit să înțeleg funcțiile HTTP și a trebuit să le creez. Mi-am imbunătățit cunpștințele de EntityFramework și am învățat cum să creez o bază de date code first. Am fost nevoit să îmi creez și niște funcții speciale, una de salvat poze, pe care le dau din Front-End atunci când modific sau creez vreun obiect.

5.2. Dezvoltare Ulterioară

Aplicația dezvoltată oferă utilizatorului o experiență nouă, a unui site construit cu ultimele tehnologii. Site-ul are o înfățișare îngrijită și curată, și este ușor de navigat, dar și așa consider că sunt destule îmbunătățiri de făcut mai ales pe partea de design. Sunt câteva părți în care mărimea textului sau așezarea acestuia în pagină poate fi ajustat.

Un alt loc în care aplicația mai poate fi îmbunătățită este în introducerea unor câmpuri unde sunt salvate caloriile pe porție ale rețetei. Dar și o metodă de a selecta pentru câte persoane gătești și astfel ingredientele și timpul de efectuare al rețetei să fie ajustate corespunzător.

Totodată, pentru a considera aplicația cu adevărat completă, aș putea adăuga o metodă prin care utilizatorii site-ului, ar putea să își creeze proprile conturi pentru a-și salva rețetele preferate dar și pentru a-și adăuga proprile rețete și a le împărți cu publicul larg.

6. Lista Figurilor

7. Bibliografie

1. **Cooper, Jenny.** Cooking trends among millennials: Welcome to the digital kitchen. *Think with Google.* [Online] Iunie 2015. https://www.thinkwithgoogle.com/future-of-marketing/digital-transformation/cooking-trends-among-millennials/.**[Accesat: August 20, 2021.]**

2. **Magner, Erin.** People Are Cooking Less Than Ever Before. *Well and Good.* [Online] Aprilie 15, 2019. https://www.wellandgood.com/home-cooking-decline/.**[Accesat: August 20, 2021.]**

3. **.NET. *Wikipedia.* [Online] 2021. https://www.wellandgood.com/home-cooking-decline/.[Accesat: August 20, 2021.]**

4. .NET Core Overview. *TutorialsTeacher.* [Online] 2020. https://www.tutorialsteacher.com/core/dotnet-core. [Accesat: August 21, 2021.]

5. Deffo-Fotso, Jean Pierre. How dotnet core compilation work for absolute beginners. *medium.com.* [Online] Iulie 11, 2021. https://jpdeffo.medium.com/how-dotnet-core-compilation-work-for-absolute-beginners-fdba62b3167c. [Accesat: August 21, 2021.]

6. Khan, Shehryar. Top 10 .NET Core Features. *Codinginfinite.* [Online] Decembrie 3, 2018. https://codinginfinite.com/top-10-asp-net-core-features-need-know/.[Accesat: August 21, 2021.]

7. Karia, Bhavya. A quick intro to Dependency Injection. *FreeCodeCamp.* [Online] Octombrie 18, 2018. https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/.[Accesat: August 21, 2021.]

8. Model-View-Controller. *Wikipedia.* [Online] 2021. https://en.wikipedia.org/wiki/Model–view–controller. [Accesat: August 22, 2021.]

9. Sufiyan, Taha. What is ReactJSȘ Introduction To React and Its Features. *Somplilearn.* [Online] Aprilie 9, 2021. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs. [Accesat: August 22, 2021.]

10. Representational State Transfer. *Wikipedia.* [Online] 8 25, 2021. https://en.wikipedia.org/wiki/Representational_state_transfer. [Accesat: August 22, 2021.]

11. Code First Migrations. ***docs.microsoft.com.*** **[Online] Octombrie 23, 2016. https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/#:~:text=Code%20First%20Migrations%20is%20the,works%20with%20your%20EF%20model&text=Keep%20your%20database%20up%20to%20date%20with%20those%20changes. [Accesat: August 23, 2021.]**

12. C#Interface. ***TutorialsTeacher.*** **[Online] 2021. https://www.tutorialsteacher.com/csharp/csharp-interface. [Accesat: August 23, 2021.]**

13. Create Data Transfer Objects. ***Microsoft.*** **[Online] Iunie 6, 2014. https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5. [Accesat: August 25, 2021.]**

14. Web API Controllers. ***TutorialsTeacher.*** **[Online] 2021. https://www.tutorialsteacher.com/webapi/web-api-controller. [Vizitat: August 25, 2021.]**

15. Single-Page Application. ***Wikipedia.*** **[Online] https://en.wikipedia.org/wiki/Single-page_application. [Accesat: August 29, 2021.]**

16. React-Bootstrap. ***React-Boostrap.*** **[Online] https://react-bootstrap.github.io.**

17. React Router. ***Java T Point.*** **[Online] https://www.javatpoint.com/react-router. [Accesat: August 31, 2021.]**

18. Jackson, Les. .NET Core 3.1 MVC REST API. ***YouTube.*** **[Online] Aprilie 22, 2020. https://www.youtube.com/watch?v=fmvcAzHpsk8. [Accesat: August 31, 2021.]**