Recipe Generator

1. *Introducere*

RecipeGenerator este o aplicatie web construita cu .NET si Angular. Scopul nostru principal este de a oferi o experienta de utilizare placuta si facila in gestionarea retetelor culinare.

2. *Prezentare arhitecturala*

Backend (.NET)

Backend-ul aplicatiei este implementat in .NET. folosind ASP.NET Web Api, Entity Framework Core, PostgreSQL, MSTest, Moq si Microsoft Identity.

Proiectul este construit in stil obiect-orientat, respecta o arhitectura pe straturi fiind separat in domeniile: API, Domain, DataAccess, Controller, Application, UnitTests. Fluxul aplicatiei urmeaza modelul de Controller -> Service -> Repository.

Pentru a gestiona dependentele intre componente, este implementat Dependency Injection iar prin intermediul DTO-urilor am asigurat transferul datelor intr-un mod eficient si sigur.

La fiecare strat al aplicatiei sunt implementate operatii asincrone pentru a gestiona eficient sarcini paralele, astfel asigurand o performanta optima.

Interfata Grafica (Angular)

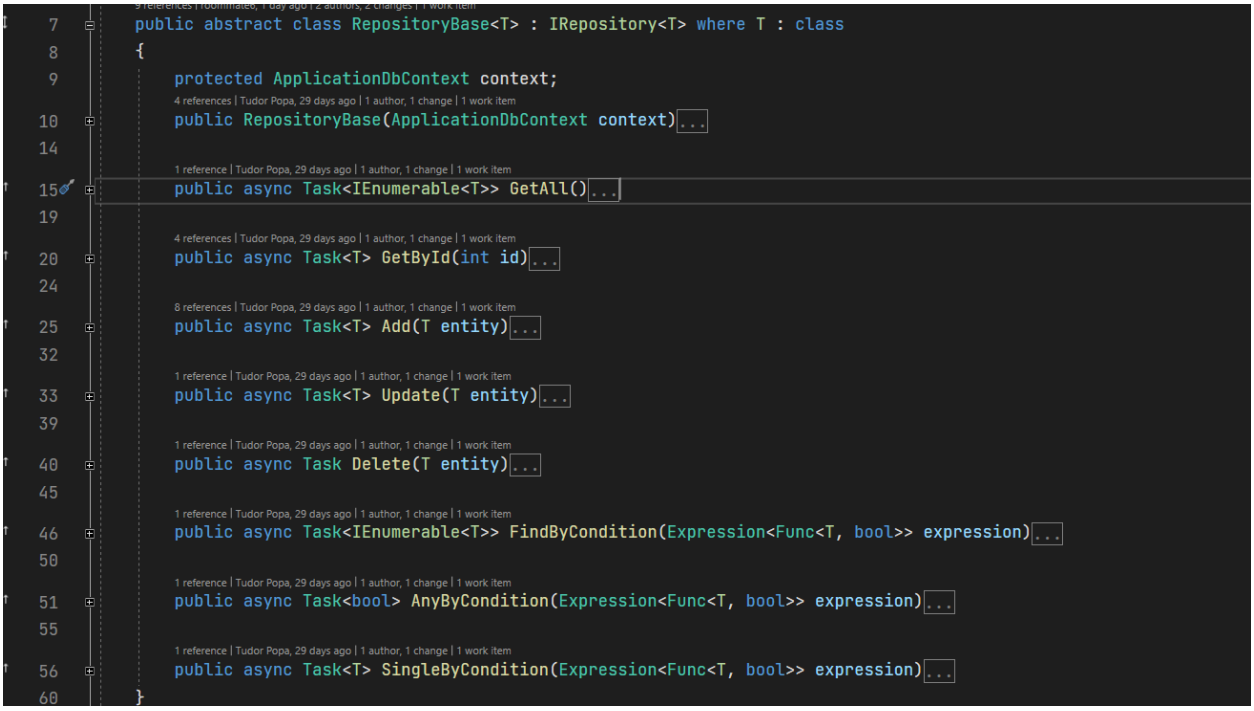
Frontend-ul aplicatiei este implementat in Angular, adera la un design modular, organizandu-se in componente individuale (.html, .scss, .ts, .spec.ts)

3. *Design Patterns*

Template

RepositoryBase<T> este o clasa abstracta templatizata ce ofera un model flexibil si usor de extins pentru alte clase IRepository.

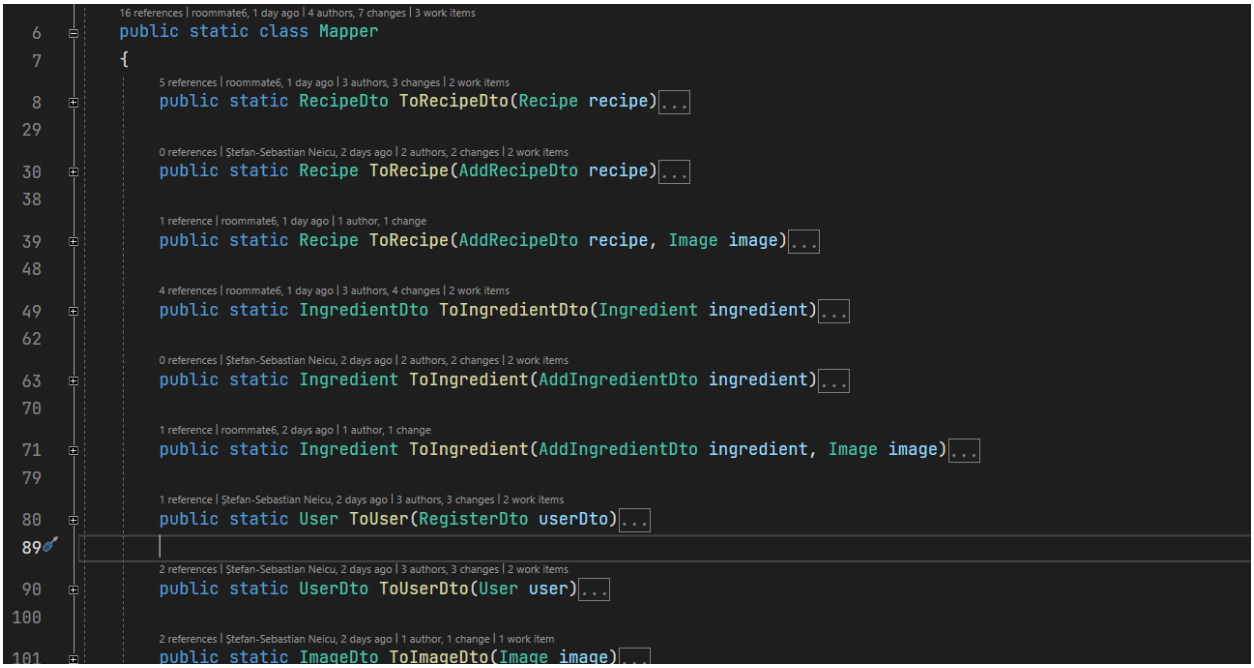
Aceasta implementare ofera metode standardizate pentru a efectua operatiuni CRUD (Create, Read, Update, Delete) si cautari in baza de date.



```
public abstract class RepositoryBase<T> : IRepository<T> where T : class
{
    protected ApplicationDbContext context;
    4 references | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public RepositoryBase(ApplicationDbContext context)...

    1 reference | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<IEnumerable<T>> GetAll()...

    4 references | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<T> GetById(int id)...

    8 references | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<T> Add(T entity)...

    1 reference | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<T> Update(T entity)...

    1 reference | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task Delete(T entity)...

    1 reference | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<IEnumerable<T>> FindByCondition(Expression<Func<T, bool>> expression)...

    1 reference | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<bool> AnyByCondition(Expression<Func<T, bool>> expression)...

    1 reference | Tudor Popa, 29 days ago | 1 author, 1 change | 1 work item
    public async Task<T> SingleByCondition(Expression<Func<T, bool>> expression)...
}
```

Data Transfer Object (DTO)

Mapper este o clasa statica care faciliteaza transferul de date intre modelele (entities) si obiectele de transfer de date (DTOs) in cardul aplicatiei.



```
16 references | roommate6, 1 day ago | 4 authors, 7 changes | 3 work items
public static class Mapper
{
    5 references | roommate6, 1 day ago | 3 authors, 3 changes | 2 work items
    public static RecipeDto ToRecipeDto(Recipe recipe)...

    0 references | Ștefan-Sebastian Neicu, 2 days ago | 2 authors, 2 changes | 2 work items
    public static Recipe ToRecipe(AddRecipeDto recipe)...

    1 reference | roommate6, 1 day ago | 1 author, 1 change
    public static Recipe ToRecipe(AddRecipeDto recipe, Image image)...

    4 references | roommate6, 1 day ago | 3 authors, 4 changes | 2 work items
    public static IngredientDto ToIngredientDto(Ingredient ingredient)...

    0 references | Ștefan-Sebastian Neicu, 2 days ago | 2 authors, 2 changes | 2 work items
    public static Ingredient ToIngredient(AddIngredientDto ingredient)...

    1 reference | roommate6, 2 days ago | 1 author, 1 change
    public static Ingredient ToIngredient(AddIngredientDto ingredient, Image image)...

    1 reference | Ștefan-Sebastian Neicu, 2 days ago | 3 authors, 3 changes | 2 work items
    public static User ToUser(RegisterDto userDto)...

    2 references | Ștefan-Sebastian Neicu, 2 days ago | 3 authors, 3 changes | 2 work items
    public static UserDto ToUserDto(User user)...

    2 references | Ștefan-Sebastian Neicu, 2 days ago | 1 author, 1 change | 1 work item
    public static ImageDto ToImageDto(Image image)...
```

Singleton

AppStateTracker este o clasa TypeScript care administreaza starea legata de utilizatorul autentificat. Prin implementarea design pattern-ului Singleton, este creata si partajata singura instanța a acestei clase pe parcursul aplicatiei.

```typescript
import { User } from '../interfaces/user.interface';
import { EventBusService } from '../services/event-bus.service';

import { AppInjector } from '../app.module';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root',
})

export class AppStateTracker {
  private static _instance: AppStateTracker | null = null;
  private _loggedUser?: User;
  private _eventBusService!: EventBusService;

  private constructor() {
    this._eventBusService = AppInjector.get(EventBusService);
  }

  get loggedUser(): User {
    if (this._loggedUser === undefined) {
      throw new Error('User undefined.');
    }
    return this._loggedUser;
  }
  set loggedUser(user: User) {
    this._loggedUser = user;
  }

  static get instance(): AppStateTracker {
    if (AppStateTracker._instance === null) {
      AppStateTracker._instance = new AppStateTracker();
    }
    return AppStateTracker._instance;
  }
}
```

Dependency Injection

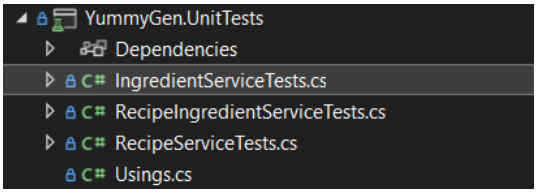
In Back-end clasa Startup este responsabila pentru inregistrarea serviciilor, astfel am putut utiliza clasa WebApplicationBuilder furnizata de AspNetCore pentru injectarea dependintelor.

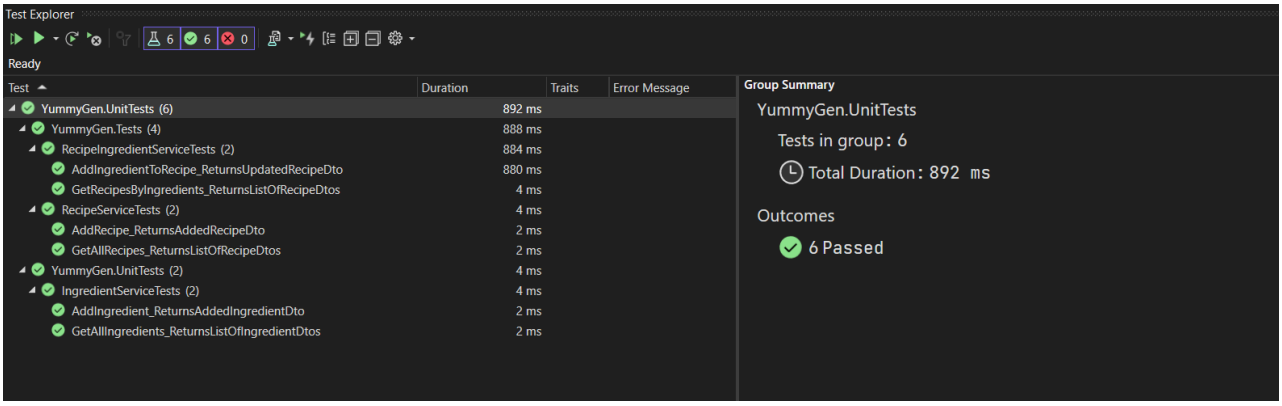
```csharp
public class Startup
{
    1 reference | Ștefan-Sebastian Neicu, 2 days ago | 1 author, 1 change | 1 work item
    public static void RegisterServices(WebApplicationBuilder builder, string originsName, string originsUrl)
    {
        builder.Services.AddCors(options =>...);



        builder.Services.AddControllers();
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen();

        var connectionString = builder.Configuration.GetConnectionString("WebApiDatabase");
        builder.Services.AddDbContext<ApplicationDbContext>(options => options.UseNpgsql(connectionString));

        builder.Services.AddIdentity<User, IdentityRole>(options =>...)
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();

        builder.Services.Configure<FilePathsService>(builder.Configuration.GetSection("RelativePaths"));

        builder.Services.AddSingleton<IFilePathsService>(serviceProvider =>
            serviceProvider.GetRequiredService<IOptions<FilePathsService>>().Value);

        builder.Services.AddScoped<IRecipeIngredientRepository, RecipeIngredientRepository>();
        builder.Services.AddScoped<IRecipeRepository, RecipeRepository>();
        builder.Services.AddScoped<IIngredientRepository, IngredientRepository>();
        builder.Services.AddScoped<IUserRepository, UserRepository>();
```

4. *Unit tests*

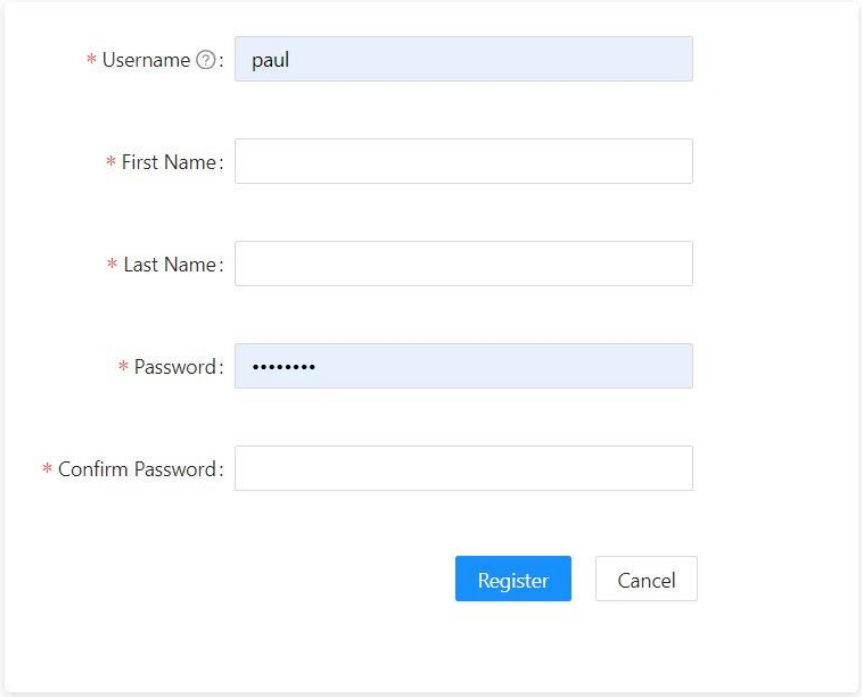
Testele se afla intr-un proiect destinat MsTest.

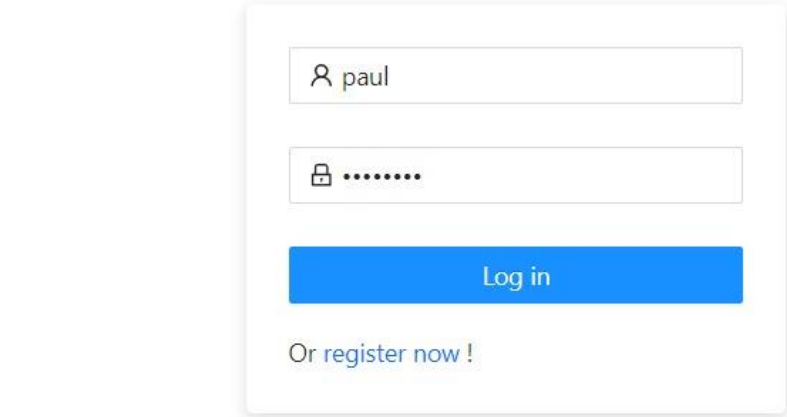
Fiecare clasa service are o clasa de teste ce acopera metodele implementate. Pentru configurarea testelor am folosit mocking cu Moq.

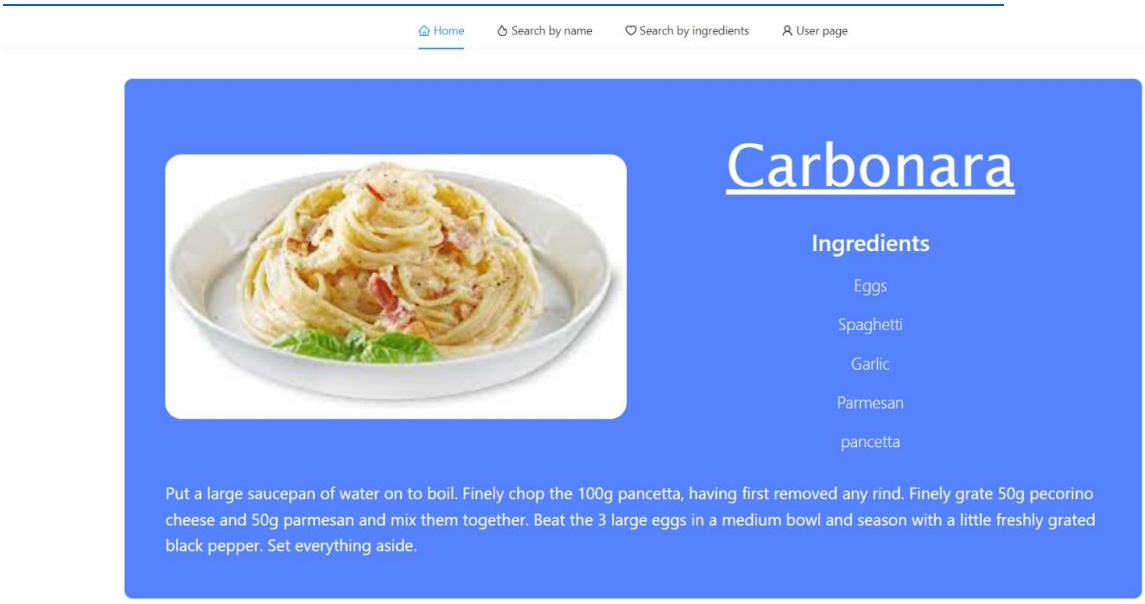


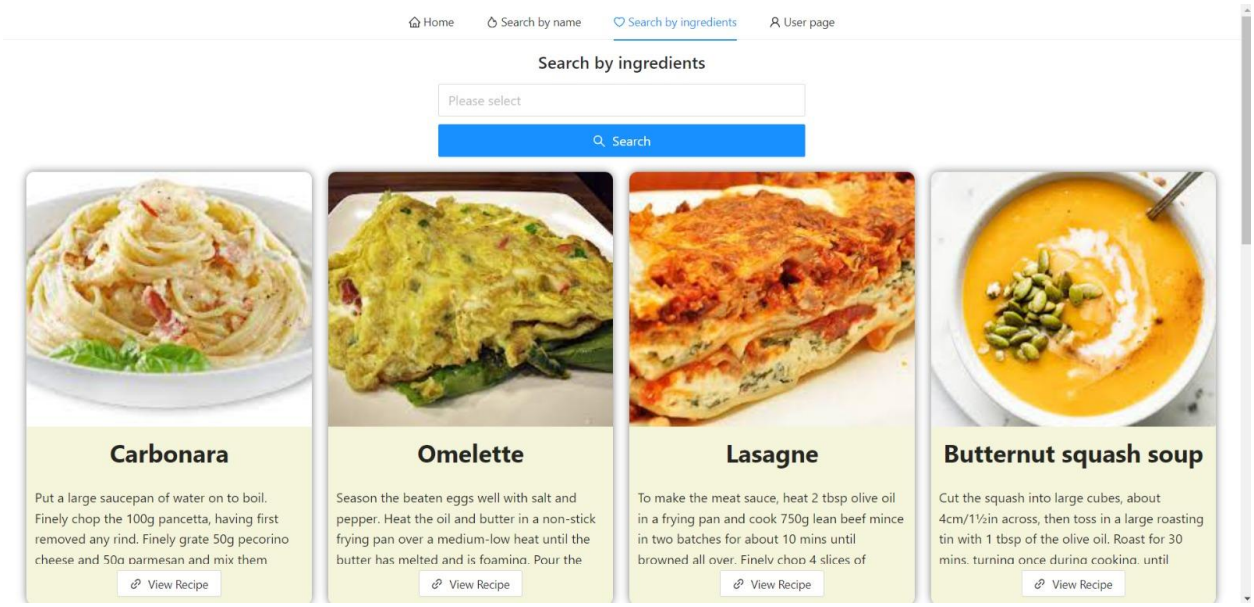


5. *Utilizarea aplicatiei*


* Username :
paul
* First Name :
* Last Name :
* Password :
* Confirm Password :
Register
Cancel


paul
Log in
Or register now !


Home
Search by name
Search by ingredients
User page
Carbonara
Ingredients
Eggs
Spaghetti
Garlic
Parmesan
pancetta
Put a large saucepan of water on to boil. Finely chop the 100g pancetta, having first removed any rind. Finely grate 50g pecorino cheese and 50g parmesan and mix them together. Beat the 3 large eggs in a medium bowl and season with a little freshly grated black pepper. Set everything aside.



6. *Diagrame*

Use Case Diagram (click me)

Clientul are acces sa vada retete, sa se logheze, sa vada detalii despre retete, sa selecteze metoda de cautare a retetelor, sa-si vada profilul.

Class Diagram

Diagrama de calsa explica relatia claselor din back end. Prin ea se inteleg utilizarea design patternurilor precum **template, data transfer objects, mappers(adapters)**.

State Chart Diagram

Aceasta diagrama expune starile in care se poate afla un client in relatia cu aplicatia (in stare de creare a unui cont, in stare de unlogged, logged in, in dashboard, in search bar).

Activity Diagram

Aici se arata felul in care se paseaza controlul de la o entitatea la alta in functie de anumite cazuri si stari in care se afla clientul.

Sequence Diagram

Logica aplicatiei plasata in contextul timpului. Fiecare request asteapta raspunsul din partea celui responsabil cu procesarea lui (back end). Este pus accentul pe relatia dintre back end si front end.

Collaboration Diagram

Prezinta relatiile dintre elemente din aplicatie, fara a se mai pune accent pe timpul efectuarii anumitor sarcini.