**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.

1. *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)**

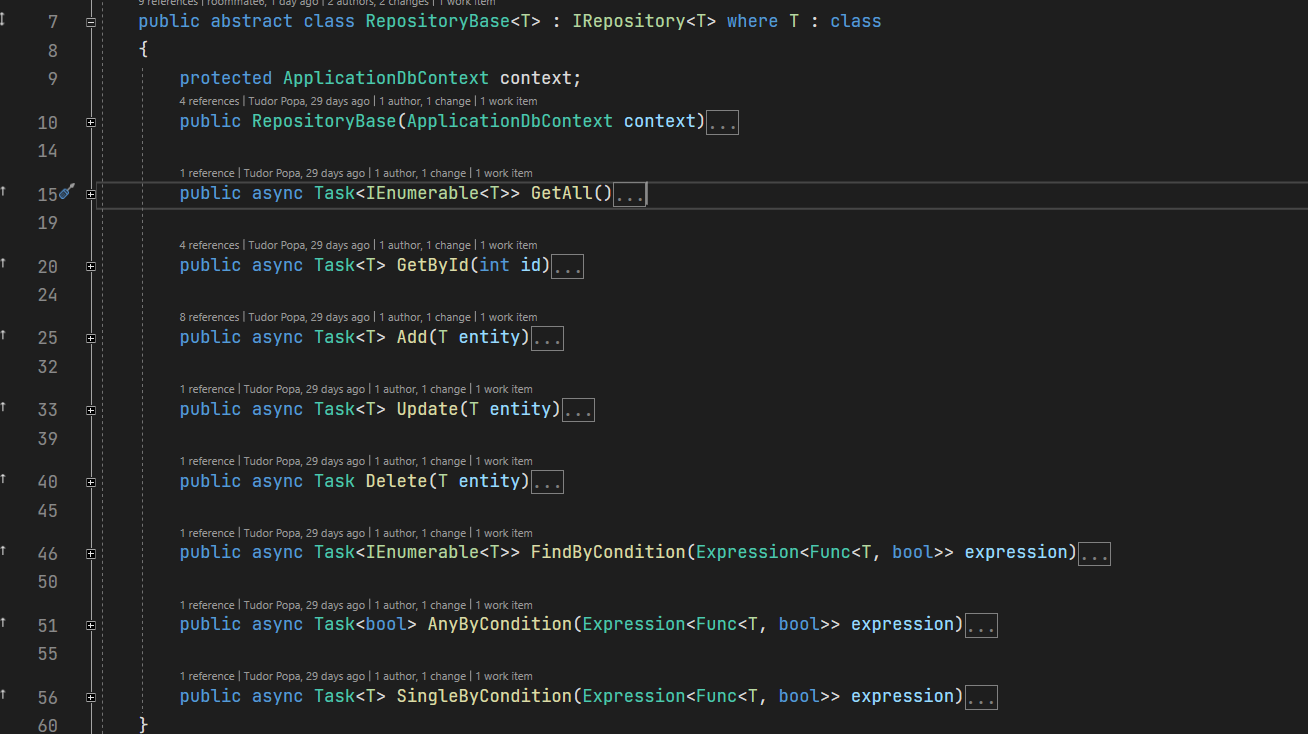
Frontend-ul aplicatiei este implementat in Angular, adera la un design modular, organizandu-se in componente individuale (.html, .scss, .ts, .spec.ts)

1. *Design Patterns*

**Template**

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

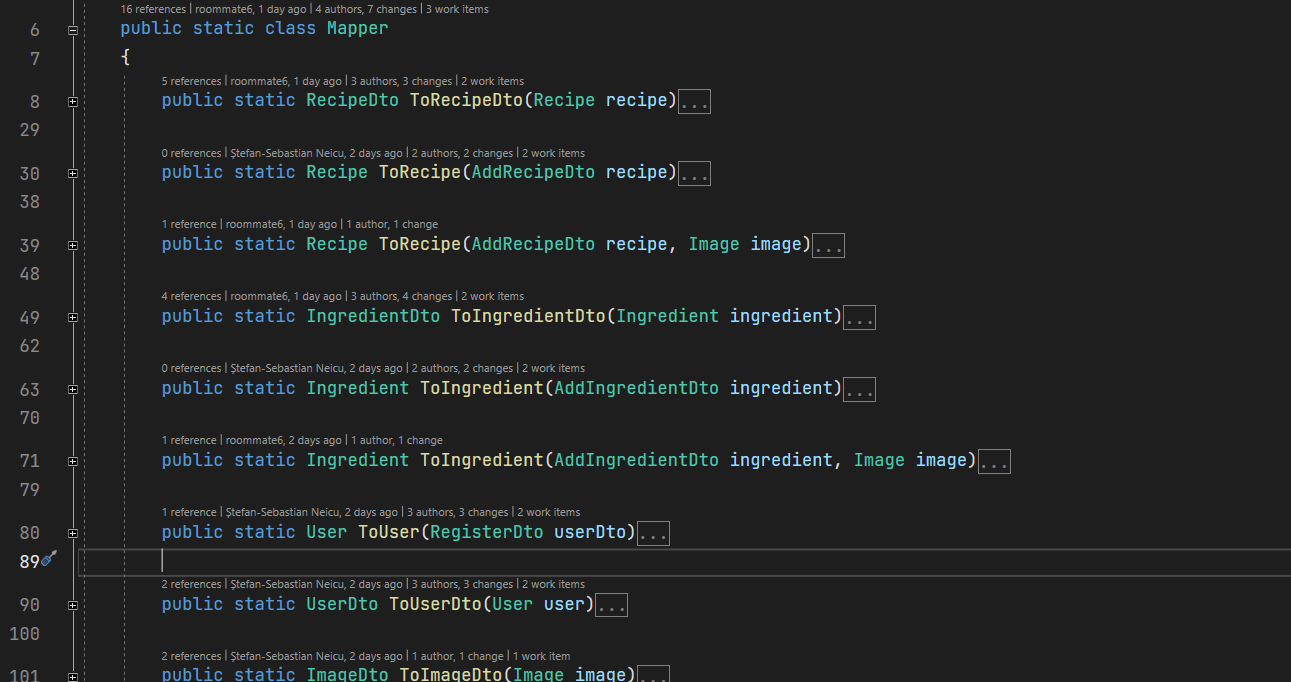
Aceasta implementare ofera metode standardizate pentru a efectua operatiuni CRUD (Create, Read, Update, Delete) si cautari in baza de date.



**Data Transfer Object (DTO)**

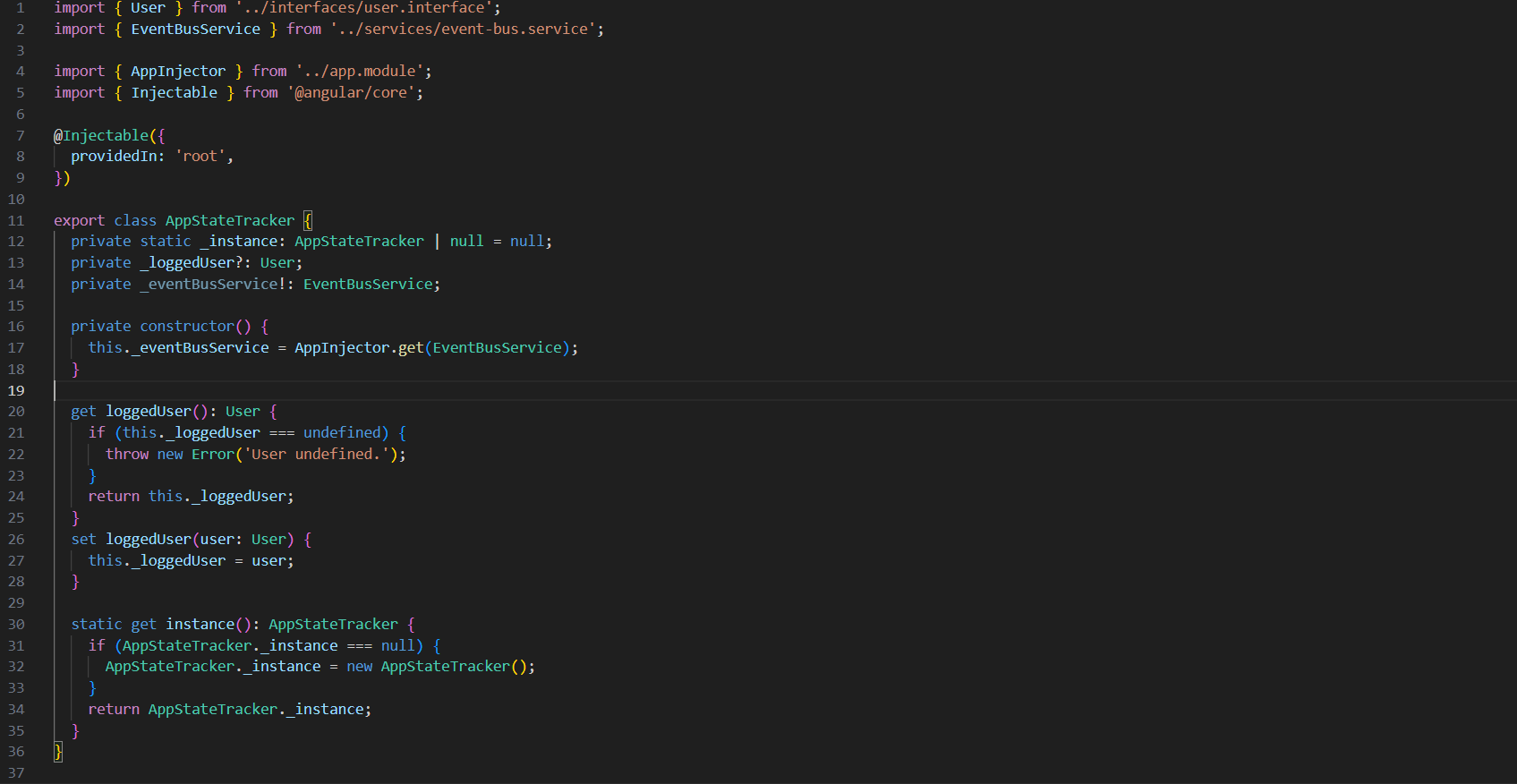
Mapper este o clasa statica care faciliteaza transferul de date intre modelele (entities) si

obiectele de transfer de date (DTOs) in cardul aplicatiei.



**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.



**Dependency Injection**

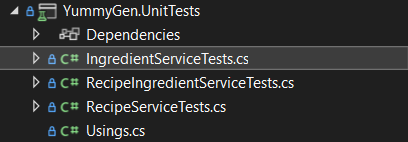
In Back-end clasa Startup este responsabila pentru inregistrarea serviciilor, astfel am putut utiliza clasa WebApplicationBuilder furnizata de AspNetCore pentru injectarea dependintelor.

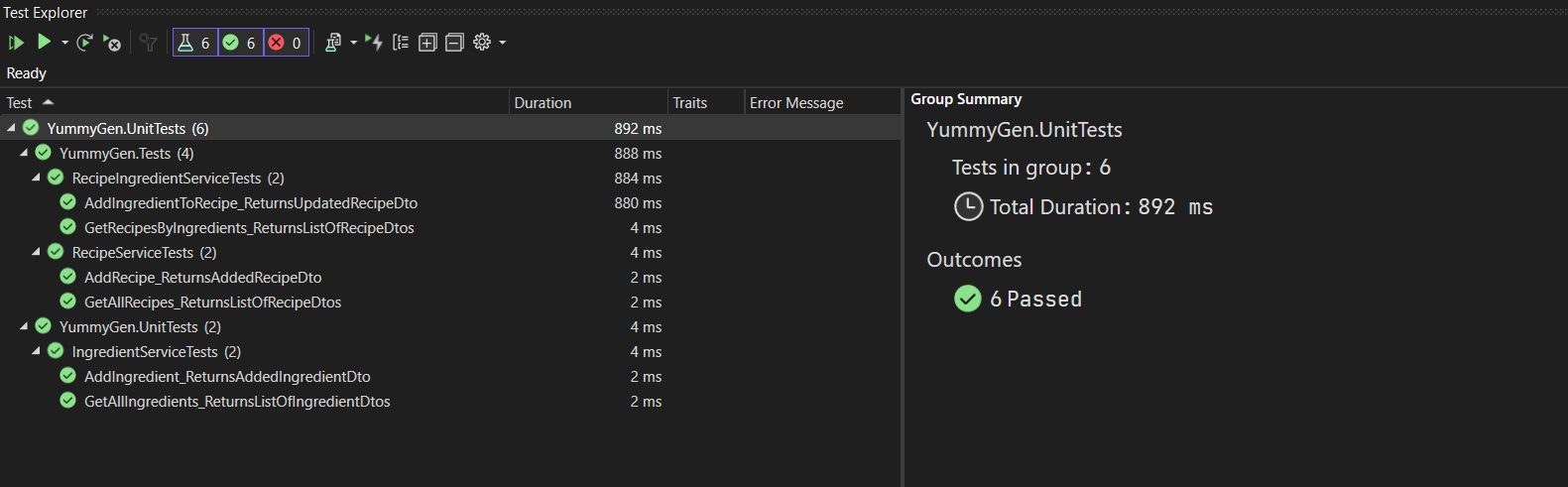


1. *Unit tests*

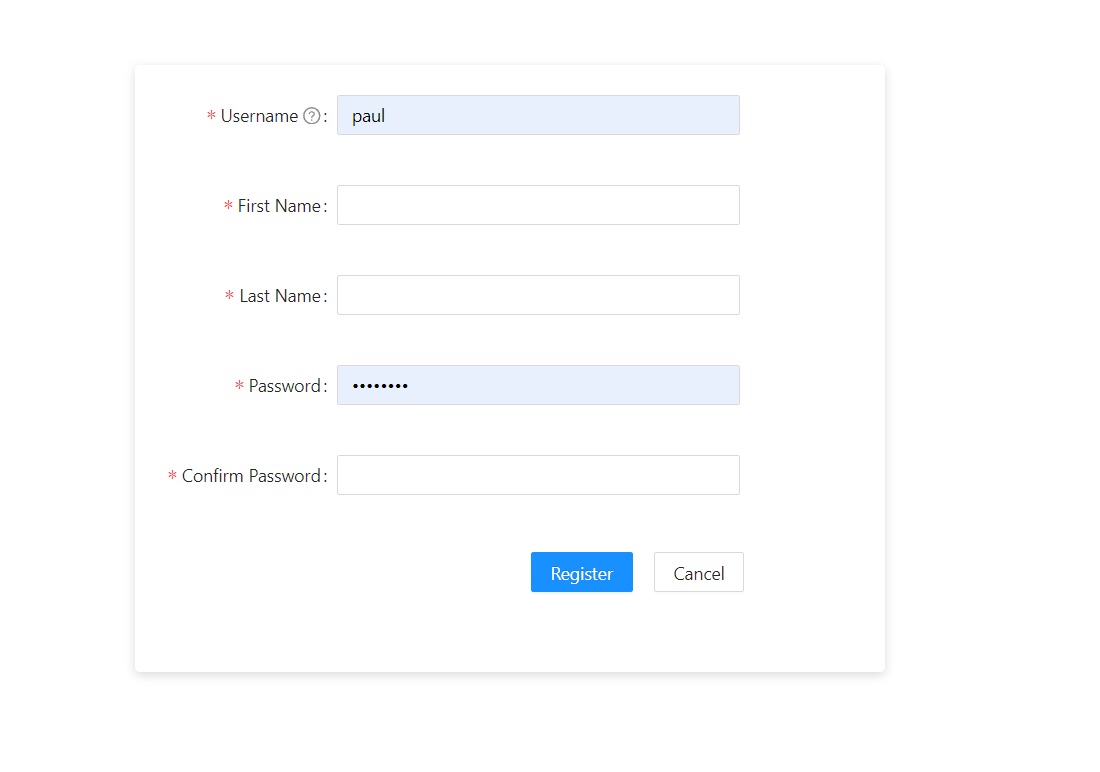
Testele se afla intr-un proiect destinat MsTest.

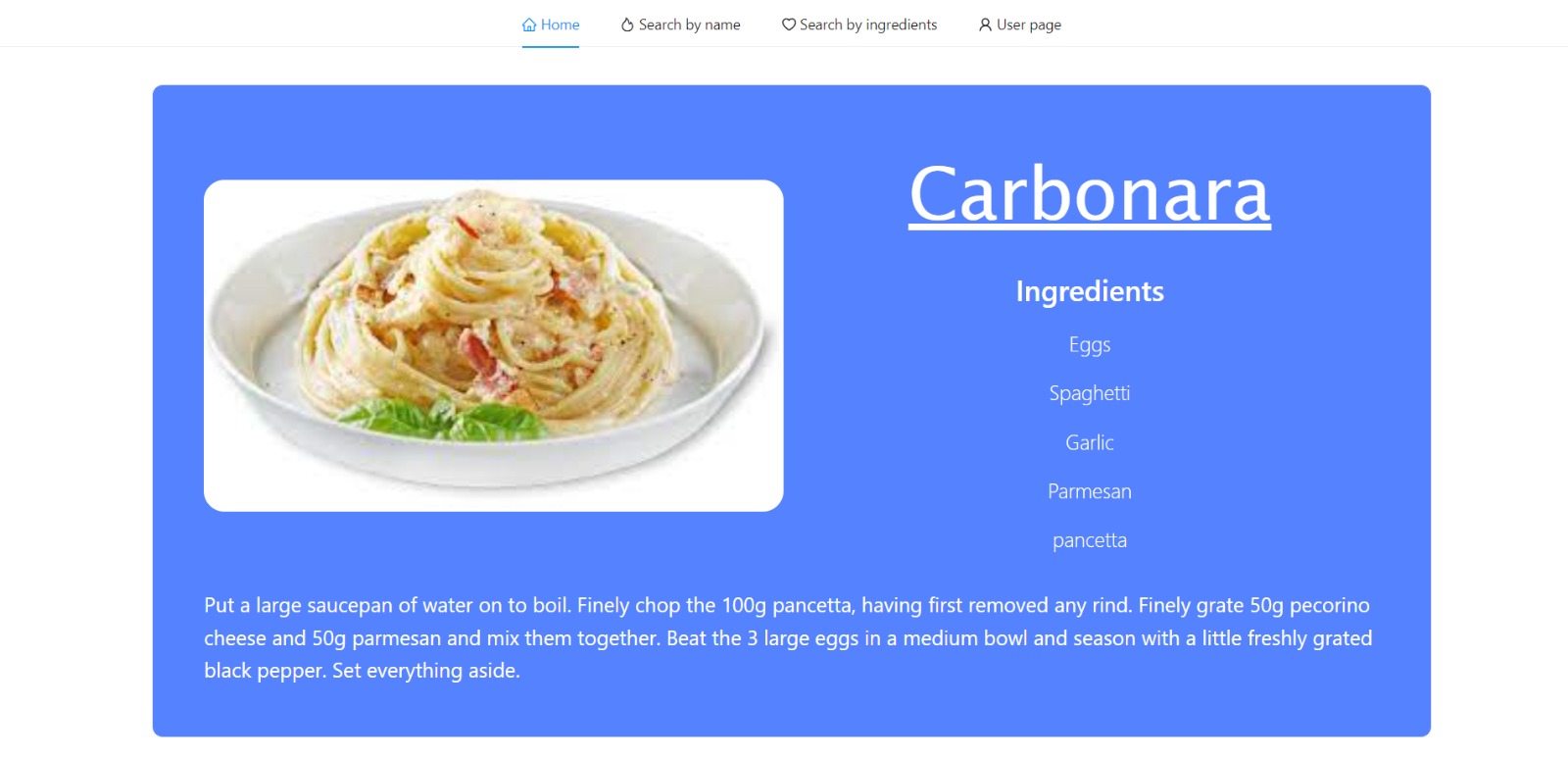
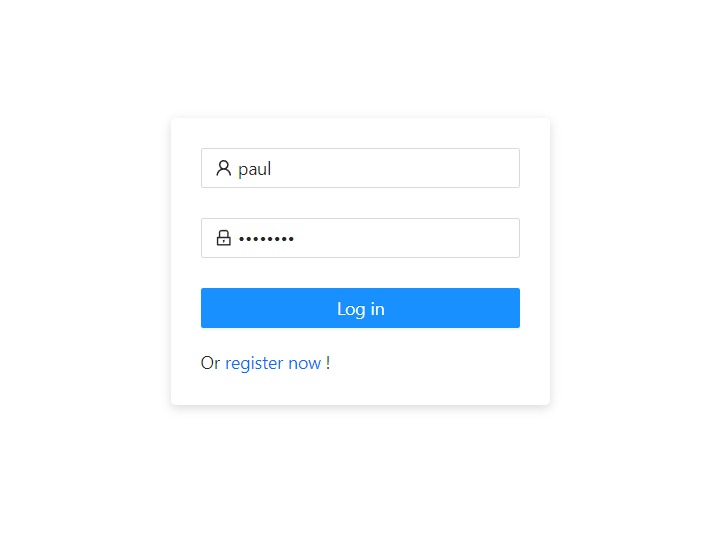
Fiecare clasa service are o clasa de teste ce acopera metodele implementate. Pentru configurarea testelor am folosit mocking cu Moq.

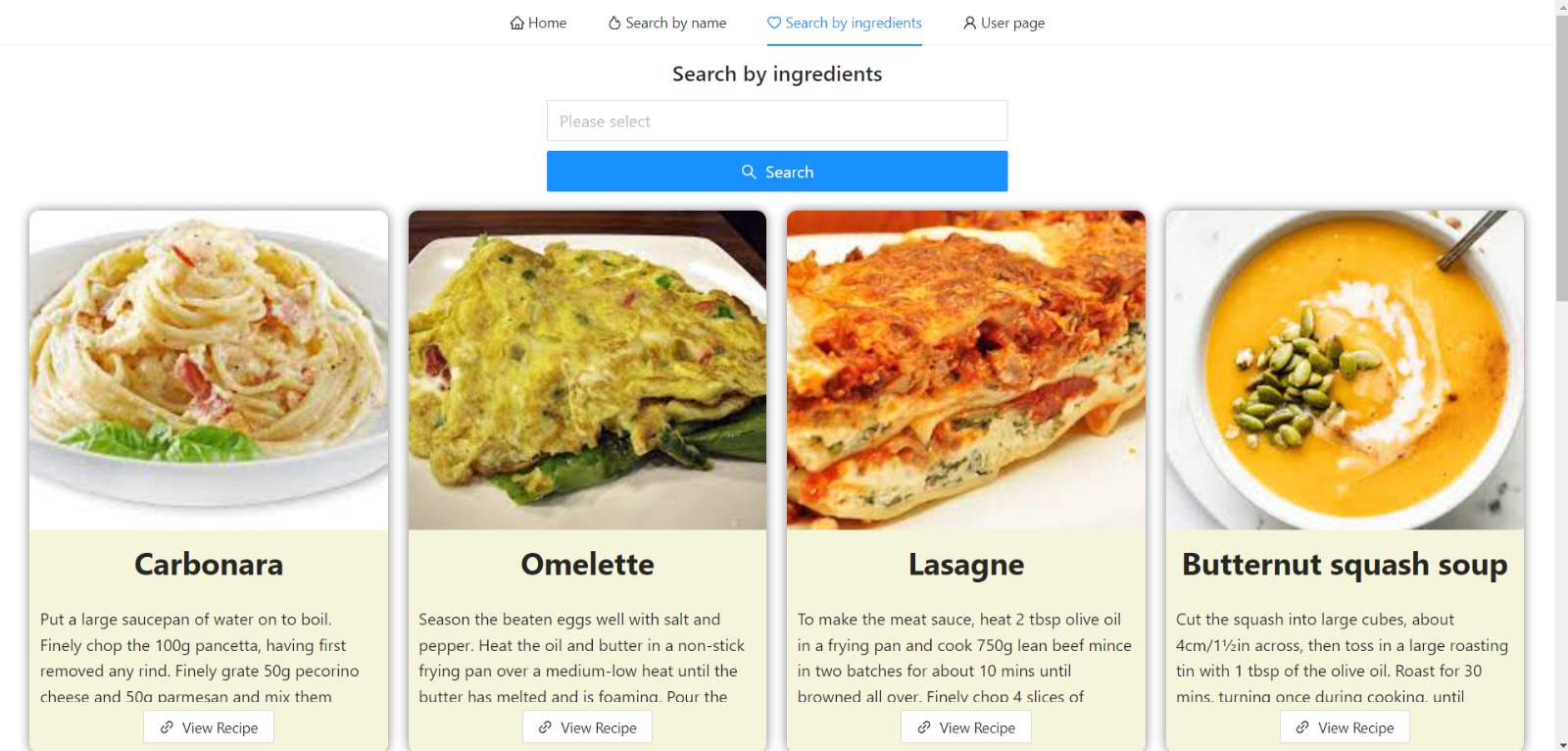




1. *Utilizarea aplicatiei*







1. *Diagrame*

[Use Case Diagram](https://github.com/roommate6/team-pst/blob/main/diagrams/Use%20case%20diagram.png)

[Class Diagram](https://github.com/roommate6/team-pst/blob/main/diagrams/Class%20diagram.png)

[State Chart Diagram](https://github.com/roommate6/team-pst/blob/main/diagrams/State%20Machine%20Diagram.png)

[Activity Diagram](https://github.com/roommate6/team-pst/blob/main/diagrams/Activity%20Diagram.png)

[Sequence Diagram](https://github.com/roommate6/team-pst/blob/main/diagrams/Sequence%20Diagram.png)

[Collaboration Diagram](https://github.com/roommate6/team-pst/blob/main/diagrams/Communication%20Diagram.png)