Raport final KunFood

Membrii: Turcu Nicușor, Trifan Tamara, Rusu Cristian, Puiu Dănuț

Introducere

Scopul aplicației este de a ajuta utilizatorul la gătitul/achiziționarea mâncării. Aplicația pune la dispoziție o gamă largă de clasificatori pentru diferite rețete. Rețetele sunt stocate într-o bază de date, preluate din surse diferite de site-uri culinare.

Ce s-a propus vs. ce s-a implementat

Pentru început, ne-am propus să implementăm această aplicație având 3 componente principale: Food, Forum și User (autentificarea).

În partea de Food intră tot ce ține de managementul rețetelor și ingredientelor. Partea de backend a fost în totalitate implementată, inclusiv partea de sortare și filtrare a rețetelor. În plus, a fost creat un script prin care au fost importate informații despre rețete din mai multe site-uri, aceste rețete au fost apoi adăugate în baza de date pentru a fi prelucrate de către aplicație. Pe partea de frontend s-a putut integra doar afișarea rețetelor și accesarea fiecăreia pentru detalii.

Pe partea de Forum s-a implementat în totalitate backend-ul, însă în frontend s-a reușit numai integrarea astfel încât să se poată face adăugarea categoriilor de către administrator, iar userii obișnuiți să poată să adauge thread-uri noi.

Partea de User a fost implementată și integrată complet. Când un user se conectează, acestuia îi este atribuit un token cu ajutorul JWT. Parola este transmisă după ce îi este aplicată o funcție de hash.

```
...
using System.Security.Cryptography;
...
[HttpPost]
 public async Task<IActionResult> Login(LoginDTO dto)
 {
     User user = await _repository.GetByEmail(dto.Email);

     // Check if the user exists
     if (user != null)
     {
         // Hash the model password
         byte[] bytes = Encoding.UTF8.GetBytes(dto.Password);
         SHA256Managed cipher = new SHA256Managed();
         byte[] hash = cipher.ComputeHash(bytes);
         string hashStr = "";
         foreach (byte b in hash)
             hashStr += string.Format("{0:x2}", b);

         // Check if the passwords match
         if (user.Password.Equals(hashStr))
         {
             // Generate the token
             var token = JwtToken.GenerateToken(user);

             // Write the token to the cookie
             Response.Cookies.Append("SessionId", new
               JwtSecurityTokenHandler().WriteToken(token));

             HttpContext.Session.SetString("user_id", user.Id.ToString());
             // Redirect to homepage
             return RedirectToAction("Index", "Home");
         }
         ModelState.AddModelError("", "Invalid password!");
     }
     else
         ModelState.AddModelError("", "Account is not registered!");

     // If something went bad, return the model back to view
     return View(dto);
 }

...
```

Pattern-uri și best practice-uri folosite

În implementarea aplicației s-a folosit design pattern-ul MVC, ce a ajutat la modularizarea aplicației, separând interfața de logică. În plus, a fost folosit generic repository pattern.

Pentru lucrul cu bazele de date din cod s-a folosit ORM-ul Entity Framwork Core, ce a permis o abordare mult mai facilă a transpunerii structurii și relațiilor dintre obiectele folosite de model în structura bazei de date. Pentru crearea și eventualele upgrade-uri ale bazei de date s-au folosit migrații. A fost definit un filtru implicit pentru controllere pentru a fi evitată duplicarea codului.

Principiile SCRUM au fost aplicate în acest proiect prin întâlniri cu echipa aproape săptămânal, cod review-uri, pair programming, learning by doing și brainstorming.

Au fost create teste pentru repository, toate sunt rulate cu succes, iar code coverage este de 73%.