Universitatea Tehnica a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator Nr. 3
la Tehnologii Web
Tema: Modele de proiectare. BusinessLogic

A efectuat — St. gr. TI-216
Rosca Dorin

A verificat — Asist. univ.
Gaidarji Alina

Chișinău 2023

Sarcina lucrării de laborator: Familiarizarea cu structura șablonului de proiectare BusinessLogic și modelarea unui proiect ASP.NET, în baza lucrării de laborator Nr2, în conformitate cu modelulBusinessLogic.

Considerații teoretice

Proiectul MVC Asp.NET poate fi împărțit pe 3 nivele: nivelul prezentării, nivelul BusinessLogic și nivelul de acces la date. Această împărțire îmbunătățește procesul de dezvoltareși îmbunătățește performanța sistemului.

Nivelul *BusinessLogic* incapsulează toată logica de afaceri a proiectului, toate calculele necesare. Aces nivel primește obiecte din nivelul de acces la date și le transferă la nivelul de prezentare(Web) și invers. Obiectele Business stochează date și comportament, nu numai date.

Implementarea practică a sarcinilor de laborator

Deoarece principalele niveluri ale aplicației sunt Domain, Model, Data, Web, prin urmare,este necesar de împărțit sistemul proiectat în niveluri corespunzătoare.

Pentru a face acest lucru, este necesar de adăugat încă 3 proiecte suplimentare la soluția MS Visual Studio. Rezultatul este prezentat în figura 1.

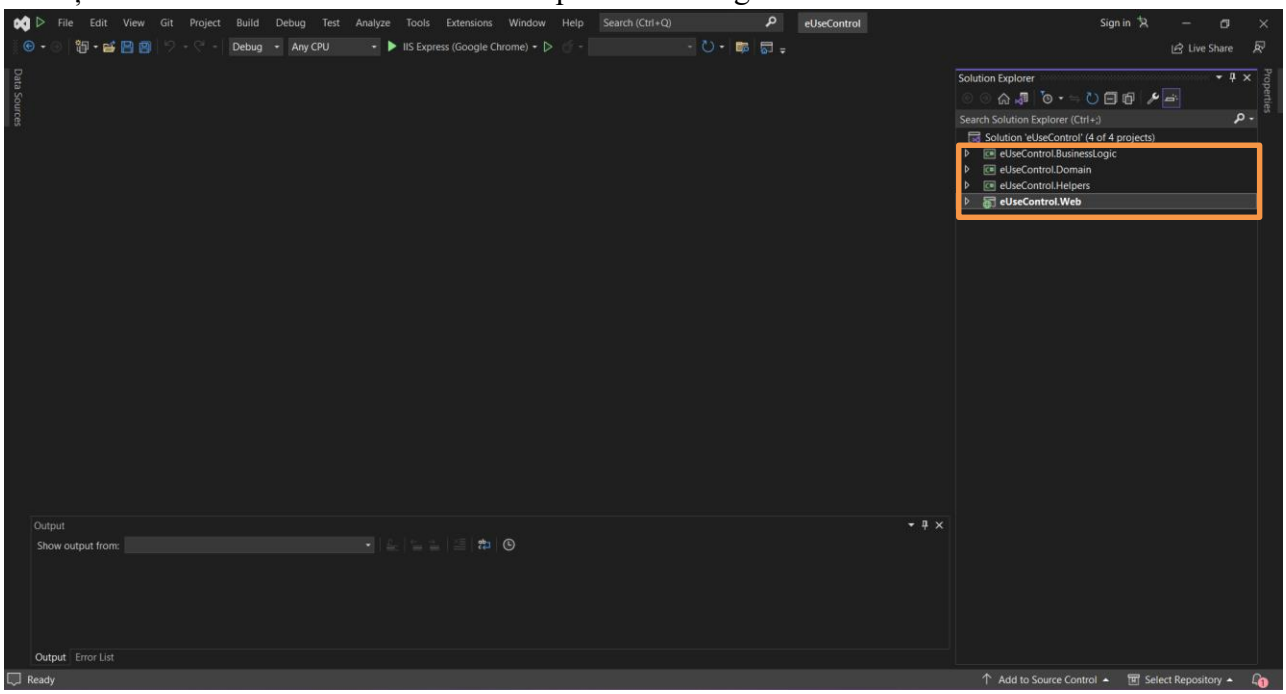

Figura 1 - Proiectele aplicației web

În figura de mai sus este reprezentat structura soluției care constă din 3 proiecteeUseControl, BusinessLogic, Domain și Helpers.

De asemenea este necesar să se stabilească legături (dependențe) între aceste proiecte.

Dependențele pentru stratul Businesslogic sunt reprezentate în figura 2.

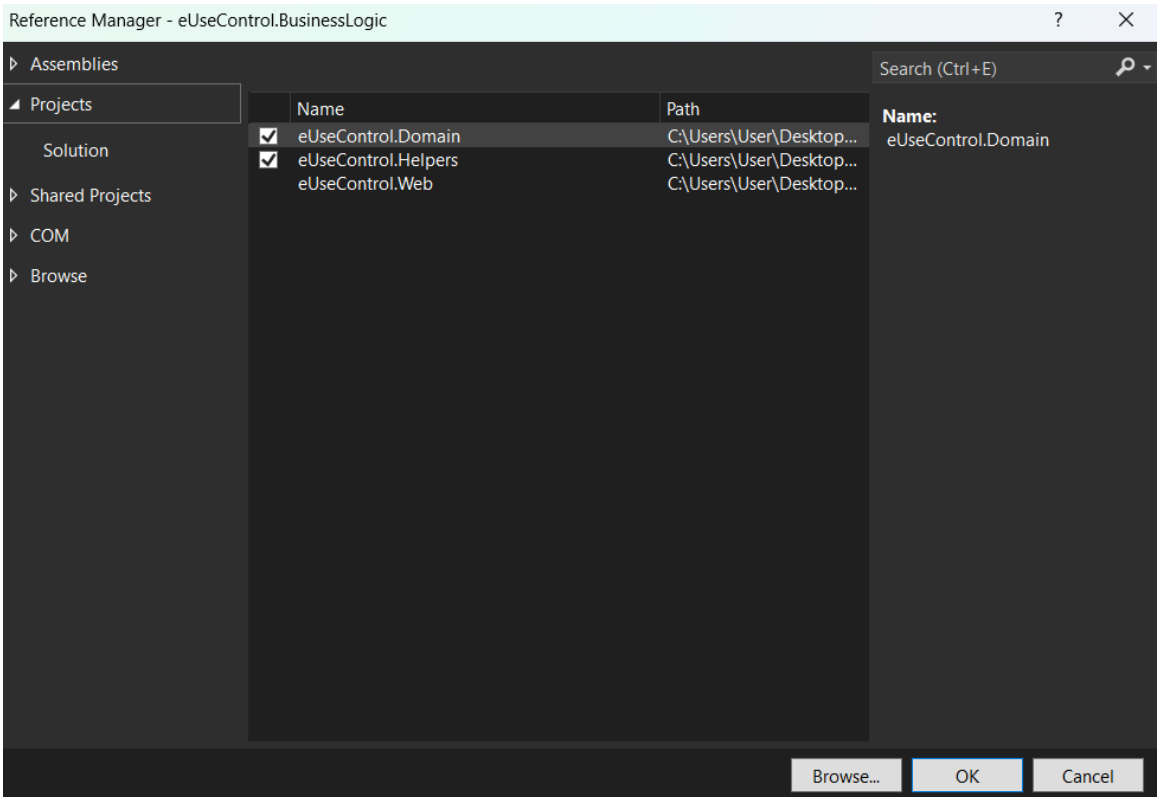

Figura 2 - Dependențele proiectului BusinessLogic

În același mod se stabilesc dependențele și pentru celelalte proiecte:

1) Proiectul Domain are referință la proiectul Helpers;
2) Proiectul de prezentare eUseControl are referință la proiectele BusinessLogic și Domain;
3) Proiectul Helpers nu are nici o referință;

La formarea proiectului BusinessLogic este nevoie de a crea două mape în interiorul său:Core și Interfaces. În mapa *Core*, se creează 2 clase *AdminApi* și *UserApi*.

În mapa *Interfaces* este creată clasa *ISession*. Conform regulilor C#, toate denumirileinterfețelor încep cu litera majuscula I.

Următorul element creat în cadrul proiectului *BusinessLogic* este clasa *SessionBL*, care seaflă în rădăcina a acestui proiect.

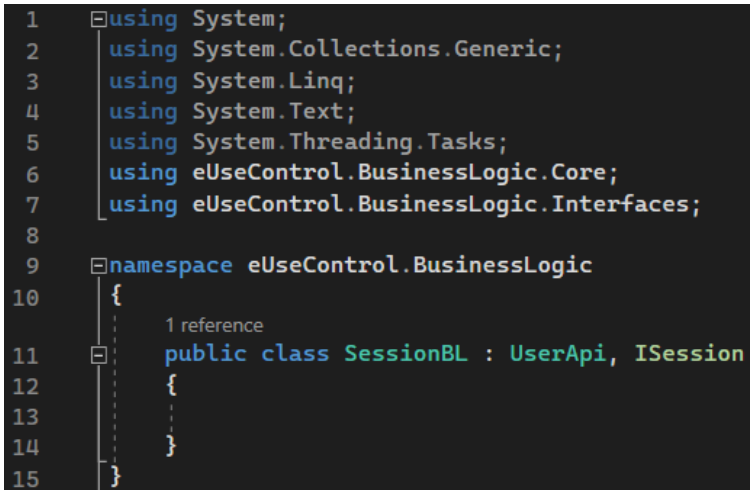

Figura 3 - Conținutul clasei *SessionBL*

Din fragmentul de cod prezentat se poate vedea că clasa SessionBL este moștenește clasaUserApi și implementează interfața ISession creată anterior.

S-a adăugat și clasa My*BussinesLogic*. Această clasă conține o metodă ce returnează unobiect de tip *ISession*.

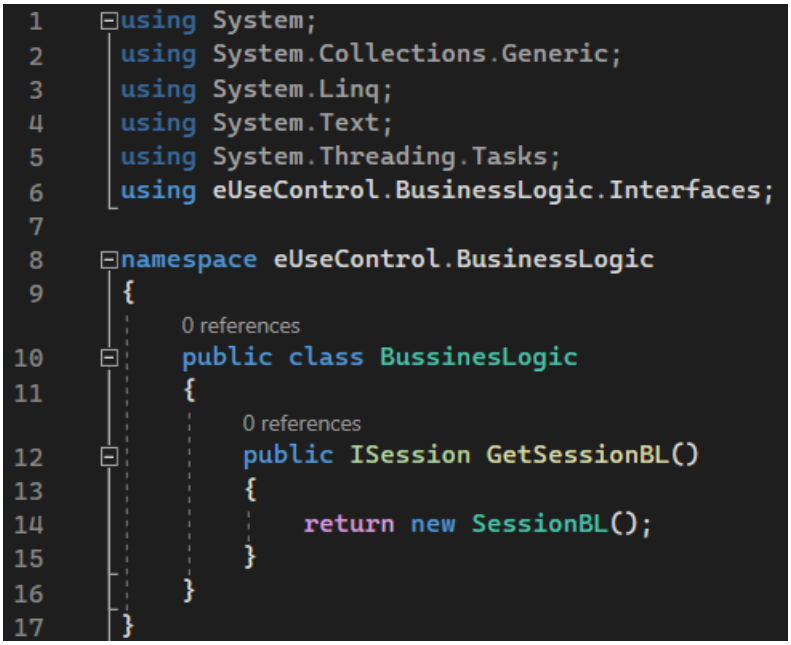

Figura 4 - Conținutul clasei *BussinesLogic*

Structura la momentul dat a proiectului *BusinessLogic* este reprezentată în figura 5.

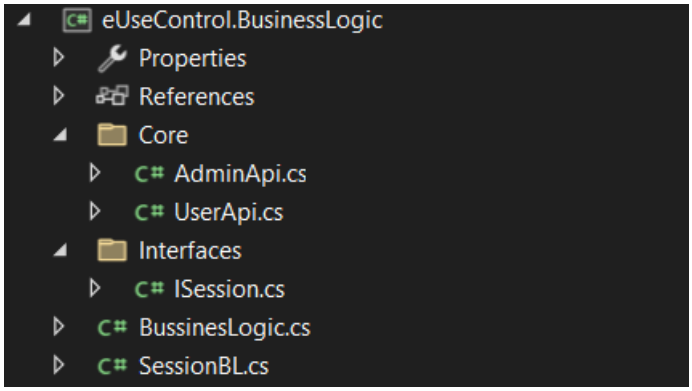

Figura 5 - Structura proiectului *BusinessLogic*

Următorul proiect la care trebuie de implementat funcționalitatea este proiectul *Domain*. Pentru aceasta se creează două mape: *Entities* și *Enums*. *Entities* conține clase care vor fi utilizateîn viitoarele lucrări cu baza de date. La această etapă, în mapa *Entities* se creează o mapă *User* ceconține două clase în interiorul său: *UloginData* și *UloginResp*.

Clasa *ULoginData* conține cîmpurile necesare pentru obținerea informațiilor de autentificare a utilizatorului. Pentru comoditate, am folosit proprietăți implementate automat, caresunt niște cîmpuri de rezervă privat, anonim, care poate fi accesat numai prin accesorii get și set.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace eUseControl.Domain.Entities.User
8   {
        2 references
9       public class ULoginData
10      {
            1 reference
11          public string Credential { get; set; }
            1 reference
12          public string Password { get; set; }
            1 reference
13          public string LoginIp{ get; set; }
            1 reference
14          public string LoginDateTime{ get; set; }
15      }
16  }
```

Figura 6 - Clasa *ULoginData*

Structura finală a proiectului *Domain* este reprezentată în figura 7.

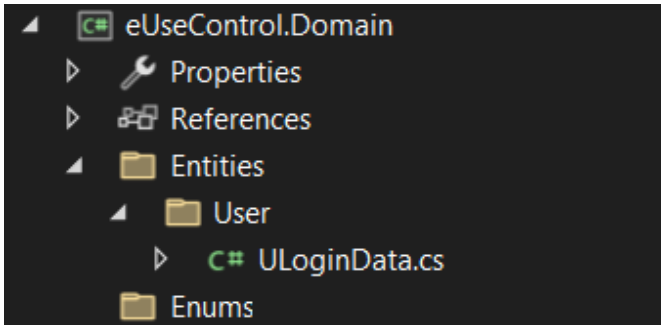

Figura 7 - Structura proiectului *Domain*

În nivelul de prezentare în mapa Controllers s-a creat un nou controller *LoginController* reprezentat în figura 8.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Web;
5   using System.Web.Mvc;
6   using eUseControl.BusinessLogic.Interfaces;
7   using eUseControl.Domain.Entities.User;
8
9   namespace eUseControl.Web.Controllers
10  {
        1 reference
11      public class LoginController : Controller
12      {
13          private readonly ISession _session;
14
            0 references
15          public LoginController()
16          {
17              var bl = new BusinessLogic();
18              _session = bl.GetSession();
19          }
20
21          // GET: Login
            0 references
22          public ActionResult Index()
23          {
24              return View();
25          }
26
27          [HttpPost]
28          [ValidateAntiForgeryToken]
29
            0 references
30          public ActionResult Index(UserLogin login)
31          {
```

Figura 8 – Inițializarea sesiunii

În figura de mai sus se reprezintă constructorul clasei *LoginController*, în interiorul căruia seinițializează sesiunea utilizatorului în aplicația noastră.

```
0 references
public ActionResult Index(UserLogin login)
{
    if (ModelState.IsValid)
    {
        ULoginData data = new ULoginData
        {
            Credential = login.Credential,
            Password = login.Password,
            LoginIp = Request.UserHostAddress,
            LoginDateTime = DateTime.Now
        };

        var userLogin = _session.UserLogin(data);
        if (userLogin.Status)
        {
            // ADD COOKIE

            return RedirectToAction("Index", "Home");
        }
        else
        {
            ModelState.AddModelError("", userLogin.StatusMsg);
            return View();
        }
    }
    return View();
}
```

Figura 9 – Metoda de acțiune *Index*

În figura de mai sus observăm metoda de acțiune *Index* care răspunde la solicitarea POST a motorului de rutare atunci cînd este trimis formularul de autentificare. Atributul [HttpPost] este untip de supraîncărcare a metodei.

Concluzii:

Lucrarea de laborator a avut ca obiectiv familiarizarea cu structura modelului de proiectare BusinessLogic și modelarea proiectului finalizat ASP.NET, conform modelului BusinessLogic. Bazele teoretice ale proiectului MVC ASP.NET au fost prezentate, inclusiv împărțirea acestuia în 3 niveluri: nivelul de prezentare, nivelul BusinessLogic și nivelul de acces la date, care îmbunătățesc procesul de dezvoltare și performanța sistemului.

Pentru simplificarea comparării claselor de modele, a fost prezentată extensia AutoMapper, care permite conversia unui obiect în altul și poate fi utilă în cazul în care obiectul depășește limitele aplicației sau nivelului. În general, lucrarea a oferit o perspectivă completă asupra structurii și importanței modelului de proiectare BusinessLogic în dezvoltarea de proiecte ASP.NET.