Ministerul Educaţiei și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică

RAPORT

Lucrarea de laborator nr.3
la Tehnologii Web

Au efectuat: Bagrin Andeea,
st. gr. TI-216 Miron Anastasia,
Muntean Mihai,
Ungureanu Liviu

A verificat:
asist. univ. Gaidarji Alina

Chişinău - 2023

Lucrare de laborator nr. 3

Tema: Modele de proiectare. Pattern BusinessLogic

Scopul lucrării: Studierea pattern-ului BusinessLogic.

Sarcina de lucru: Familiarizarea cu structura modelului de proiectare BusinessLogic și modelarea proiectul finalizat ASP.NET, obținut ca rezultat al efectuării lucrărilor de laborator nr.2, în conformitate cu modelul BusinessLogic.

Mersul lucrării:

Proiectul MVC ASP.NET poate fi împărțit în mod condiționat în 3 niveluri: nivelul de prezentare(.Web), nivelul de business logică(.BusinessLogic) și nivelul de acces la date(.Domain). Această separare îmbunătățește procesul de dezvoltare și îmbunătățește performanța sistemului.

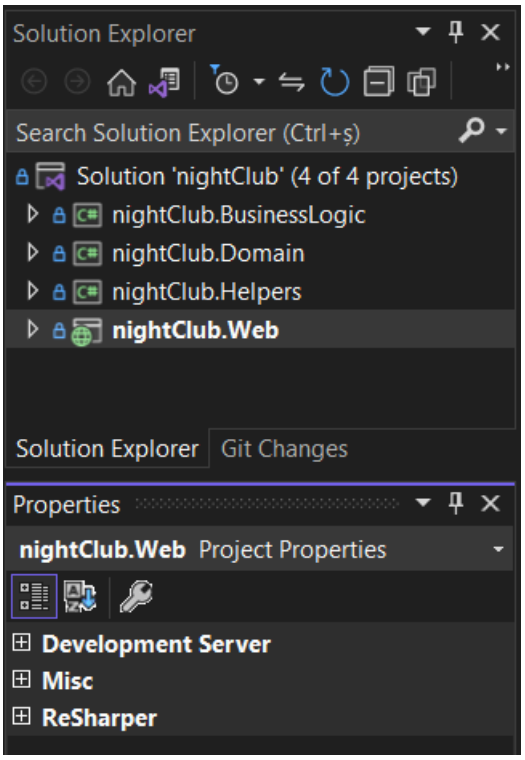


Figura 1 – Nivelele proiectului

În continuare se stabilesc legăturile dintre nivele proiectului:

- BusinessLogic: Domain și Helpers;
- Domain: Helpers;
- Web: BusinessLogic și Domain.(Figura 2)

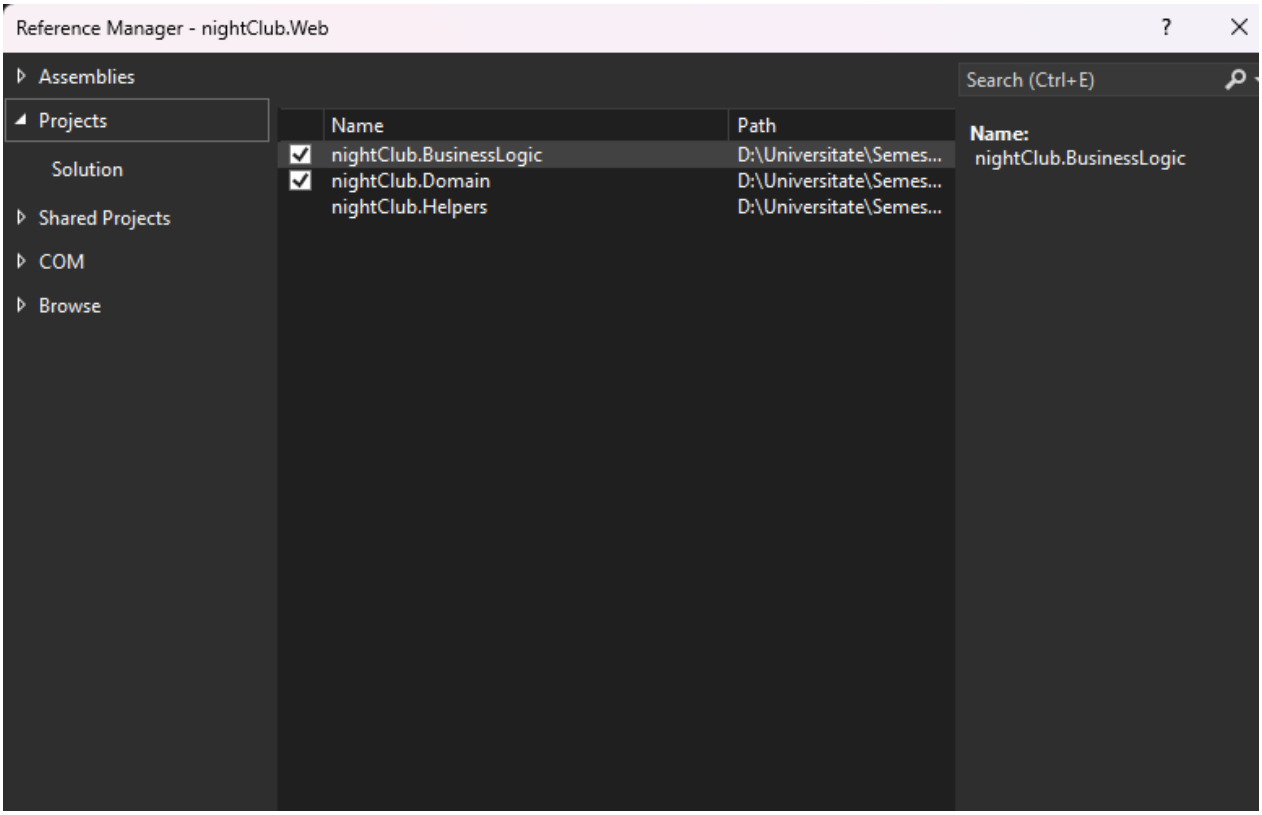


Figura 2 – Stabilirea legăturilor dintre nivele

Nivelul *BusinessLogic* încorporează întreaga logică de afaceri a proiectului, este responsabil pentru prelucrarea datelor și aplicarea logicii specifice proiectului dezvoltat. Acest nivel se ocupă cu toate calculele, validările și verificările necesare pentru a asigura integritatea și corectitudinea datelor. Scopul principal al nivelului de business logică este de a asigura că datele sunt procesate în conformitate cu regulile și politicile de afaceri, precum și de a asigura că datele sunt corecte și relevante pentru utilizatori.

Nivelul de prezentare a datelor transmite datele introduse de utilizator la nivelul de business logică, care le prelucrează și returnează datele procesate înapoi la nivelul de prezentare a datelor pentru a fi afișate utilizatorului. Acest proces se repetă până când utilizatorul obține rezultatul dorit.

Structura proiectului *BusinessLogic* este vizibilă în figura 3.

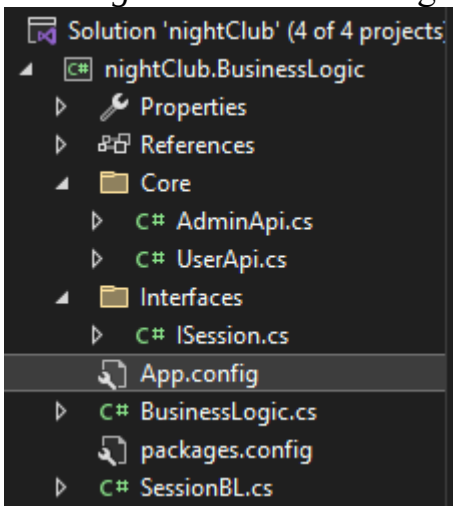


Figura 3 – Structura proiectului BusinessLogic

După cum se vede în figura 3, la formarea nivelului BusinessLogic adăugăm

două fișiere în interiorul său: *Core* și *Interfaces*, unde la rândul său, în fișierul *Core* se creează 2 clase *AdminApi* și *UserApi*, iar în fișierul *Interfaces* este creată clasa *ISession*.

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

```
10  namespace nightClub.BusinessLogic
11  {
        1 reference
12      public class SessionBL: UserApi, ISession
13      {
            2 references
14          public ULoginResponse UserLogin(ULoginData data)
15          {
16              throw new NotImplementedException();
17          }
18
            1 reference
19          public string GetCookies(ULoginData loginData)
20          {
21              return UserAssignCookies(loginData);
22          }
23
            1 reference
24          public bool ValidateCookies(string cookies)
25          {
26              return UserCookiesValidator(cookies);
27          }
```

Figura 4 – Conținutul SessionBL

Analizând acest conținut, vedem că clasa *SessionBL* generalizează clasa *UserAPi* și implementeaza intefața *ISession*.

De asemenea s-a adăugat și clasa *BusinessLogic*, care conține o metodă ce returnează un obiect de tip *ISession*.

```
1   using nightClub.BusinessLogic.Interfaces;
2
3   namespace nightClub.BusinessLogic
4   {
        1 reference
5       public class BussinesLogic
6       {
            1 reference
7           public ISession GetSessionBL()
8           {
9               return new SessionBL();
10          }
11      }
12  }
```

Figura 5 – Conținutul BusinessLogic

Următorul nivel este Domain, structura căruia este în figura 6.

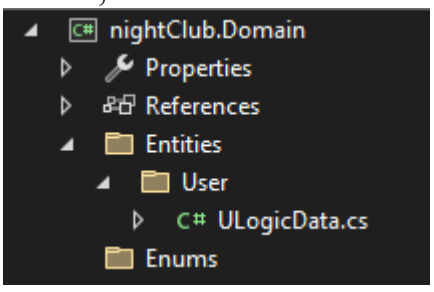


Figura 6 – Structura proiectului Domain

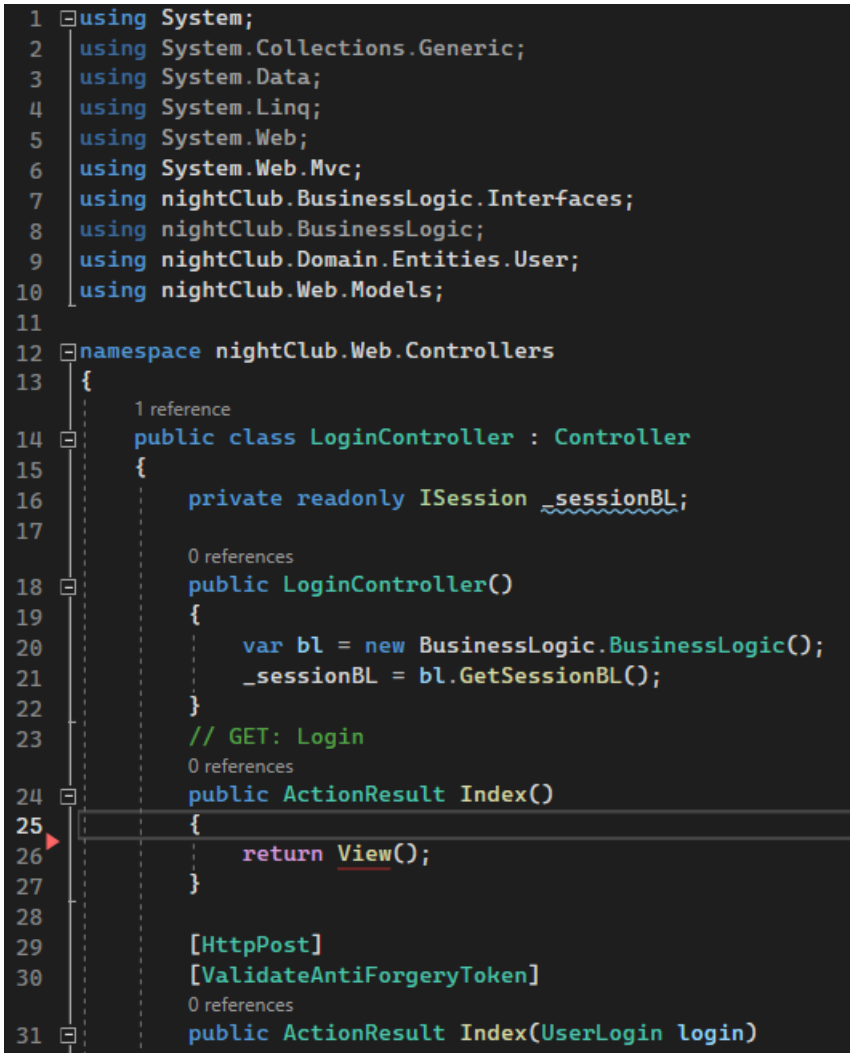
Pentru acest nivel s-au creeat 2 fișiere: *Entities* și *Enums*. *Entities* conține clase care vor fi utilizate în viitoarele lucrări cu baza de date, *Enums* destinat enumerărilor. La moment, în fișierul *Entities* se creează un fișier *User* ce conține o singură clasă în interiorul său – *UloginData*.

Clasa *ULoginData* conține cîmpurile necesare pentru obținerea informațiilor de autentificare a utilizatorului. Pentru comoditate, am folosit proprietăți automate(auto-properties),proprietăți speciale care permit accesul la câmpurile private ale clasei utilizând get/set acessors implicit generați.

```
1  namespace nightClub.Domain.Entities.User
2  {
       2 references
3      public class ULoginData
4      {
           1 reference
5          public string Credential { get; set; }
           1 reference
6          public string Password { get; set; }
           1 reference
7          public string LoginIp { get; set; }
           1 reference
8          public string LoginDateTime { get; set; }
9      }
10 }
```

Figura 7 – Conținutul UloginData

La final, în nivelul de prezentare, în fișierul Controllers s-a creat un nou controller – *LoginController* afișat în figurele 8 și 9.



```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using nightClub.BusinessLogic.Interfaces;
using nightClub.BusinessLogic;
using nightClub.Domain.Entities.User;
using nightClub.Web.Models;

namespace nightClub.Web.Controllers
{
    1 reference
    public class LoginController : Controller
    {
        private readonly ISession _sessionBL;

        0 references
        public LoginController()
        {
            var bl = new BusinessLogic.BusinessLogic();
            _sessionBL = bl.GetSessionBL();
        }
        // GET: Login
        0 references
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        0 references
        public ActionResult Index(UserLogin login)
```

Figura 8 – Conținutul LoginController

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

```
29          [HttpPost]
30          [ValidateAntiForgeryToken]
            0 references
31          public ActionResult Index(UserLogin login)
32          {
33              if (ModelState.IsValid)
34              {
35                  ULoginData data = new ULoginData()
36                  {
37                      Credential = login.Credential,
38                      Password = login.Password,
39                      LoginIP = Request.UserHostAddress,
40                      LoginDateTime = DateTime.Now
41                  };
42                  var userLogin = _sessionBL.UserLogin(data);
43                  if (userLogin.Status)
44                  {
45                      //ADD COOKie
46                      return RedirectToAction("Index", controllerName: "Home");
47                  }
48                  else
49                  {
50                      ModelState.AddModelError(key: "", errorMessage: userLogin.StatusMsg);
51                      return View();
52                  }
53              }
54              return View();
55          }
```

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 un tip de supraîncărcare a metodei.

Concluzii:

Familiarizarea cu modelul de proiectare BusinessLogic este esențială pentru dezvoltarea unui proiect ASP.NET. Acest model permite separarea logică a aplicației de baza de date și a interfeței utilizatorului, ceea ce face ca aplicația să fie mai ușor de întreținut și de dezvoltat pe viitor.

Proiectul modelat anterior a fost actualizat și completat cu modificările necesare.

Pentru a simplifica compararea claselor de modele în proiectele MVC ASP.NET, se folosește extensia AutoMapper. Cu ajutorul acestei biblioteci devine posibilă conversia unui obiect în altul. Cartografia poate fi utilă în cazul acestora când obiectul depășește limitele aplicației sau nivelului.