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, nivelul *BusinessLogic* și nivelul de acces la date. Această separare îmbunătățește procesul de dezvoltare și îmbunătățește performanța sistemului.

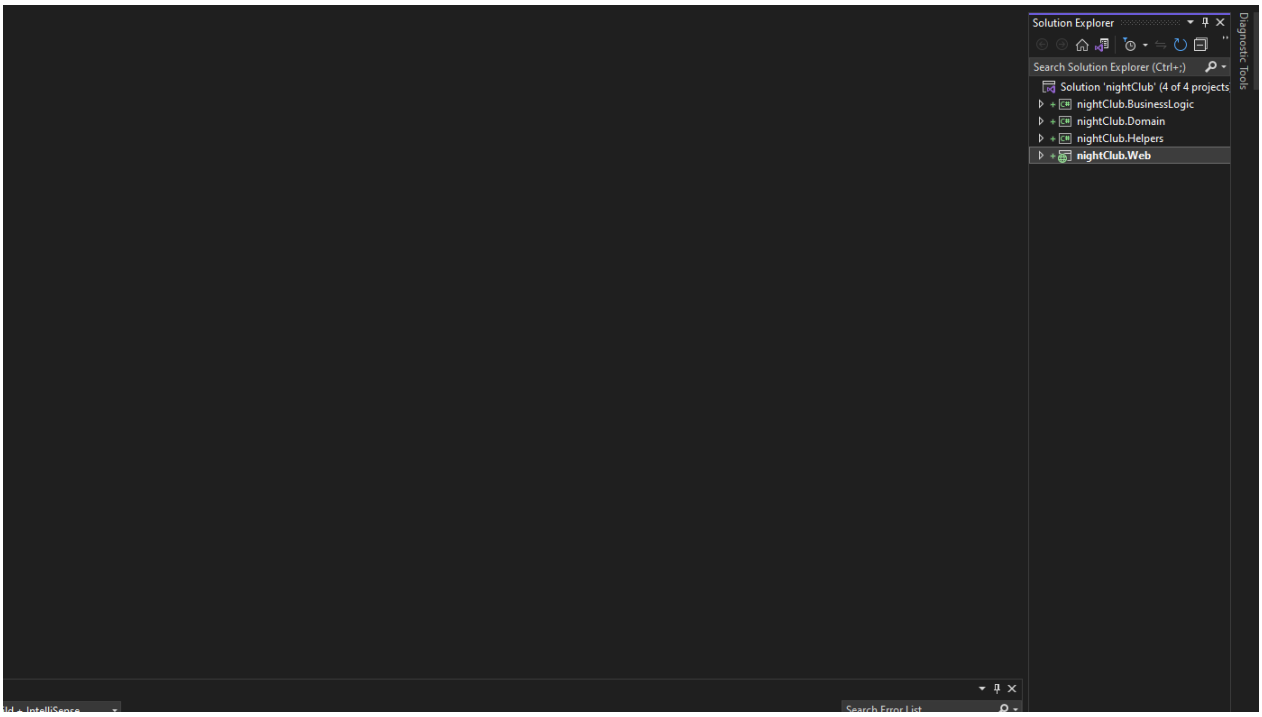


Figura 1 – Nivelele proiectului

În continuare se stabilesc legăturile dintre nivele proiectului:

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

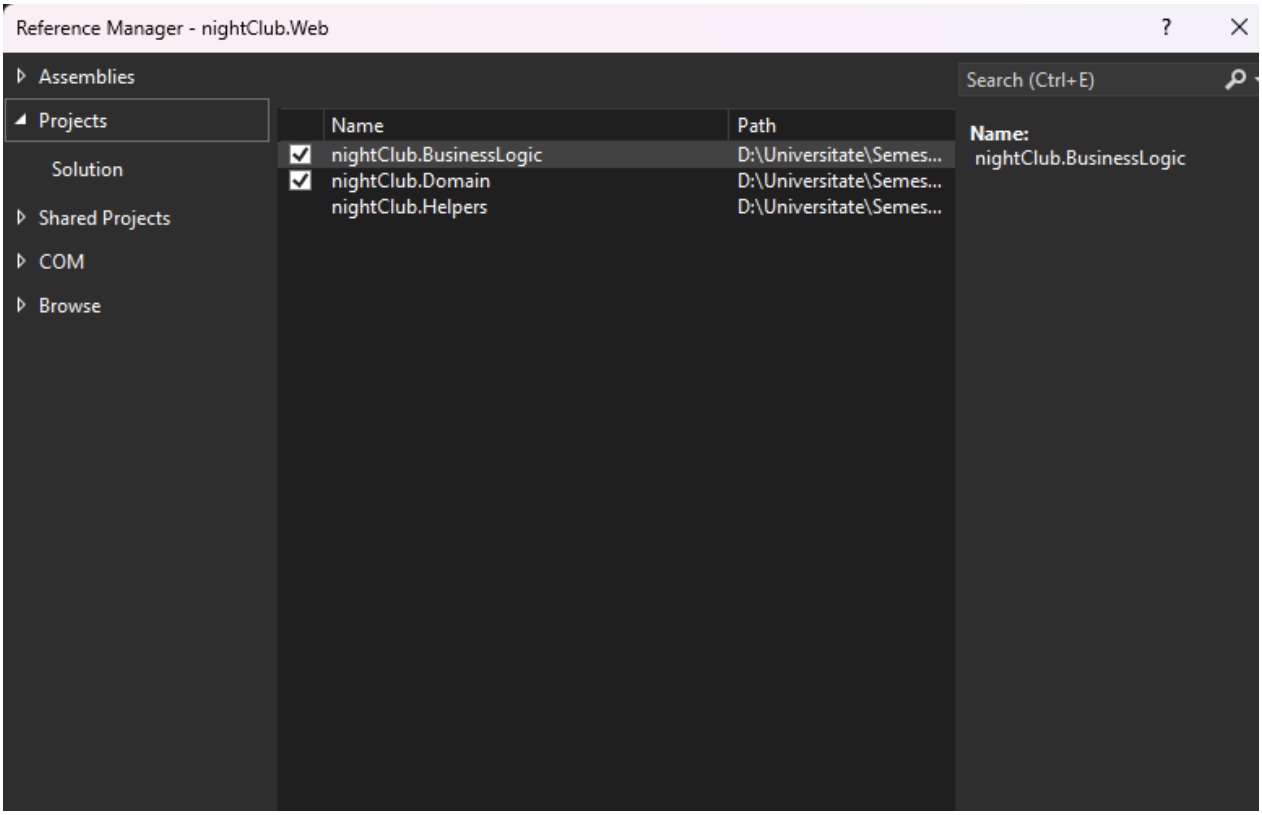


Figura 2 – Stabilirea legăturilor dintre nivele

Nivelul *BusinessLogic* încorporează întreaga logică de afaceri a proiectului, toate calculele necesare. Acest strat obține obiecte din stratul de acces la date și le transmite la nivelul reprezentărilor sau invers. Obiectele de afaceri stochează date și comportamentul, nu doar date.

Structura proiectului *BusinessLogic* este vizibilă în figura 3.

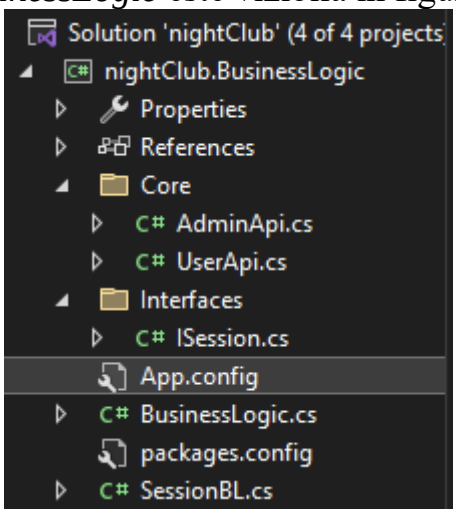


Figura 3 – Structura proiectului BusinessLogic

După cum se vede în figura 3, la formarea proiectului BusinessLogic este nevoie de a crea două mape în interiorul său: *Core* și *Interfaces*,unde la rândul său, în mapa *Core* se creează 2 clase *AdminApi* și *UserApi*, iar în mapa *Interfaces* este creată clasa *ISession*.

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

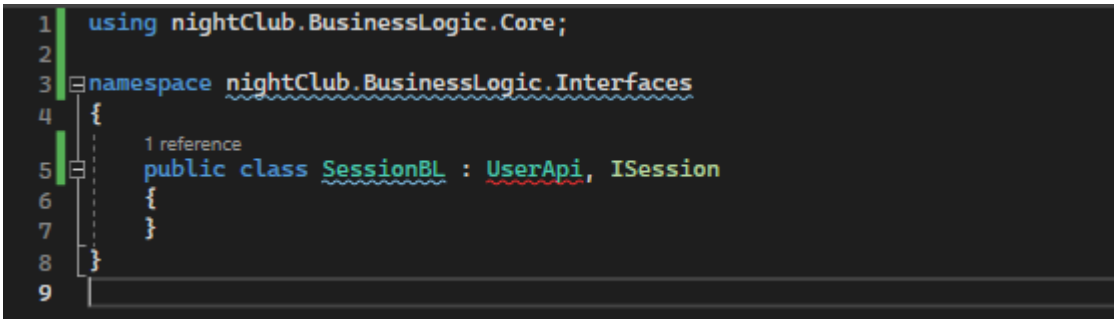


Figura 4 – Conținutul SessionBL

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 și returnează un obiect de tip *ISession*.

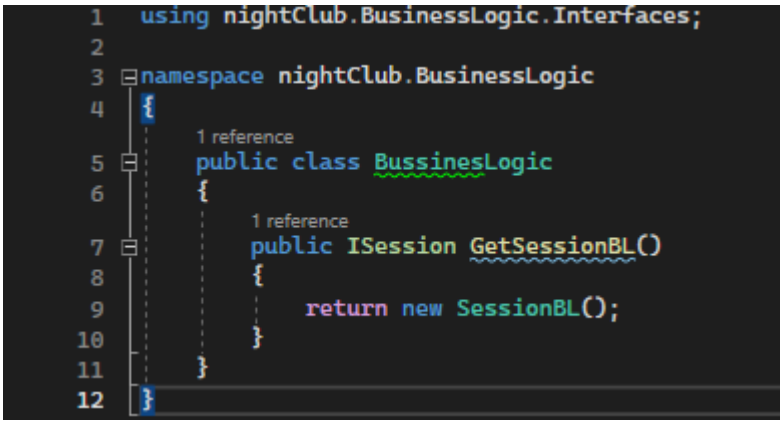


Figura 5 – Conținutul BusinessLogic

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

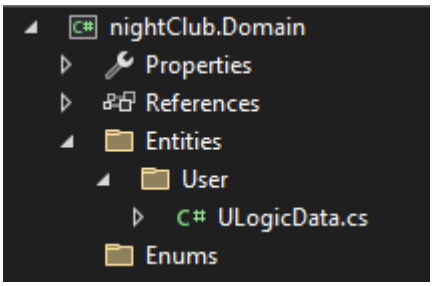


Figura 6 – Structura proiectului Domain

Pentru acest proiect s-a creeat 2 mape: *Entities* și *Enums*. *Entities* conține clase care vor fi utilizate în viitoarele lucrări cu baza de date. La moment, în mapa *Entities* se creează o mapă *User* ceconține o singură clasă în interiorul său – *UloginData*.

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

```
namespace nightClub.Domain.Entities.User
{
    2 references
    public class ULoginData
    {
        1 reference
        public string Credential { get; set; }
        1 reference
        public string Password { get; set; }
        1 reference
        public string LoginIp { get; set; }
        1 reference
        public string LoginDateTime { get; set; }
    }
}
```

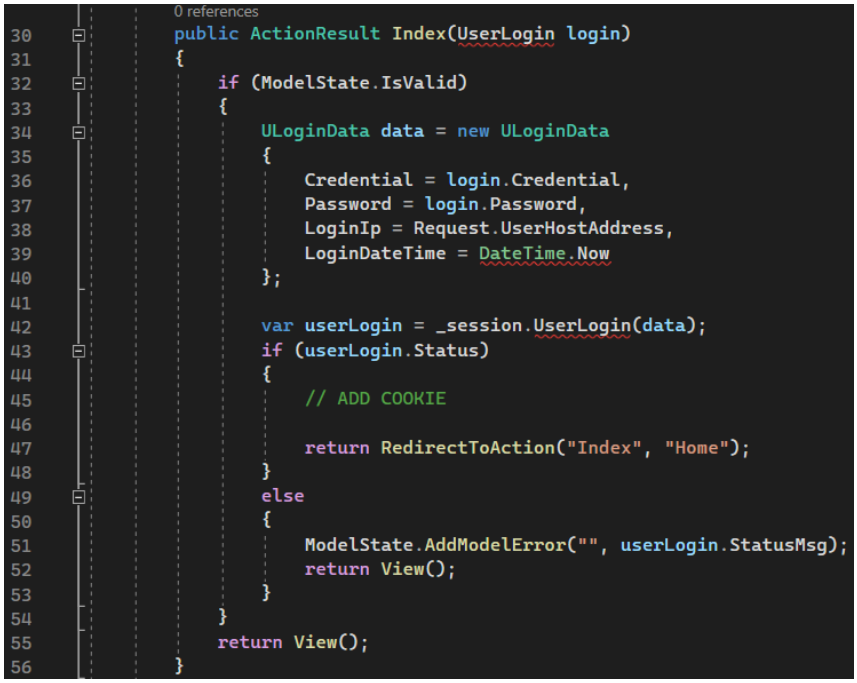
Figura 7 – Conținutul UloginData

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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using eUseControl.BusinessLogic.Interfaces;
using eUseControl.Domain.Entities.User;

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

        0 references
        public LoginController()
        {
            var bl = new BusinessLogic();
            _session = bl.GetSession();
        }

        // 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.



```
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 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.