Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Raport

la Lucrarea de laborator nr. 3
la disciplina Tehnologii Web

Tema: Modele de proiectare. Pattern BusinessLogic

Studentul gr. TI-173: Heghea Nicolae
Conducător: asistent universitar, Rusu Cristian

Chișinău 2019

Cuprins

1. Scopul :

Familiarizarea cu structura și designul modelului BusinessLogic și modelarea unui proiectul finalizat ASP.NET, primit din rezultatul din execuție lucrarii de laborator nr. 2, în conformitate cu modelul BusinessLogic.

2. Sarcina lucării

Implementarea cunoștințelor acumulate pentr a crea modelul BusinessLogic întrun proiect în ASP .Net.

3. Scurtă teorie

Proiectul MVC ASP.NET poate fi împărțit în 3 niveluri: vizualizări, niveluri BusinessLogic și niveluri de acces la date. Această separare îmbunătățește procesul de dezvoltare și îmbunătățește performanța sistemului.

Nivelul Business Logic 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 comportament, nu doar date.

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.

4. Realizare

Deoarece nivelurile principale ale aplicației sunt Domeniu, Model, Date, Web, este necesar să se împartă sistemul proiectat în nivele adecvate.

Pentru aceasta, adăugați 3 proiecte suplimentare la soluția MS Visual Studio. Arborele de decizie rezultat este prezentat în Figura 1.

Conținutul proiectului :

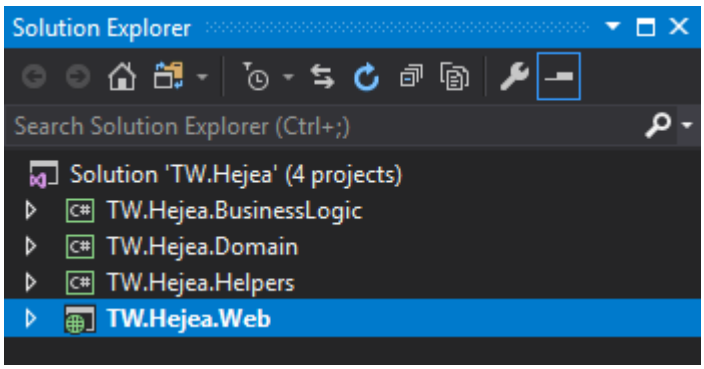


Figura 1 Stratura proiectului

Pentru a continua munca cu succes cu structura rezultată, este necesar să se stabilească legături în cadrul acestor biblioteci și modul în care acestea vor fi legate între ele. Legăturile pentru toate bibliotecile sunt prezentate în Figura 2.

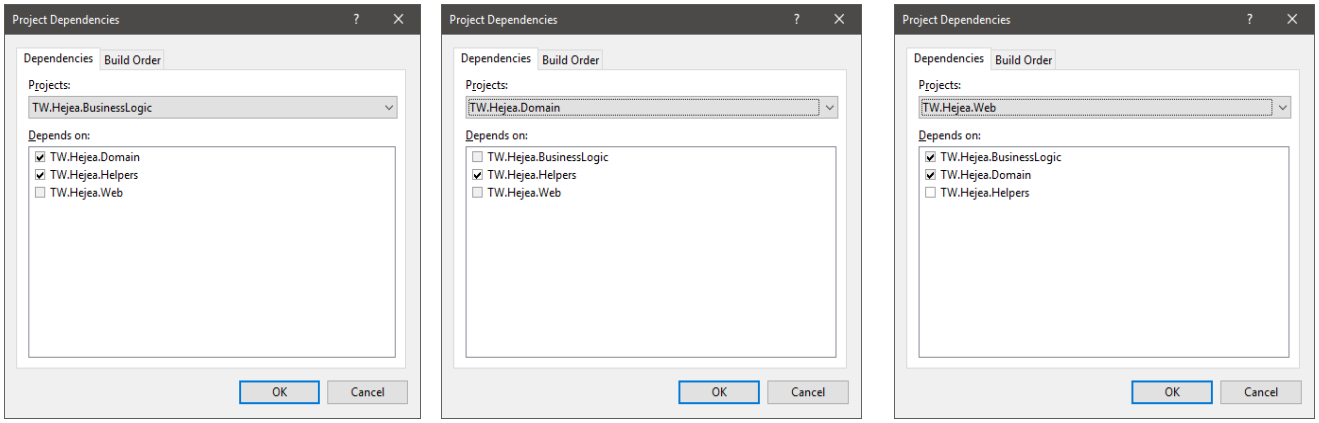


Figura 2 Legaturile între biblioteci

Următorul pas va fi formarea proiectului BusinessLogic. Pentru a face acest lucru, trebuie să creați două foldere în interiorul acestuia: Core, Interfaces. Clasa AdminApi si UserApi sunt create în folderul Core. Procesul de creare este prezentat în Figura 3.

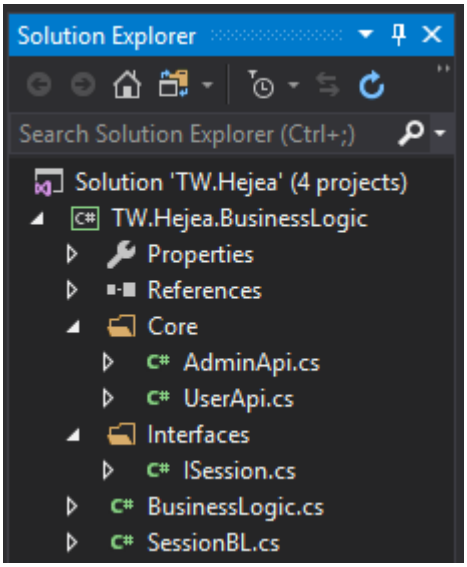


Figura 3 Biblioteca BusinessLogic

Conținutul fișierului *AdminApi.cs*

```
namespace TW.Hejea.BusinessLogic.Core
{
    class AdminApi
    {
    }
}
```

Conținutul fișierului *UserApi.cs*

```
namespace TW.Hejea.BusinessLogic.Core
{
    public class UserApi
    {
    }
}
```

Următorul pas este să completați dosarul „**Interface**” în care este creată interfața *ISession*. Conform regulilor limbajului C #, toate denumirile de interfețe încep cu litera mare „*I*”. Procesul de creare a interfeței în cauză este prezentat în Figura 3.

Conținutul fișierului *ISession.cs*

```
using TW.Hejea.Domain.Entities.User;

namespace TW.Hejea.BusinessLogic.Interfaces
{
    public interface ISession
    {
        ULoginResp UserLogin(ULoginData data);
    }
}
```

Conținutul fișierului *BusinessLogic.cs*

```
using TW.Hejea.BusinessLogic.Interfaces;

namespace TW.Hejea.BusinessLogic
{
    public class BussinesLogic
    {
        public ISession GetSessionBL()
        {
            return new SessionBL();
        }
    }
}
```

Conținutul fișierului *SessionBL.cs*

```
using TW.Hejea.BusinessLogic.Core;
using TW.Hejea.BusinessLogic.Interfaces;
using TW.Hejea.Domain.Entities.User;

namespace TW.Hejea.BusinessLogic
{
    public class SessionBL : UserApi, ISession
    {
        public ULoginResp UserLogin(ULoginData data)
        {
            return new ULoginResp();
        }
    }
}
```

Următorul pas este să creăm TW.Hejea.**Domain**, în care vom adăuga 2 foldere *Entities* și *Enums*. Iar în *Entities* vom adăga folderul *User*, care va conține 2 fișiere *ULoginData.cs* și *ULoginResp.cs*. Structura este reprezentată în figura 4.

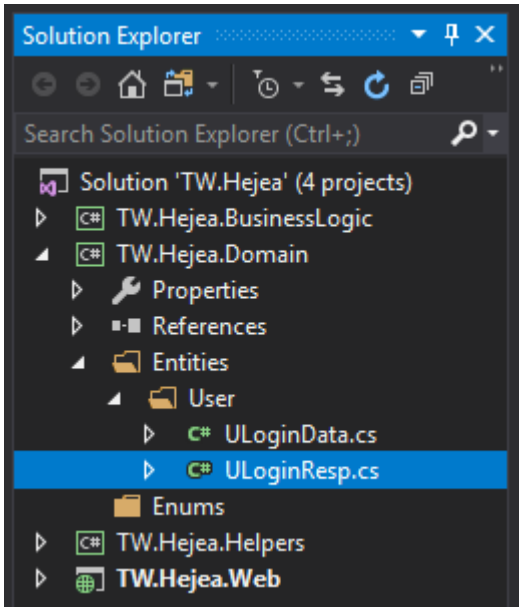


Figura 4 Conținutul bibliotecii TW.Hejea.**Domain**

Conținutul fișierului *ULoginData.cs*

```
using System;

namespace TW.Hejea.Domain.Entities.User
{
    public class ULoginData
    {
        public string Credential { get; set; }
        public string Password { get; set; }
        public string LoginIp { get; set; }
        public DateTime LoginDateTime { get; set; }
    }
}
```

Conținutul fișierului *ULoginResp.cs*

```
namespace TW.Hejea.Domain.Entities.User
{
    public class ULoginResp
    {
        public bool Status { get; set; }
        public string StatusMsg { get; set; }
    }
}
```

În biblioteca TW.Hejea.**Web**, în folderul **Controllers**, vom crea fișierul *LoginController.cs*.

Conținutul fișierului *LoginController.cs*

```
using AutoMapper;
using System;
using System.Web.Mvc;
using TW.Hejea.BusinessLogic;
using TW.Hejea.BusinessLogic.Interfaces;
using TW.Hejea.Domain.Entities.User;
using TW.Hejea.Web.Models;

namespace TW.Hejea.Web.Controllers
{
    public class LoginController : Controller
    {
        private readonly ISession _session;
        public LoginController()
        {
            var bl = new BussinesLogic();
            _session = bl.GetSessionBL();
        }

        // GET: Login
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Index(UserLogin login)
        {
            if (ModelState.IsValid)
            {
                Mapper.Initialize(cfg => cfg.CreateMap<UserLogin, ULoginData>());
                var data = Mapper.Map<ULoginData>(login);
                data.LoginIp = Request.UserHostAddress;
                data.LoginDateTime = DateTime.Now;
                var userLogin = _session.UserLogin(data);
            }
            return View();
        }
    }
}
```

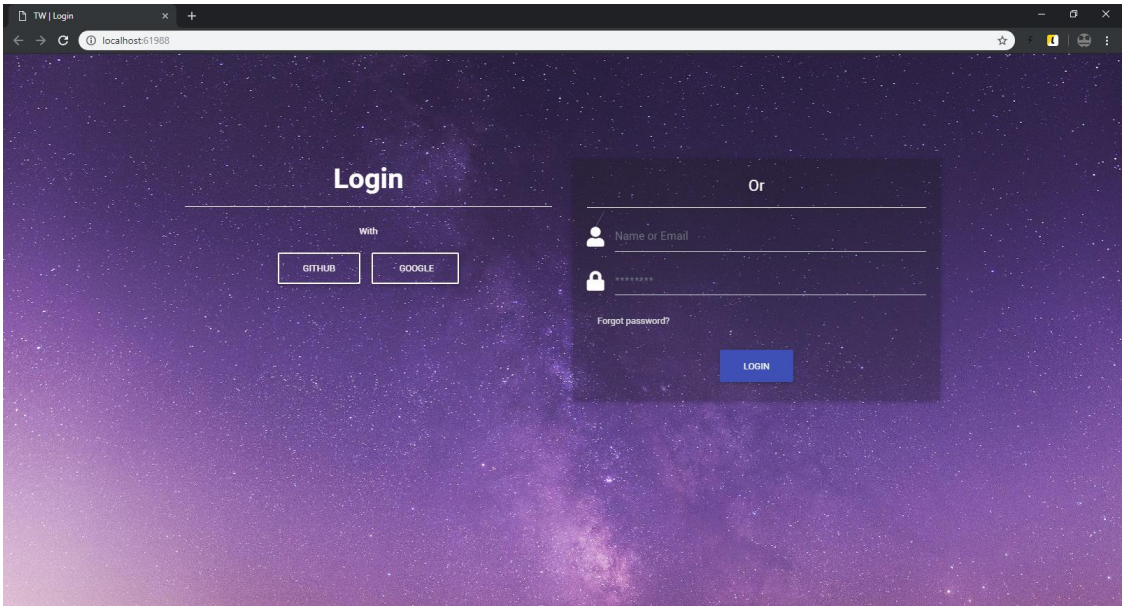
Iar în folderul **Modules**, vom adăuga fișierul *UserLogin.cs*.

Conținutul fișierului *UserLogin.cs*

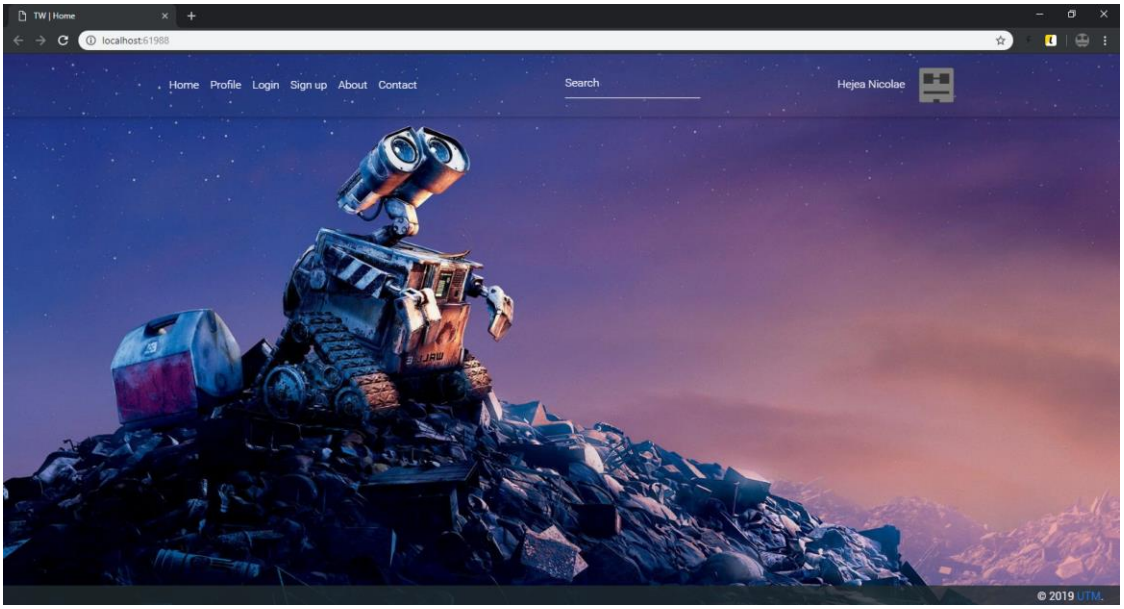
```
namespace TW.Hejea.Web.Models
{
    public class UserLogin
    {
        public string Credential { get; set; }
        public string Password { get; set; }
    }
}
```

5. Rezultat

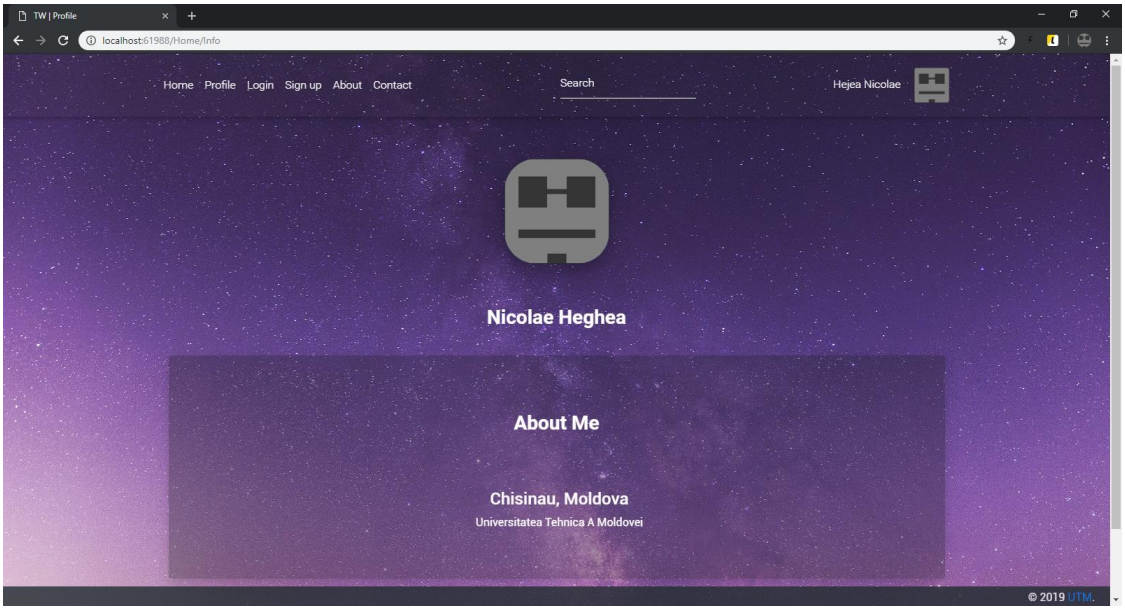
Pagina *login* :



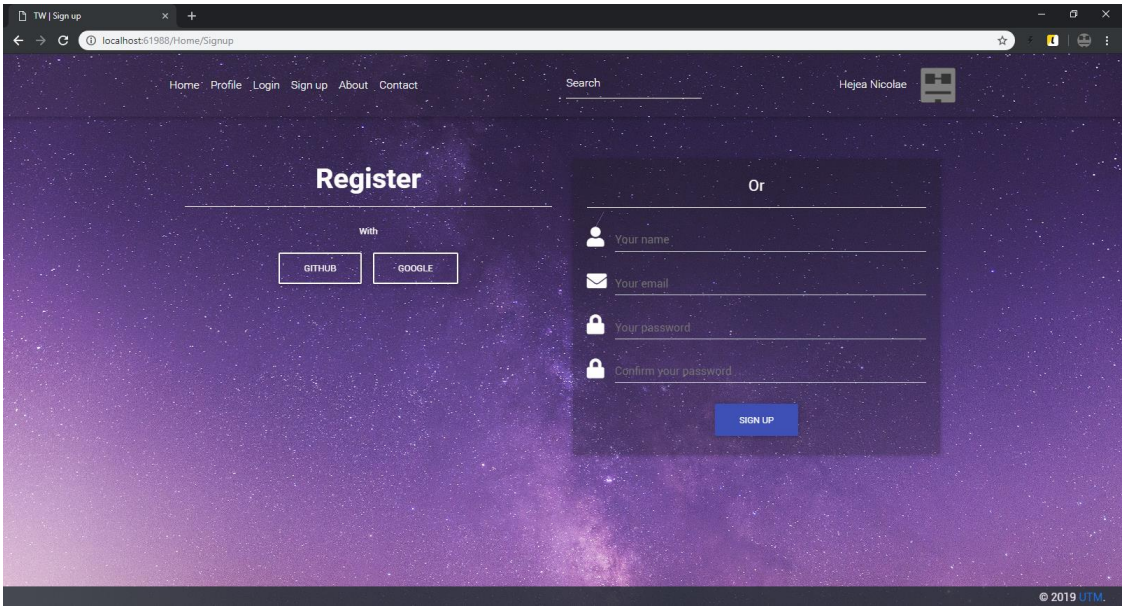
Pagina *home* :



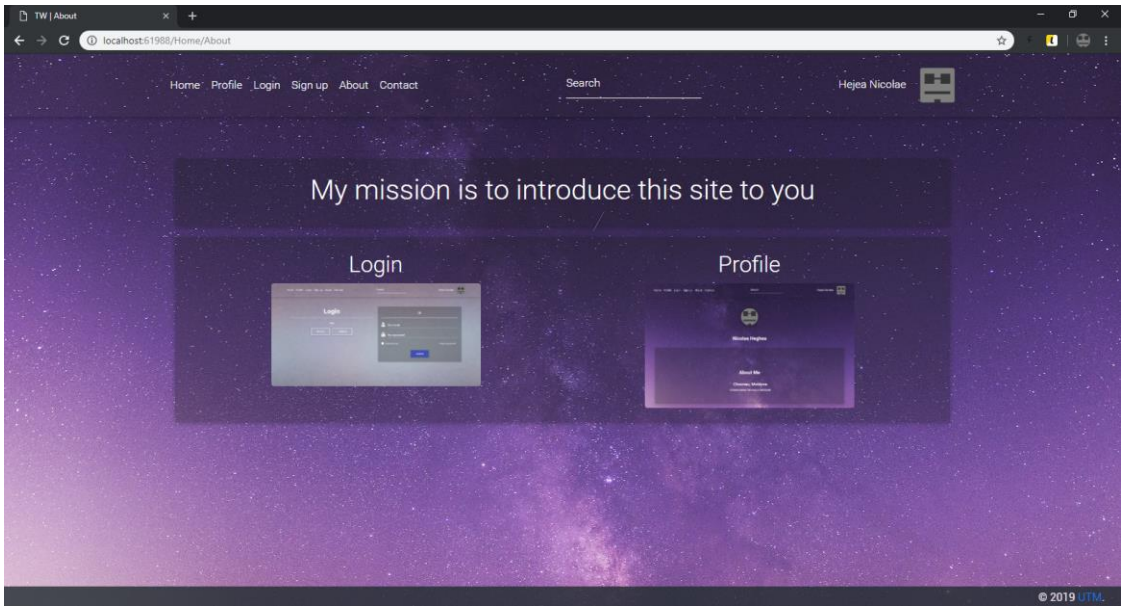
Pagina *profile* :



Pagina *sign up* :



Pagina *about* :



Pagina *contact* :