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

Raport

la Lucrarea de laborator nr. 5
la disciplina Tehnologii Web

Tema: Admin, User interface

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

Chișinău 2019

Cuprins

1. Scopul :

Implementarea restricționării accesului utilizatorilor la anumite pagini ale site-ului (admin, utilizator).

2. Scurtă teorie

Implementarea mecanismului de restricționare a anumitor grupuri de utilizatori launele acțiuni disponibile pe site trebuie să utilizeze mecanisme de atribute și de atributul ActionFilterAttribute.

Atributul Admin Mod vă permite să restricționați accesul la toate acțiunile controlerului, precum și numai la anumite acțiuni ale controlorului.

Pentru a utiliza atributul Admin Mod Trebuie să adăugați o adnotare înaintea numelui metodei sau clasei.

3. Realizare

În același timp, deoarece metodele onActionExecuting ale acestor atribute trebuie să fie sunt executate secvențial, mai întâi CookieCheck și apoi AdminMod, folosite ordinul de proprietate suplimentar, care vă permite să executați metode într-o anumită ordine.

```
TW.Hejea.Domain
1   using System;
2   using TW.Hejea.Domain.Enums;
3
4   namespace TW.Hejea.Domain.Entities.User
5   {
6       public class UserMinimal
7       {
8           public int Id { get; set; }
9           public string Username { get; set; }
10          public string Email { get; set; }
11          public DateTime LastLogin { get; set; }
12          public string LasIp { get; set; }
13          public URole Role { get; set; }
14      }
15  }
16
```

Figura 1 Entitatea utilizator

Următorul pas este fișier pentru generarea cookie-urilor în ajutoare.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

namespace TW.Hejea.Helpers
{
    public static class CookieGenerator
    {
        private const string SaltData = "QADLz4qk3rVgBSGjDfAH3XWV" + "qKKagMXezBPv7TmXvwnXDDeR" +
"pHaLBv4JnTGRwLg9tzbmV77g"
                                        + "8DUEAEa6JPv66hy7SwHBL4z4" + "FbGdh2MVs4kq9RcaZEAszuP5"
+ "ccLsEfqCpwdSvVVt479DCZrw"
                                        + "jSHrJVwaja9WQaWAmEY9NsPv" + "EHKnFwHTGAvPXpjpCxkbedYq"
+ "uEauLvZLphwmJLUteZ4QAXU6" + "Z4F3PDmh3wsQXvSctQBHvNWf";

        private static readonly byte[] Salt = Encoding.ASCII.GetBytes(SaltData);

        public static string Create(string value)
        {
            return EncryptStringAes(value,
"BjXNmq5MKKaraLwxz9uaATvFwE4Rj679KguTRE8c2j56FnkuKJKfkGbZEeDGFDvsGYNHpUXFUUUuUHBR4UV3T2kumguhubg6G
pt7CyqGDbUPrMvPc67kX3yP");
        }
```

```
        public static string Validate(string value)
        {
            return DecryptStringAes(value,
"BjXNmq5MKKaraLwxz9uaATvFwE4Rj679KguTRE8c2j56FnkuKJKfkGbZEeDGFDvsGYNHpUXFUUUuUHBR4UV3T2kumguhubg6G
pt7CyqGDbUPrMvPc67kX3yP");
        }

        /// <summary>
              /// Encrypt the given string using AES.  The string can be decrypted using
              /// DecryptStringAES().  The sharedSecret parameters must match.
              /// </summary>
              /// <param name="plainText">The text to encrypt.</param>
              /// <param name="sharedSecret">A password used to generate a key for
encryption.</param>
        private static string EncryptStringAes(string plainText, string sharedSecret)
        {
            if (string.IsNullOrEmpty(plainText))
                throw new ArgumentNullException(nameof(plainText));
            if (string.IsNullOrEmpty(sharedSecret))
                throw new ArgumentNullException(nameof(sharedSecret));

            string outStr;                       // Encrypted string to return
            RijndaelManaged aesAlg = null;              // RijndaelManaged object used to encrypt
the data.

            try
            {
                // generate the key from the shared secret and the salt
                var key = new Rfc2898DeriveBytes(sharedSecret, Salt);

                // Create a RijndaelManaged object
                aesAlg = new RijndaelManaged();
                aesAlg.Key = key.GetBytes(aesAlg.KeySize / 8);

                var encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

                using (var msEncrypt = new MemoryStream())
                {
                    // prepend the IV
                    msEncrypt.Write(BitConverter.GetBytes(aesAlg.IV.Length), 0, sizeof(int));
                    msEncrypt.Write(aesAlg.IV, 0, aesAlg.IV.Length);
                    using (var csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
                    {
                        using (var swEncrypt = new StreamWriter(csEncrypt))
                        {
                            //Write all data to the stream.
                            swEncrypt.Write(plainText);
                        }
                    }
                    outStr = Convert.ToBase64String(msEncrypt.ToArray());
                }
            }
            finally
            {
                aesAlg?.Clear();
            }

            // Return the encrypted bytes from the memory stream.
            return outStr;
```

```
        }

        /// <summary>
        /// Decrypt the given string.  Assumes the string was encrypted using
        /// EncryptStringAES(), using an identical sharedSecret.
        /// </summary>
        /// <param name="cipherText">The text to decrypt.</param>
        /// <param name="sharedSecret">A password used to generate a key for decryption.</param>
        private static string DecryptStringAes(string cipherText, string sharedSecret)
        {
            if (string.IsNullOrEmpty(cipherText))
                throw new ArgumentNullException(nameof(cipherText));
            if (string.IsNullOrEmpty(sharedSecret))
                throw new ArgumentNullException(nameof(sharedSecret));

            // Declare the RijndaelManaged object
            // used to decrypt the data.
            RijndaelManaged aesAlg = null;

            string plaintext;

            try
            {
                // generate the key from the shared secret and the salt
                var key = new Rfc2898DeriveBytes(sharedSecret, Salt);

                // Create the streams used for decryption.
                var bytes = Convert.FromBase64String(cipherText);
                using (var msDecrypt = new MemoryStream(bytes))
                {
                    // Create a RijndaelManaged object
                    // with the specified key and IV.
                    aesAlg = new RijndaelManaged();
                    aesAlg.Key = key.GetBytes(aesAlg.KeySize / 8);
                    // Get the initialization vector from the encrypted stream
                    aesAlg.IV = ReadByteArray(msDecrypt);
                    // Create a decrytor to perform the stream transform.
                    ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
                    using (var csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read))
                    {
                        using (var srDecrypt = new StreamReader(csDecrypt))

                            plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
            finally
            {
                // Clear the RijndaelManaged object.
                if (aesAlg != null)
                    aesAlg.Clear();
            }

            return plaintext;
        }

        private static byte[] ReadByteArray(Stream s)
        {
            var rawLength = new byte[sizeof(int)];
```

```
            if (s.Read(rawLength, 0, rawLength.Length) != rawLength.Length)
            {
                throw new SystemException("Stream did not contain properly formatted byte array");
            }

            var buffer = new byte[BitConverter.ToInt32(rawLength, 0)];
            if (s.Read(buffer, 0, buffer.Length) != buffer.Length)
            {
                throw new SystemException("Did not read byte array properly");
            }

            return buffer;
        }
    }
}
```

Fișier extinderea extensiei HTTP cu metode noi.

```
TW.Hejea.Web                                   TW.Hejea.Web.Extension.HttpContextExtensions
1   using System.Web;
2   using TW.Hejea.Domain.Entities.User;
3
4   namespace TW.Hejea.Web.Extension
5   {
6       public static class HttpContextExtensions
7       {
8           public static UserMinimal GetMySessionObject(this HttpContext current)
9           {
10              return (UserMinimal)current?.Session["__SessionObject"];
11          }
12
13          public static void SetMySessionObject(this HttpContext current, UserMinimal profile)
14          {
15              current.Session.Add("__SessionObject", profile);
16          }
17      }
18  }
```

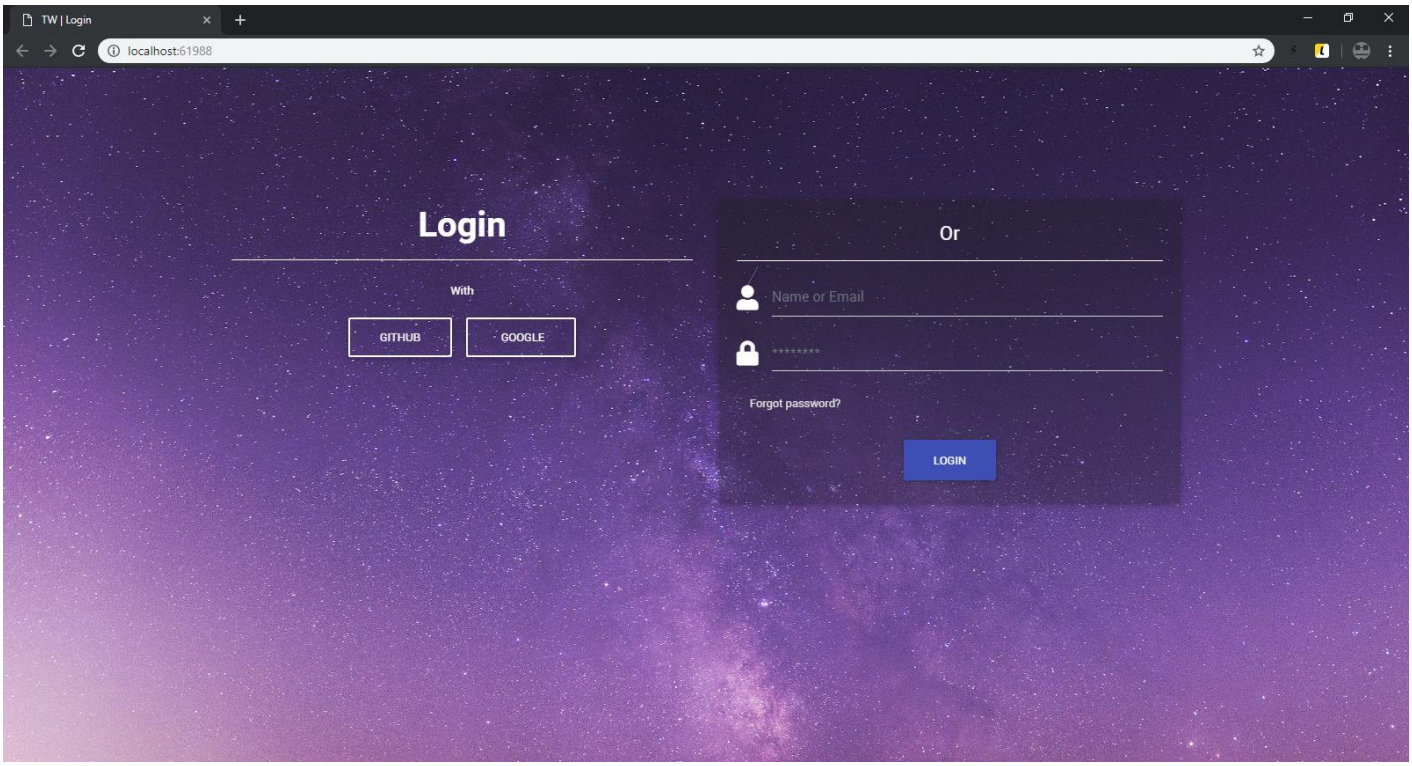
Fișierul AdminControler.cs .

TW.Hejea.Web

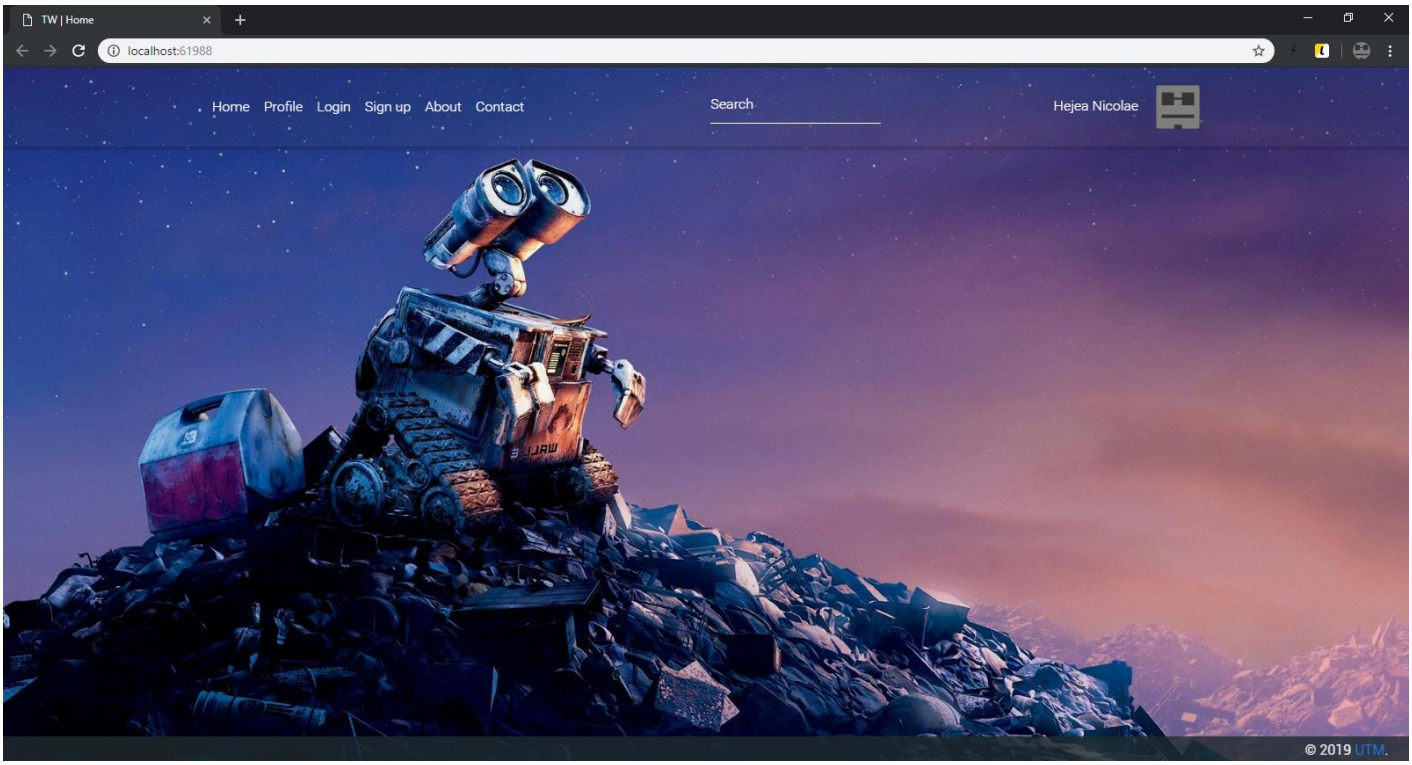
```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Web;
5   using System.Web.Mvc;
6   using TW.Hejea.Web.Controllers.Atribute;
7
8   namespace TW.Hejea.Web.Controllers
9   {
10      [CookieCheck(Order = 1)]
11      [AdminMod(Order = 1)]
12      public class AdminController : Controller
13      {
14          // GET: Admin
15          public ActionResult Index()
16          {
17              return View();
18          }
19      }
20  }
```

4. Rezultat

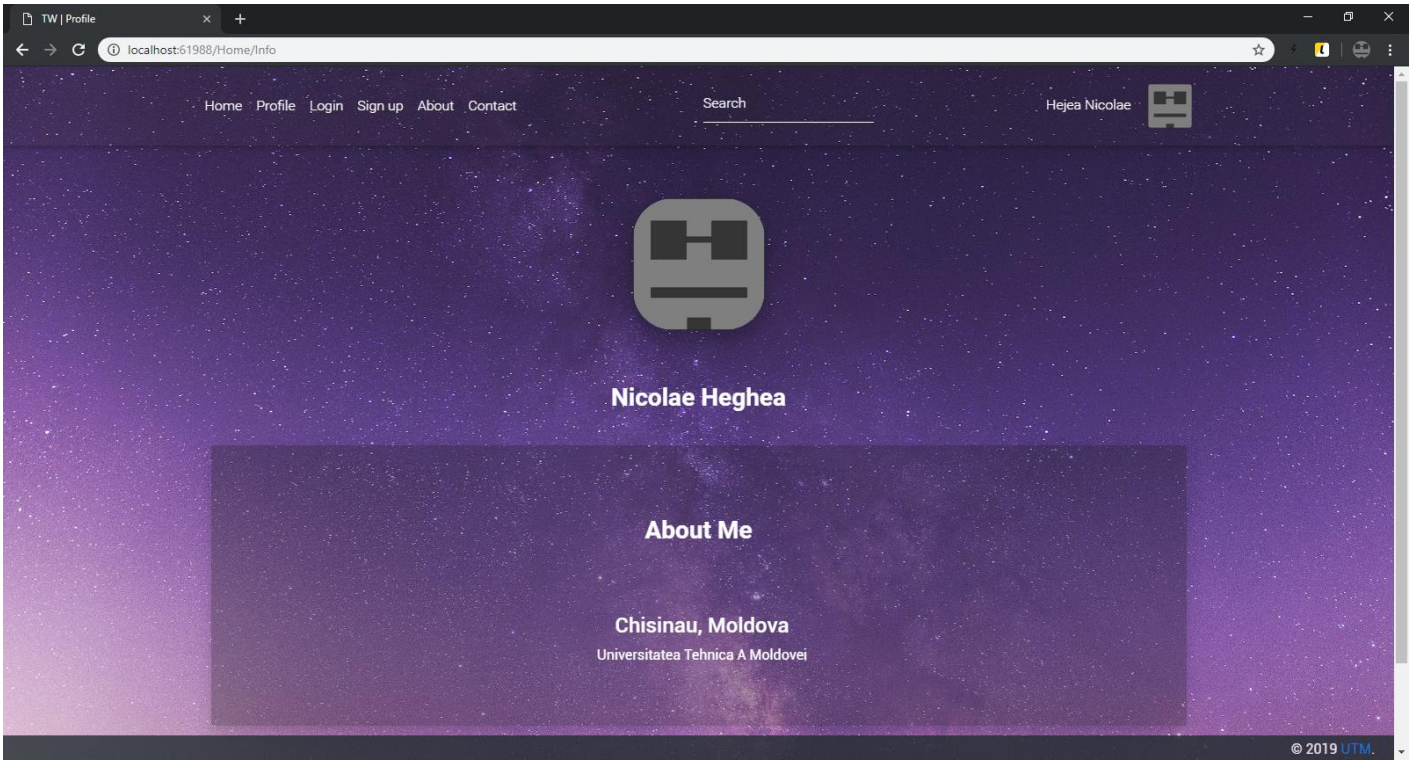
Pagina *login* :



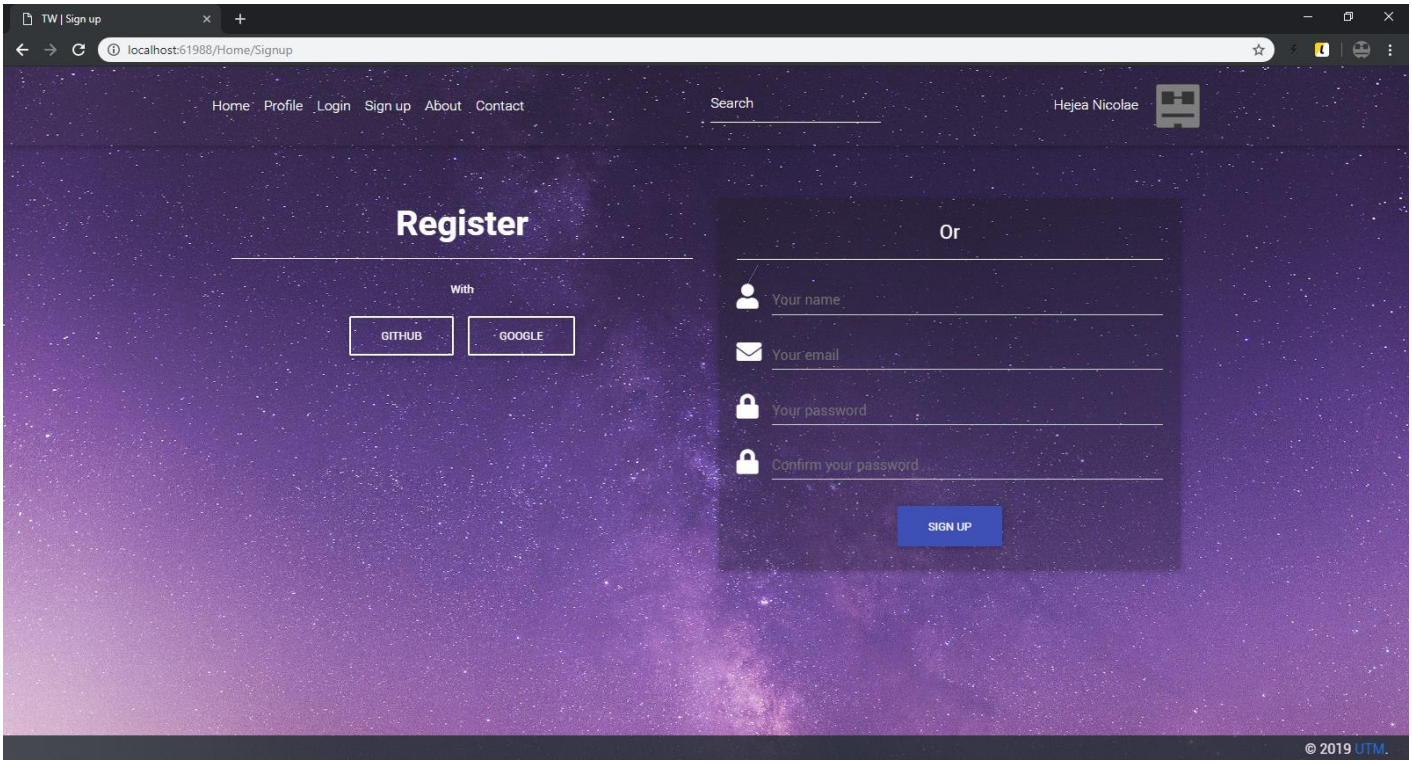
Pagina *home* :



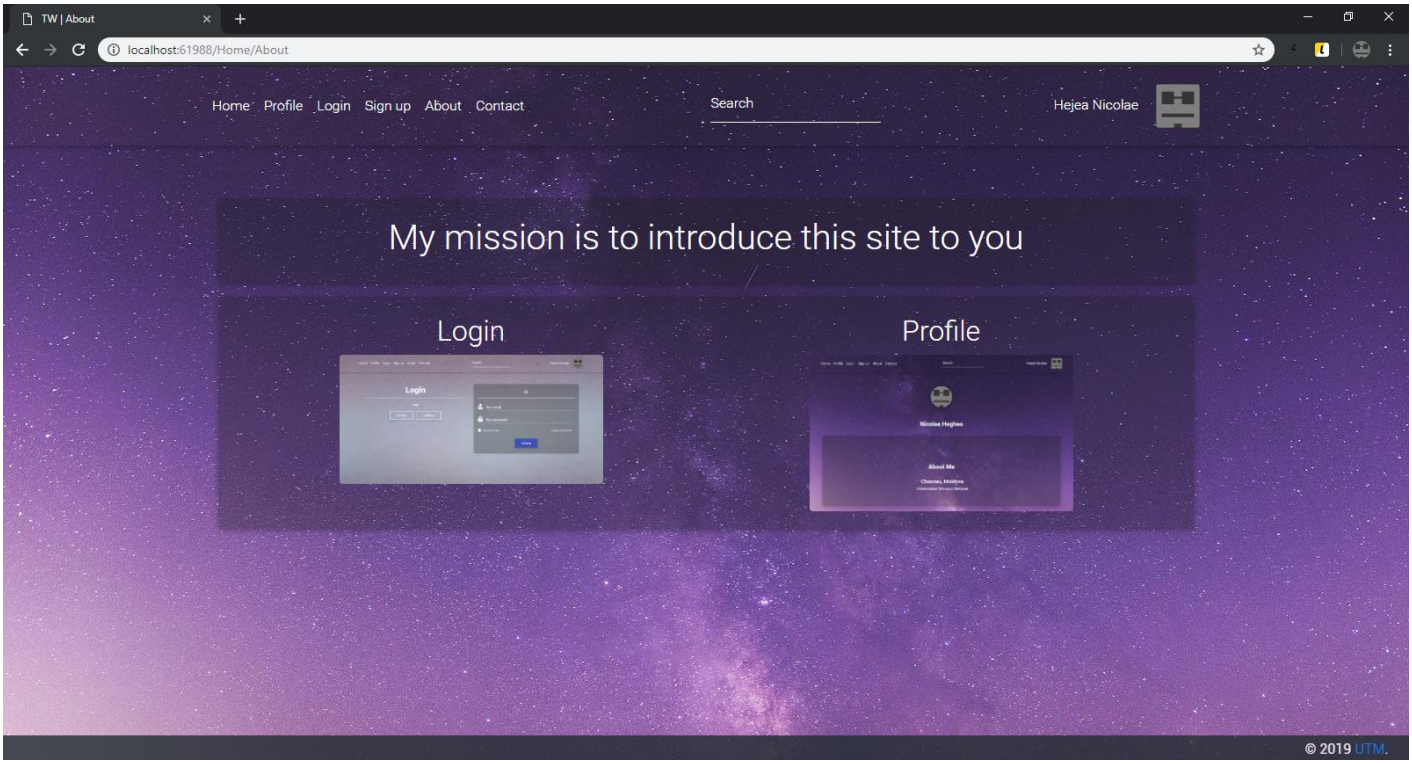
Pagina *profile* :



Pagina *sign up* :



Pagina *about* :



Pagina *contact* :