

Functionalitatea de schimbare a rolurilor de catre Administrator.
Autentificare cu Facebook.

Functionalitatea de schimbare a rolurilor de catre Administrator

Administratorul este cel care are dreptul sa modifice rolul unui utilizator. De asemenea, tot administratorul poate sterge utilizatori sau orice exista in aplicatie si nu respecta anumite reguli (de ex: comentarii nepotrivite, articole, poze, utilizatori, grupuri, subiecte de discutie, categorii, etc.)

In exemplul urmator ne vom axa doar pe functionalitatea de modificare a unui rol. Presupunem ca avem cele trei roluri pe care le-am utilizat si in exemplul din cursul anterior: **User**, **Editor** si **Administrator**. Orice utilizator in momentul in care se inregistreaza in aplicatie are rolul de **User**, Administratorul fiind cel care poate modifica un rol, daca este necesar.

In pagina **Index.cshtml** vom lista toti utilizatorii (doar utilizatorul cu rolul **Administrator** are acces la aceste informatii). Din aceasta pagina vom putea edita informatiile pentru fiecare utilizator, inclusiv rolul acestora. Pagina **Edit.cshtml** va contine formularul de editare.

Pagina **Index.cshtml**

Nume utilizator: user@test.com

✉ Email utilizator: user@test.com

Afisare utilizator



Nume utilizator: user2@test.com

✉ Email utilizator: user2@test.com

Afisare utilizator

Pagina **Edit.cshtml**

👤 Nume utilizator

user@test.com



✉ Email utilizator

user@test.com

📞 Numar telefon


☰ Selectati rolul


Editor





Modifica datele utilizatorului

Modificarea rolului are loc prin intermediul unui **DropDown**

 **Nume utilizator**

 **Email utilizator**

 **Numar telefon**


 **Selectati rolul**

Editor ▼

Administrator


Editor


User




Afisarea numarului de telefon se face doar daca acesta exista
(**PhoneNumber** din baza de date nu este un camp obligatoriu)


Nume utilizator: user@test.com


 Email utilizator:

 Telefon utilizator:



Nume utilizator: user2@test.com

 Email utilizator:



Primul pas îl constituie crearea unui nou Controller -> **UsersController**

Modelul care se ocupa de utilizatori (**ApplicationUser**) se genereaza automat atunci cand folosim sistemul de autentificare oferit de framework.

Pentru inceput se instantiaza in Controller clasa ApplicationDbContext:

```
[Authorize(Roles = "Administrator")]
public class UsersController : Controller
{
    private ApplicationDbContext db = ApplicationDbContext.Create();

    ...

}
```

In **UsersController**, in metoda **Index**, selectam toti utilizatorii astfel incat Administratorul sa poata edita orice utilizator, inclusiv rolul.

```
public ActionResult Index()
{
    var users = from user in db.Users
                orderby user.UserName
                select user;

    ViewBag.UsersList = users;
    return View();
}
```

In **View**, in **Index.cshtml**, afisam datele corespunzatoare utilizatorilor (doar o parte din informatii: nume, email, numar de telefon)

```

@{
    ViewBag.Title = "Afisare utilizatori";
}

<ol class="breadcrumb">
    <li><h3>@ViewBag.Title</h3></li>
</ol>

    @foreach (var user in ViewBag.UsersList)
    {
        <div class="panel-heading">Nume utilizator: @user.UserName</div>

        <div class="panel-body">

            <i class="glyphicon glyphicon-envelope"></i> Email utilizator:
            <span class="label label-default">@user.Email</span>

            <br /><br />

            @if (@user.PhoneNumber != null)
            {
                <i class="glyphicon glyphicon-phone"></i>@:Telefon
                utilizator:<span class="label label-default">@user.PhoneNumber</span>

            }

        </div>

        <div class="panel-footer">

            <a class="btn btn-sm btn-success" href="/Users/Edit/@user.Id">
            Editare utilizator</a>

        </div>

        <br /><br />
    }

```

In **UsersController**, in metoda **Edit** cu **HttpGet**, vom afisa formularul de editare, iar in metoda **Edit** cu **HttpPut** vom face update cu noile date adaugate.

Pentru editarea rolului unui utilizator vom utiliza un **Dropdown** in care vom incarca toate rolurile existente cu ajutorul unei metode. De asemenea, pentru editarea unui rol este necesara stergerea rolului existent si adaugarea noului rol.

```
public ActionResult Edit(string id)
{
    ApplicationUser user = db.Users.Find(id);
    user.AllRoles = GetAllRoles();
    var userRole = user.Roles.FirstOrDefault();
    ViewBag.userRole = userRole.RoleId;
    return View(user);
}

[NonAction]
public IEnumerable GetAllRoles()
{
    var selectList = new List();

    var roles = from role in db.Roles select role;
    foreach (var role in roles)
    {
        selectList.Add(new SelectListItem
        {
            Value = role.Id.ToString(),
            Text = role.Name.ToString()
        });
    }
    return selectList;
}

[HttpPut]
public ActionResult Edit(string id, ApplicationUser newData)
{
    ApplicationUser user = db.Users.Find(id);
    user.AllRoles = GetAllRoles();
    var userRole = user.Roles.FirstOrDefault();
    ViewBag.userRole = userRole.RoleId;
```



In IdentityModels.cs se
defineste proprietatea
AllRoles

```

try
{
    ApplicationDbContext context = new ApplicationDbContext();
    var roleManager = new RoleManager<IdentityRole>(new
RoleStore<IdentityRole>(context));
    var UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(context));

    if (TryUpdateModel(user))
    {
        user.UserName = newData.UserName;
        user.Email = newData.Email;
        user.PhoneNumber = newData.PhoneNumber;

        var roles = from role in db.Roles select role;
        foreach (var role in roles)
        {
            UserManager.RemoveFromRole(id, role.Name);
        }

        var selectedRole =
db.Roles.Find(HttpContext.Request.Params.Get("newRole"));
        UserManager.AddToRole(id, selectedRole.Name);

        db.SaveChanges();
    }
    return RedirectToAction("Index");
}
catch (Exception e)
{
    Response.Write(e.Message);
    return View(user);
}
}

```

In **IdentityModels.cs**:

```

public IEnumerable<SelectListItem> AllRoles { get; set; }

```

In View-ul Edit.cshtml:

```
@model Laborator7.Models.ApplicationUser

@{
    ViewBag.Title = "Editare rol utilizator";
}

<ol class="breadcrumb">
    <li><h3>@ViewBag.Title</h3></li>
</ol>

<form method="post" action="/Users/Edit/@Model.Id">

    @Html.HttpMethodOverride(HttpVerbs.Put)

    @Html.HiddenFor(m => m.Id)

    <br />

    <i class="glyphicon glyphicon-user"></i>
    @Html.Label("UserName", "Nume utilizator", new { @class = "" })

    <br />

    @Html.EditorFor(m => m.UserName)

    <br /><br />

    <i class="glyphicon glyphicon-envelope"></i>
    @Html.Label("Email", "Email utilizator", new { @class = "" })

    <br />

    @Html.EditorFor(m => m.Email)

    <br /><br />

    <i class="glyphicon glyphicon-phone"></i>
    @Html.Label("PhoneNumber", "Numar telefon")

    <br />

    @Html.EditorFor(m => m.PhoneNumber)

    <br /><br />

    <i class="glyphicon glyphicon-th-list"></i>
    <label>Selectati rolul</label>
    @Html.DropDownList("newRole", new SelectList(Model.AllRoles, "Value",
    "Text", (string)ViewBag.userRole), null, new { @class = "form-control" })

    <br />
```



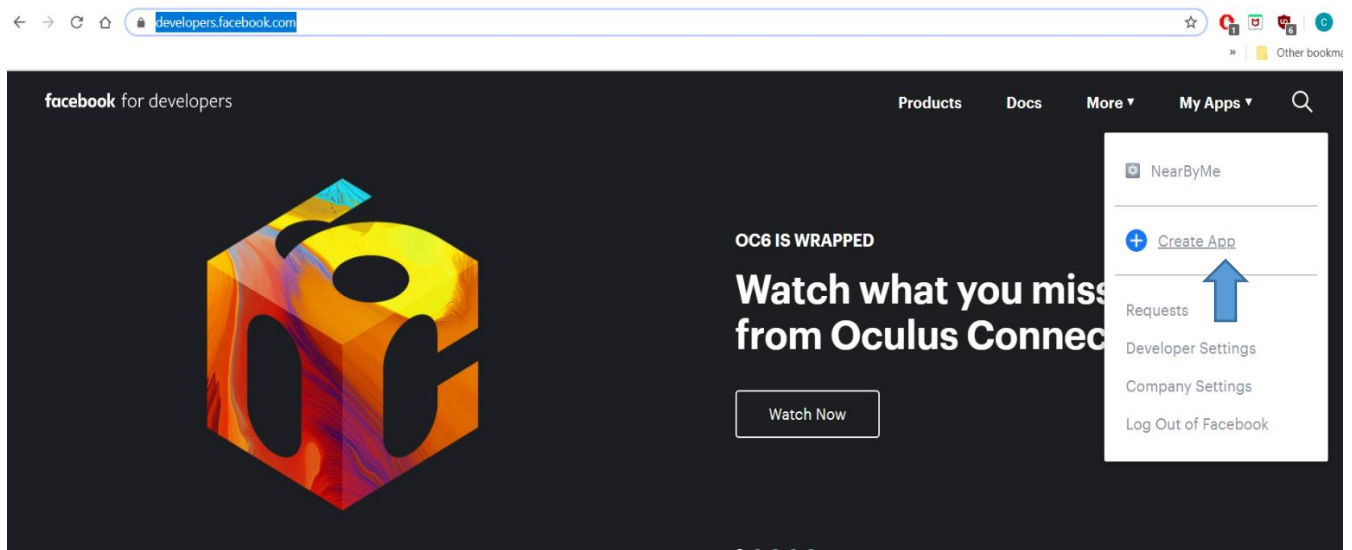
```
<button class="btn btn-sm btn-success" type="submit">Modifica datele  
utilizatorului</button>  
  
</form>
```

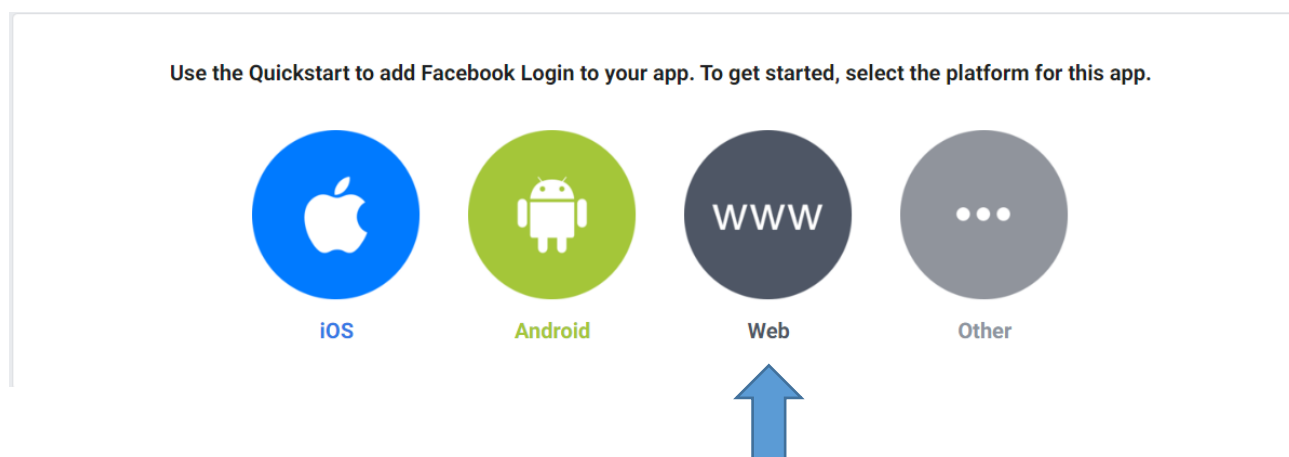
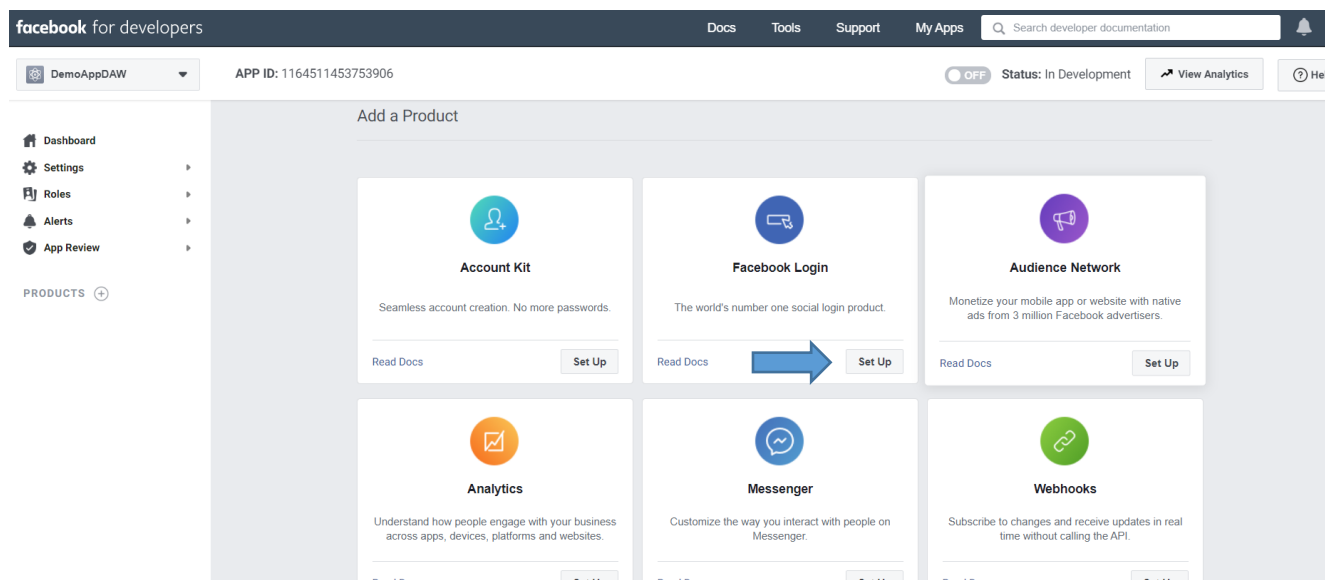
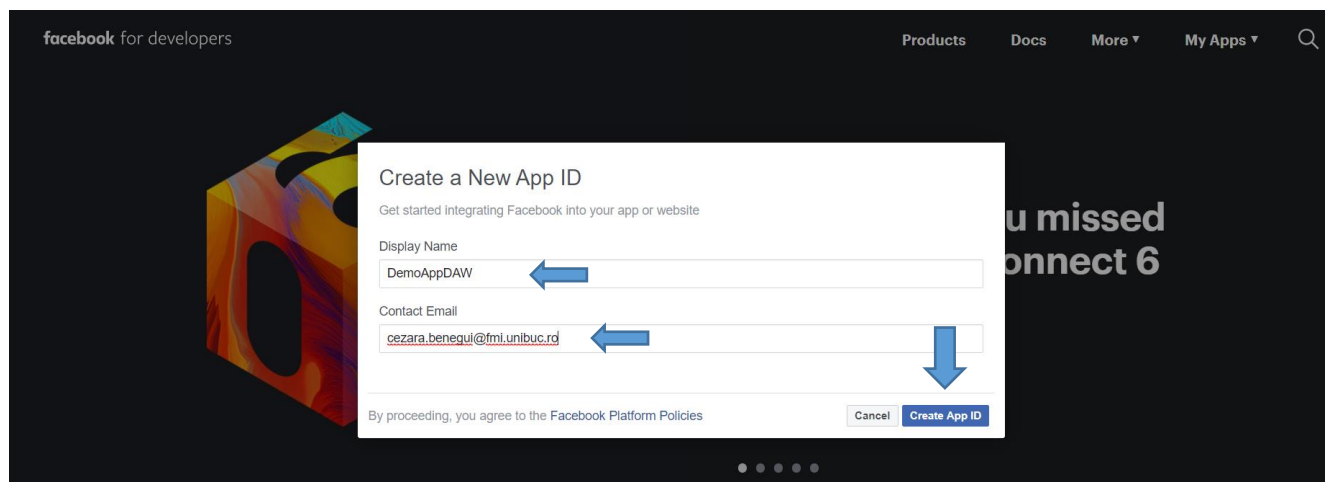
De asemenea, pentru `<form>` se poate utiliza si helper-ul `Html.BeginForm`

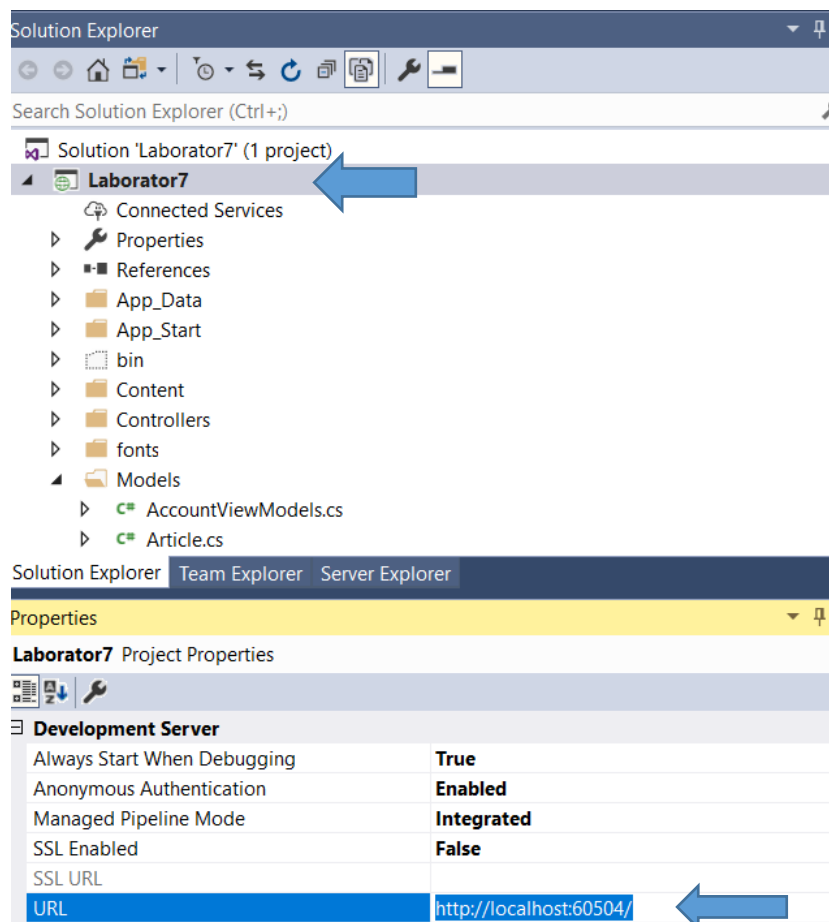
```
@using (Html.BeginForm(actionName: "Edit", controllerName: "Users", routeValues: new  
{ id = @Model.Id })))
```

Autentificare cu Facebook

Pentru a integra autentificarea cu Facebook, se foloseste <https://developers.facebook.com/> unde ne vom crea o aplicatie pe care o vom utiliza pentru procesul **oAuth** (Open Authentication).





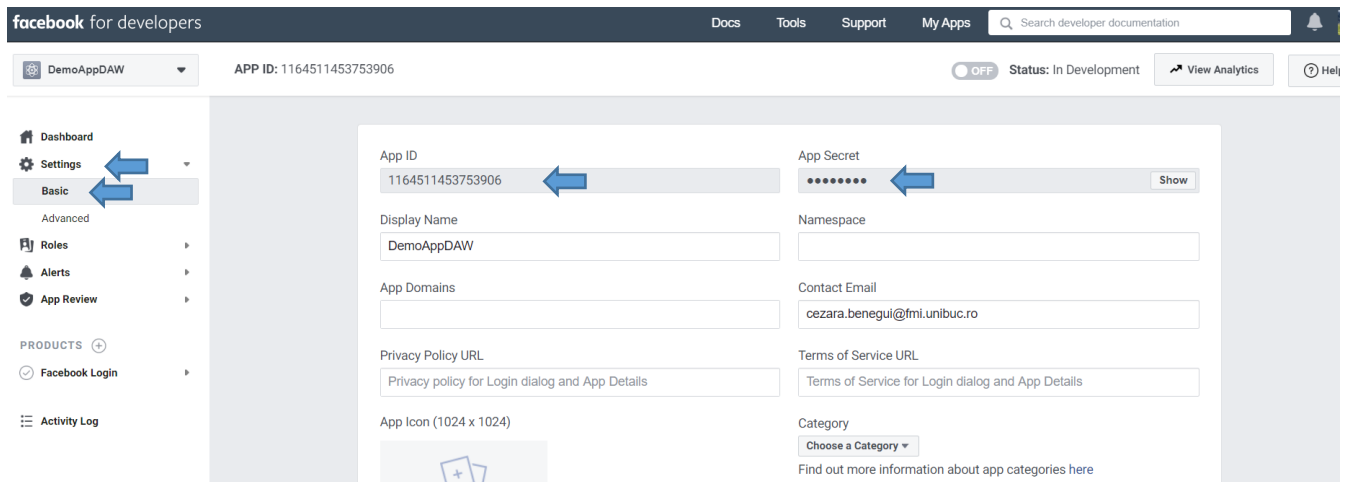


1. Tell Us about Your Website

Tell us what the URL of your site is.

Site URL

2. Set Up the Facebook SDK for Javascript



Pentru a configura sistemul de autentificare folosind Facebook, in continuare, se implementeaza urmatoorii pasi:

1. In folderul App_Start -> Startup.Auth.cs -> se configureaza

app.UseFacebookAuthentication, folosind **appId** si **appSecret** generate in aplicatia de developer

```
app.UseFacebookAuthentication(
    appId: "",
    appSecret: "");
```

2. In folderul Models -> AccountViewModels.cs -> in clasa `ExternalLoginConfirmationViewModel`, se modifica astfel incat pe langa campul existent "Email" sa existe si proprietatea "UserName":

```
public class ExternalLoginConfirmationViewModel
{
    [Required]
    [Display(Name = "UserName")]
    public string UserName { get; set; }

    [Required]
    [Display(Name = "Email")]
    public string Email { get; set; }
}
```

3. In folderul Controllers -> AccountController -> ExternalLoginCallback se modifica astfel:

```
switch (result)
{
    case SignInStatus.Success:
        return RedirectToLocal(returnUrl);
    case SignInStatus.LockedOut:
        return View("Lockout");
    case SignInStatus.RequiresVerification:
        return RedirectToAction("SendCode", new { ReturnUrl = returnUrl,
RememberMe = false });
    case SignInStatus.Failure:
    default:
        // If the user does not have an account, then prompt the user to
create an account
        ViewBag.ReturnUrl = returnUrl;
        ViewBag.LoginProvider = loginInfo.Login.LoginProvider;

        return View("ExternalLoginConfirmation",
            new ExternalLoginConfirmationViewModel { UserName =
loginInfo.DefaultUserName,
                Email = loginInfo.Email });
}
```

4. In folderul Controllers -> AccountController -> ExternalLoginConfirmation -> se modifica astfel incat in momentul autentificarii cu Facebook, pentru noul utilizator autentificat, sa se completeze in mod corect campurile **UserName** si **Email**.

```
if (ModelState.IsValid)
{
    // Get the information about the user from the external login provider
    var info = await AuthenticationManager.GetExternalLoginInfoAsync();
    if (info == null)
    {
        return View("ExternalLoginFailure");
    }

    var user = new ApplicationUser { UserName = model.UserName, Email =
model.Email };

    var result = await UserManager.CreateAsync(user);
    if (result.Succeeded)
    {
        result = await UserManager.AddLoginAsync(user.Id, info.Login);
        if (result.Succeeded)
        {

```

```

        await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);

        UserManager.AddToRole(user.Id, "User");

        return RedirectToLocal(returnUrl);
    }
}
AddErrors(result);
}

```

5. In folderul Views -> Account -> ExternalLoginConfirmation.cshtml -> se adauga in View-ul pentru Login si campul UserName astfel:

```

<div class="form-group">
    @Html.LabelFor(m => m.UserName, new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.TextBoxFor(m => m.UserName, new { @class = "form-control" })
        @Html.ValidationMessageFor(m => m.UserName, "", new { @class = "text-
danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger"
    })
    </div>
</div>

```

Dupa parcurgerea pasilor ne vom putea autentifica folosind Facebook

