

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 **Admin**. 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 **Admin** 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: 

Afisare utilizator

Nume utilizator: user2@test.com

 Email utilizator: 

Afisare 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 = "Admin")]
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>
        <a class="btn btn-sm btn-success" href="/Users/Show/@user.Id"> Vizualizare
utilizator</a>

    </div>

    <br /><br />
}

```

In **UserController**, in metoda **Show**, se afiseaza cate un utilizator in functie de id-ul pe care il are.

OBS!

In momentul in care se afiseaza rolul unui utilizator trebuie sa tinem cont de modul in care Identity stocheaza rolurile. Intre utilizatori si roluri este o relatie many-to-many, utilizatorul avand astfel unul sau mai multe roluri.

Avand in vedere ca in aplicatia noastra folosim un singur rol pentru un utilizator, se poate obtine rolul acestuia folosind `user.Roles.FirstOrDefault()`.

Pentru a determina executarea query-ului este necesara alocarea valorii rolului curent ca string -> `string` `currentRole = user.Roles.FirstOrDefault().RoleId` deoarece Linq nu poate executa un query care contine variabile complexe (care nu sunt de tip primitiv).

```
public ActionResult Show(string id)
{
    ApplicationUser user = db.Users.Find(id);

    ViewBag.utilizatorCurent = User.Identity.GetUserId();

    string currentRole = user.Roles.FirstOrDefault().RoleId;

    var userRoleName = (from role in db.Roles
                        where role.Id == currentRole
                        select role.Name).First();

    ViewBag.roleName = userRoleName;

    return View(user);
}
```

In **UserController**, 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<SelectListItem> GetAllRoles()
{
    var selectList = new List<SelectListItem>();

    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;

    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))

```



In IdentityModels.cs se
defineste proprietatea
AllRoles

```

    {
        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);
    newData.Id = id;
    return View(newData);
}
}

```

In IdentityModels.cs:

```

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

```


In View-ul Edit.cshtml:

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

    <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",
    ViewBag.userRole), null, new { @class = "form-control" })

    <br />

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

Metoda Delete:

```
[HttpDelete]
public ActionResult Delete(string id)
{
    ApplicationDbContext context = new ApplicationDbContext();

    var UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(context));

    var user = UserManager.Users.FirstOrDefault(u => u.Id == id);

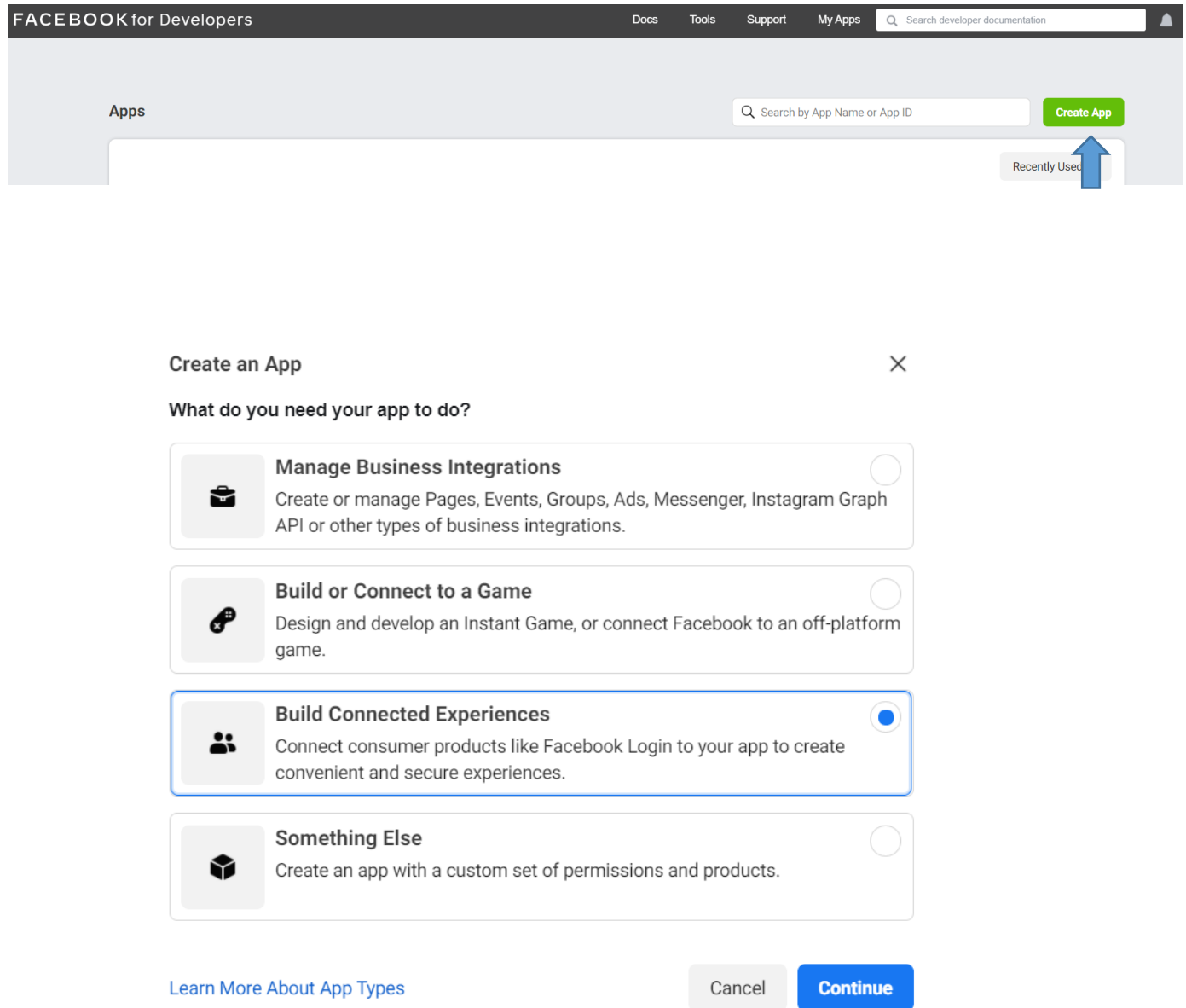
    var articles = db.Articles.Where(a => a.UserId == id);
    foreach (var article in articles)
    {
        db.Articles.Remove(article);
    }

    var comments = db.Comments.Where(comm => comm.UserId == id);
    foreach (var comment in comments)
    {
        db.Comments.Remove(comment);
    }

    db.SaveChanges();
    UserManager.Delete(user);
    return RedirectToAction("Index");
}
```

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).



FACEBOOK for Developers Docs Tools Support My Apps Search developer documentation

Apps Search by App Name or App ID Create App Recently Used

Create an App X

What do you need your app to do?

- ☐ **Manage Business Integrations**
Create or manage Pages, Events, Groups, Ads, Messenger, Instagram Graph API or other types of business integrations.
- ☐ **Build or Connect to a Game**
Design and develop an Instant Game, or connect Facebook to an off-platform game.
- ☒ **Build Connected Experiences**
Connect consumer products like Facebook Login to your app to create convenient and secure experiences.
- ☐ **Something Else**
Create an app with a custom set of permissions and products.

[Learn More About App Types](#) Cancel Continue

Create an App



App Display Name

This is the app name associated with your app ID.

App Contact Email

This email address is used to contact you about potential policy violations, app restrictions or steps to recover the app if it's been deleted or compromised.

Do you have a Business Manager account? · Optional

In order to access certain aspects of the Facebook platform, apps may need to be connected to a verified Business Manager account. If you haven't yet set up an account, you can create one now or later in the process.

By proceeding, you agree to the [Facebook Platform Terms](#) and [Developer Policies](#)

[< Back](#)[Create App](#)

The screenshot shows the Facebook for Developers dashboard. At the top, there's a navigation bar with 'facebook for developers', 'Docs', 'Tools', 'Support', 'My Apps', and a search bar. Below this, a header bar shows 'DemoAppDAW', 'APP ID: 1164511453753906', 'Status: In Development', and 'View Analytics'. The main content area is titled 'Add a Product' and displays six product cards in a 2x3 grid:

- Account Kit**: Seamless account creation. No more passwords. [Read Docs] [Set Up]
- Facebook Login**: The world's number one social login product. [Read Docs] [Set Up] (with a blue arrow pointing to it)
- Audience Network**: Monetize your mobile app or website with native ads from 3 million Facebook advertisers. [Read Docs] [Set Up]
- Analytics**: Understand how people engage with your business across apps, devices, platforms and websites. [Read Docs] [Set Up]
- Messenger**: Customize the way you interact with people on Messenger. [Read Docs] [Set Up]
- Webhooks**: Subscribe to changes and receive updates in real time without calling the API. [Read Docs] [Set Up]

A left sidebar contains navigation links: Dashboard, Settings, Roles, Alerts, App Review, and a 'PRODUCTS' section with a plus icon.

Use the Quickstart to add Facebook Login to your app. To get started, select the platform for this app.



iOS



Android



Web



Other



Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'Laborator' (1 project)

Laborator

- Connected Services
- Properties
- References
- App_Data
- App_Start
- bin
- Content
- Controllers
- fonts
- Models
 - AccountViewModels.cs
 - Article.cs

Solution Explorer | Team Explorer | Server Explorer

Properties

Laborator7 Project Properties

Development Server

Always Start When Debugging	True
Anonymous Authentication	Enabled
Managed Pipeline Mode	Integrated
SSL Enabled	False
SSL URL	
URL	http://localhost:60504/

1. Tell Us about Your Website

Tell us what the URL of your site is.

Site URL

http://localhost:60504

Save

Continue

2. Set Up the Facebook SDK for Javascript

facebook for developers
Docs
Tools
Support
My Apps
Search developer documentation

DemoAppDAW
APP ID: 1164511453753906
OFF
Status: In Development
View Analytics
Help

Dashboard
Settings
Basic
Advanced
Roles
Alerts
App Review
PRODUCTS
Facebook Login
Activity Log

App ID
1164511453753906

App Secret
.....
Show

Display Name
DemoAppDAW

Namespace

App Domains

Contact Email
cezara.benegui@fmi.unibuc.ro

Privacy Policy URL
Privacy policy for Login dialog and App Details

Terms of Service URL
Terms of Service for Login dialog and App Details

App Icon (1024 x 1024)

Category
Choose a Category
Find out more information about app categories here

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

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

            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

