Functionalitatea de schimbare a rolurilor de catre Administrator. JSON.

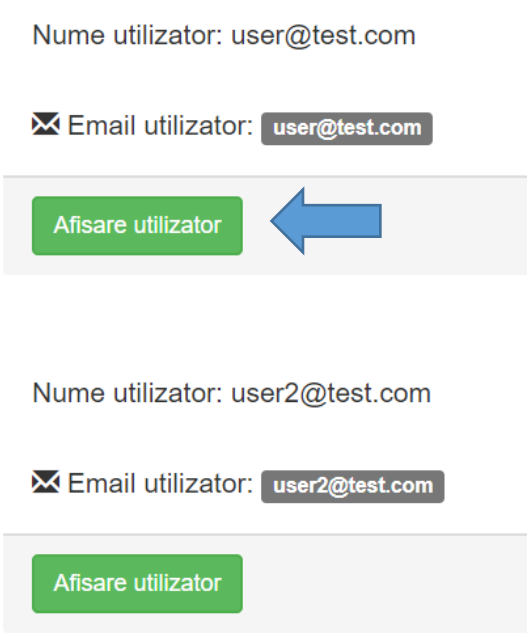
Functionalitatea de schimbare a rolurilor de catre Administrator

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.)

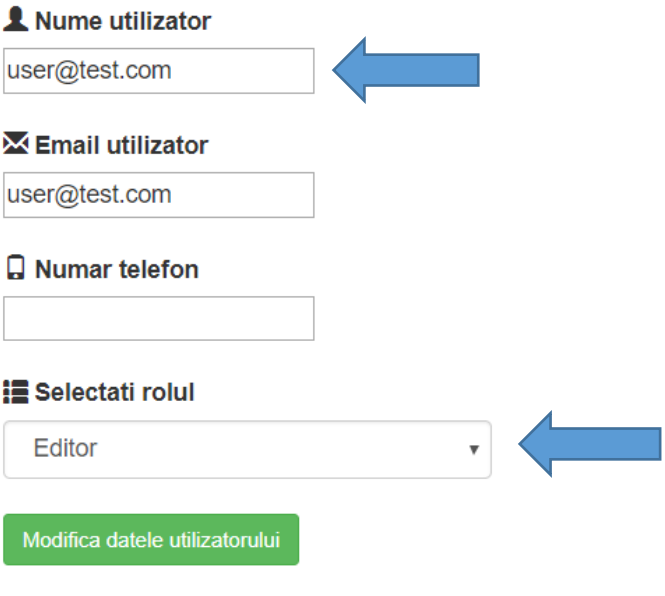
In exemplul urmator ne vom axa doar pe functionalitatea de modificare a unui rol. Presupunem ca avem cele trei roluri pe care le-am utilizat si in exemplul din cursul anterior: User, Editor si Administrator. Orice utilizator in momentul in care se inregistreaza in aplicatie are rolul de User, Administratorul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

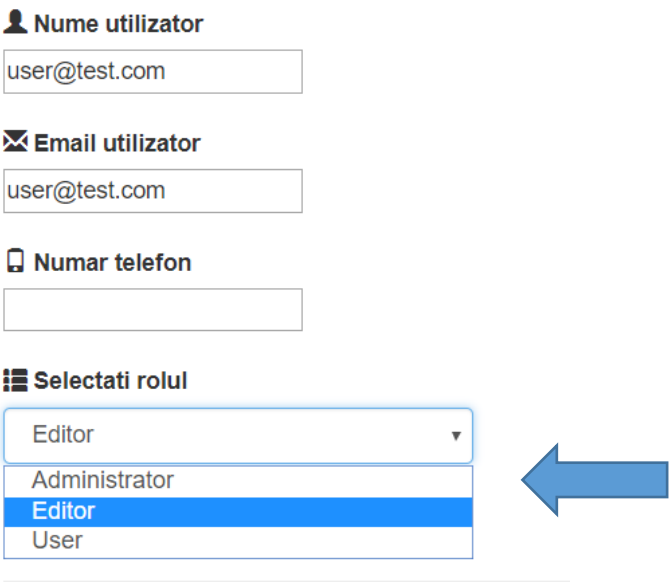
Pagina **Index.cshtml**



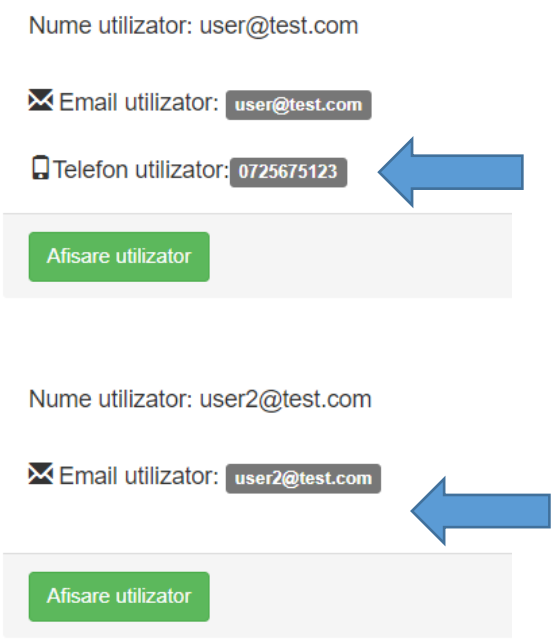
Pagina **Edit.cshtml**



Modificarea rolului are loc prin intermediul unui DropDown



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



Primul pas il constituie crearea unui nou Controller -> **UsersController** (Controller-ul se denumeste folosind pluralul, iar Modelul folosind singularul).

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

Pentru inceput se instatiaza 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)

```
@model Lab10IdentityNew.Models.ApplicationUser

@{

    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">
Afisare utilizator</a>

        </div>

        <br /><br />
     }
```

In **UsersController**, in metoda **Edit**, 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;
```

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 Lab10IdentityNew.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>
```

JSON (JavaScript Object Notation)

JSON-ul este utilizat pentru stocarea si schimbul de date intre browser si server. Orice obiect JavaScript poate fi convertit intr-un JSON si trimis catre server. JSON-ul este un text si poate fi usor trimis catre server, cat si primit de la server. **Este utilizat in orice limbaj de programare**, ceea ce duce la o comunicare eficienta intre mai multe limbaje de programare fara sa fie necesar sa stim toate aceste limbaje. Un alt avantaj al JSON-ului este structura sa simpla, ceea ce il face usor de citit si inteles.

JSON-ul este construit din doua elemente:

- O colectie de tipul cheie-valoare (in limbajele de programare aceasta colectie poate fi un obiect, o inregistrare – record, un dictionar, un struct, hash, array asociativ, etc)

- O lista ordonata de valori (in majoritatea limbajelor de programare acest lucru este reprezentat de un array, vector, lista sau chiar o secventa)

Atunci cand cream un Controller intr-un proiect MVC tipul default returnat este ActionResult. Pentru a returna JSON se utilizeaza JsonResult, care sugereaza faptul ca va fi returnat un obiect de tip JSON si nu HTML.

Exemplu:

```
public ActionResult Index()
        {
             var articles =
       db.Articles.Include("Category").Include("User").Select(row => new
             {
                 row.Title,
                 row.Id,
                 row.Date,
                 category_name = row.Category.CategoryName,
                 row.Content,
                 author_name = row.User.UserName,
                 author_email = row.User.Email
             });
             if (TempData.ContainsKey("message"))
             {
                 ViewBag.message = TempData["message"].ToString();
             }


             return Json( articles.ToList(), JsonRequestBehavior.AllowGet);
         }
```

Afisare JSON:

```
[
    {
        "Title": "Articol 1",
        "Id": 6,
        "Date": "/Date(1543864008000)/",
        "category_name": "Craciun",
        "Content": "Continut Articol 3",
        "author_name": "admin@admin.com",
        "author_email": "admin@admin.com"
    },
    {
        "Title": " Articol 2",
        "Id": 13,
        "Date": "/Date(1543865746000)/",
        "category_name": "Stiinta",
        "Content": "Continut articol",
        "author_name": "user@test.com",
        "author_email": "user@test.com"
    },
```

```
    {
        "Title": "Articol 3",
        "Id": 14,
        "Date": "/Date(1544109699000)/",
        "category_name": "Craciun",
        "Content": "Continut articol",
        "author_name": "user2@test.com",
        "author_email": "user2@test.com"
    }
]
```