C# Web – MVC

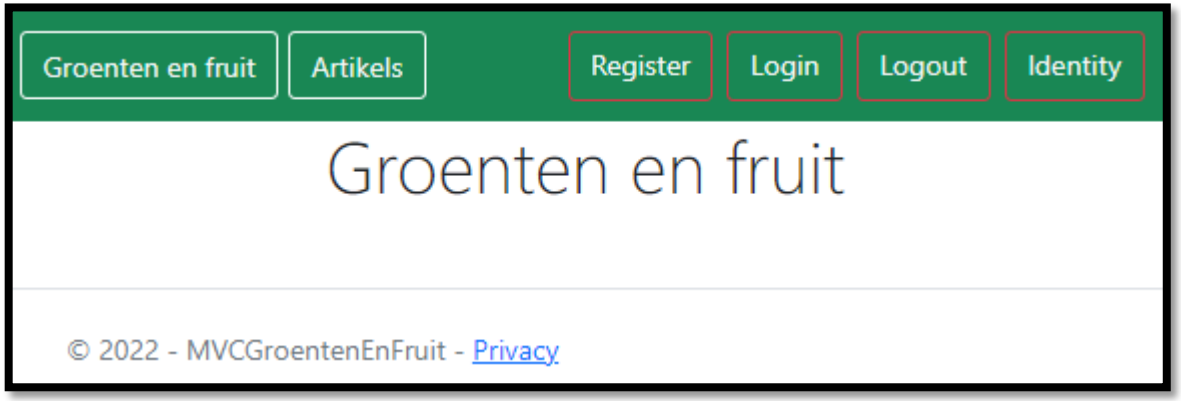
PRO/PRW - C# Web 1

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Doel

- ASP.Net Core toepassing
  - Identity Framework
    - Identity User
    - Identity Role



IdentityUser - Register

Teams – 2_MVCGroentenEnFruit
- Nuget Package Manager - Console
  - Update-Database

SeedData

- Artikels
- IdentityRole – AspNetRoles
- IdentityUser – AspNetUsers
- AspNetUserRoles

Artikels
- Ijsbergsla
- Roma tomaat

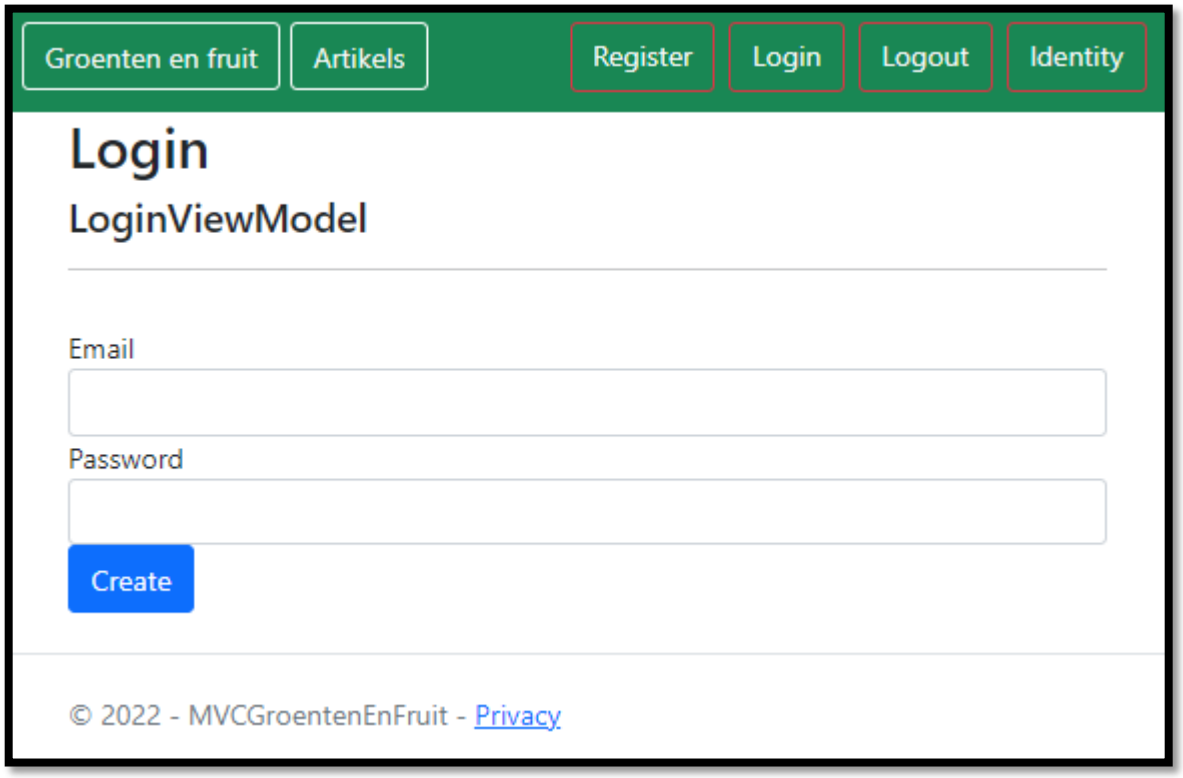
Views/Shared/_Layout

Roles
- Aankoper
- Verkoper

Register users

- aankoper@pxl.be
  - Aankoper00!
    - Role: **Aankoper**
- verkoper@pxl.be
  - Verkoper00!
    - Role: **Verkoper**

IdentityUser - Login



IdentityUser – Login Dependency Injection

```
public class AccountController : Controller
{
    AppDbContext _context;
    UserManager<IdentityUser> _userManager;
    SignInManager<IdentityUser> _signInManager;
    public AccountController(AppDbContext context,
        UserManager<IdentityUser> userManager,
        SignInManager<IdentityUser> signInManager)
    {
        _context = context;
        _userManager = userManager;
        _signInManager = signInManager;
    }
```

IdentityUser – Login
AccountController - Login

```
[HttpPost]
public async Task<IActionResult> LoginAsync(LoginViewModel login)
{
    var signInResult = await _signInManager.PasswordSignInAsync(
        login.Email, login.Password, false, false);

    if (signInResult.Succeeded)
    {
        return RedirectToAction("Index", "Home");
    }
    ModelState.AddModelError("", "Probleem met inloggen");
    return View();
}
```

IdentityUser – ArtikelsController

Beveilig de controller

```
namespace MVCGroentenEnFruit.Controllers
{
    [Authorize]
    public class ArtikelsController : Controller
    {


Men wordt automatisch door verwezen naar login pagina

Test de Login met de gegevens uit de SeedData class
```

IdentityUser – Login
AccountController - Login

```
[HttpPost]
public async Task<IActionResult> LoginAsync(LoginViewModel login)
{
    var signInResult = await _signInManager.PasswordSignInAsync(
        login.Email, login.Password, false, false);

    if (signInResult.Succeeded)
    {
        return RedirectToAction("Index", "Home");
    }
    ModelState.AddModelError("", "Probleem met inloggen");
    return View();
}
```

IdentityUser – Login
AccountController - Login

```
[HttpPost]
public async Task<IActionResult> LoginAsync(LoginViewModel login)
{
    var identityUser = await _userManager.FindByEmailAsync(login.Email);
    if (identityUser != null)
    {
        var signInResult =
           await _signInManager.PasswordSignInAsync(
                identityUser.UserName, login.Password, false, false);
        if (signInResult.Succeeded)
        {
            return RedirectToAction("Index", "Home");
        }
    }
    ModelState.AddModelError("", "Probleem met inloggen");
    return View();
}
```

IdentityUser – Login
AccountController - Logout

IdentityUser – Login AccountController - Logout

```
#region logout
public async Task<IActionResult> LogoutAsync()
{
    await _signInManager.SignOutAsync();
    return View("Login");
}
#endregion
```

IdentityUser - Register

Register users

- aankoper@pxl.be
  - Aankoper00!
    - Role: **Aankoper**
- verkoper@pxl.be
  - Verkoper00!
    - Role: **Verkoper**

AccountController – Identity Dependency Injection

```
public class AccountController : Controller
{
    AppDbContext _context;
    UserManager<IdentityUser> _userManager;
    SignInManager<IdentityUser> _signInManager;
    RoleManager<IdentityRole> _roleManager;
    public AccountController(
        AppDbContext context,
        UserManager<IdentityUser> userManager,
        SignInManager<IdentityUser> signInManager,
        RoleManager<IdentityRole> roleManager)
    {
        _context = context;
        _userManager = userManager;
        _signInManager = signInManager;
        _roleManager = roleManager;
    }
```

IdentityUser – Login
AccountController - Register

```
public async Task<IActionResult> RegisterAsync(RegisterViewModel registerViewModel)
{
    ...
    if (identityResult.Succeeded)
    {
        var identityRole =
            await _roleManager.FindByIdAsync(registerViewModel.RoleId);
        var roleResult =
            await _userManager.AddToRoleAsync(identityUser, identityRole.Name);
        if (roleResult.Succeeded)
            return View("Login");
```

IdentityUser – IdentityRole

Models/ViewModels
- Add class – IdentityViewModel

```
namespace MVCGroentenEnFruit.Models.ViewModels
{
    public class IdentityViewModel
    {
        public IEnumerable<IdentityUser>? Users { get; set; }
        public IEnumerable<IdentityRole>? Roles { get; set; }
    }
}
```

IdentityUser – IdentityRole
Views/_ViewImports

```
@using MVCGroentenEnFruit
@using MVCGroentenEnFruit.Models
@using MVCGroentenEnFruit.Models.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

IdentityUser – IdentityRole Views/Account/Identity

- Add empty Razor View – Account/Identity

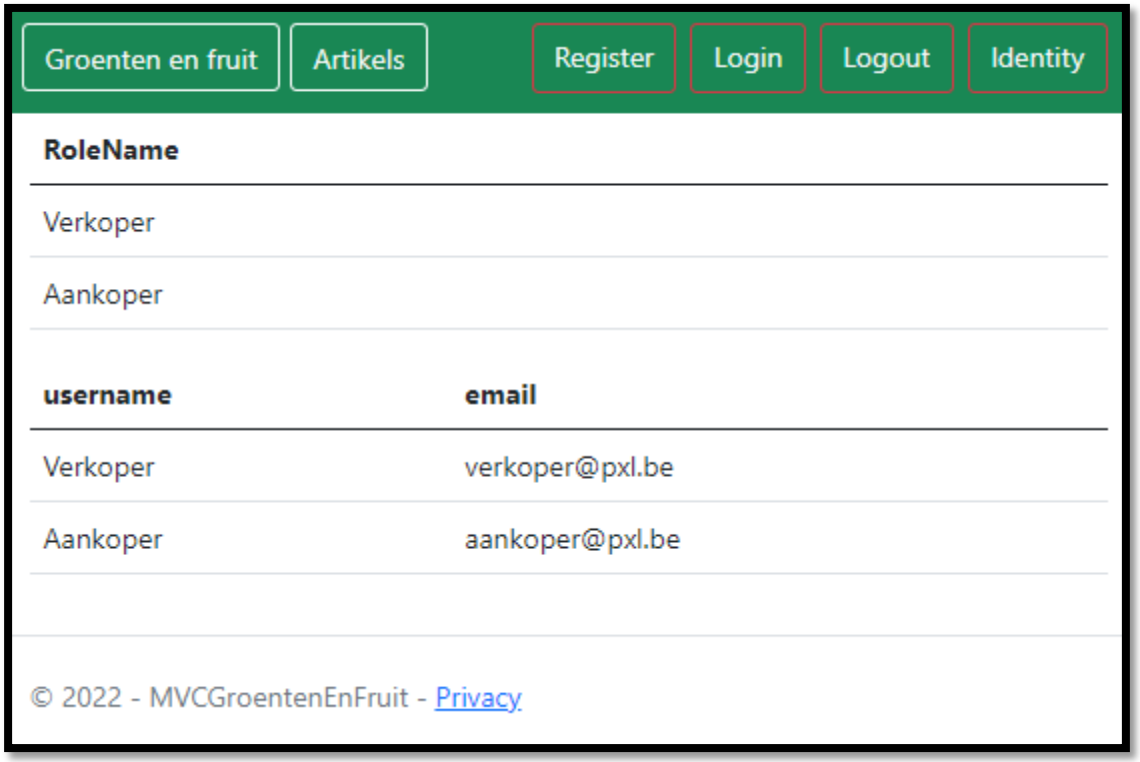
```
@model IdentityViewModel

@if(Model.Roles != null)
{
    <table class="table"><thead>
        <tr>
            <th>RoleName</th>
        </tr></thead>
        @foreach(var role in @Model.Roles)
        {
            <tr>
                <td>@role.Name</td>
            </tr>
        }
    </table>
}
```

AccountController – Identity

```
#region Identity
[HttpGet]
public IActionResult Identity()
{
    var identityViewModel = new IdentityViewModel();
    identityViewModel.Roles = _roleManager.Roles;
    return View(identityViewModel);
}
#endregion
```

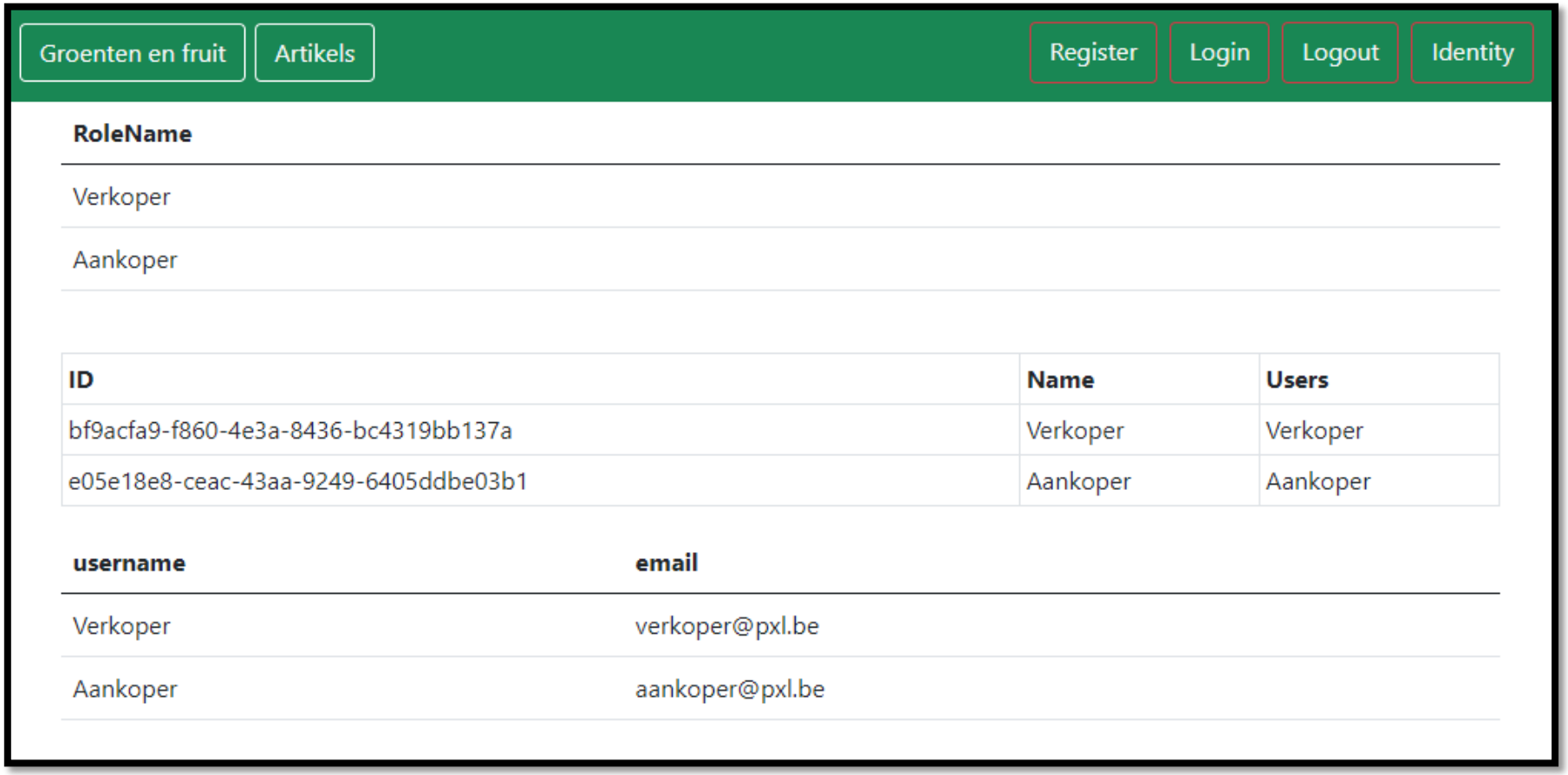
AccountController – Identity Oefening - IdentityUser



AccountController – Identity Oefening - IdentityUser

```
#region Identity
        [HttpGet]
        public IActionResult Identity()
        {
            var identityViewModel = new IdentityViewModel();
            identityViewModel.Roles = _roleManager.Roles;
            identityViewModel.Users = _userManager.Users;
            return View(identityViewModel);
        }
        #endregion
```

AccountController – Identity
TagHelper – IdentityUser - IdentiyRole



AccountController – Identity TagHelper – IdentityUser – IdentiyRole Dependency Injection

```
public class RoleUsersTagHelper : TagHelper
{
    private UserManager<IdentityUser>? _userManager;
    private RoleManager<IdentityRole>? _roleManager;
    public RoleUsersTagHelper(
        UserManager<IdentityUser> userManager,
        RoleManager<IdentityRole> roleManager)
    {
        _userManager = userManager;
        _roleManager = roleManager;
    }
    public override async Task ProcessAsync(
                TagHelperContext context, TagHelperOutput output)
    {
        …
    }
}
```

AccountController – Identity TagHelper – IdentityUser – IdentiyRole Attributes

```
namespace MVCGroentenEnFruit.TagHelpers
{
    [HtmlTargetElement("td", Attributes = "role")]
    public class RoleUsersTagHelper : TagHelper
    {
        ...
        [HtmlAttributeName("role")]
        public string Role { get; set; }
         public override async Task ProcessAsync(TagHelperContext context,
                TagHelperOutput output)
```

AccountController – Identity TagHelper – IdentityUser – IdentiyRole Process

```
public override async Task ProcessAsync(TagHelperContext context,
        TagHelperOutput output)
{
    List<string> names = new List<string>();
    IdentityRole role = await _roleManager.FindByIdAsync(Role);
    if (role != null)
    {
        foreach (var user in _userManager.Users)
        {
            if (user != null && await _userManager.IsInRoleAsync(user, role.Name))
                names.Add(user.UserName);
        }
    }
    output.Content.SetContent(names.Count == 0 ? "No Users" :
        string.Join(", ", names));
}
```

AccountController – Identity
TagHelper – IdentityUser – IdentiyRole
Views/Account/Identity

```
@if(Model.Roles != null)
{
    <table class="table">
        …
    </table>
    <br />
    <table class="table table-sm table-bordered table-bordered">
        <tr><th>ID</th><th>Name</th><th>Users</th></tr>
        @foreach(var role in Model.Roles)
        {
            <tr>
                <td>@role.Id</td>
                <td>@role.Name</td>
                <td role="@role.Id"></td>
            </tr>
        }
    </table>
```

AccountController – Identity TagHelper – IdentityUser – IdentiyRole Views/_ViewImports

```
@using MVCGroentenEnFruit
@using MVCGroentenEnFruit.Models
@using MVCGroentenEnFruit.Models.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@addTagHelper *, MVCGroentenEnFruit
```

IdentityUser – Models/Data/AankoopOrder

We gaan de persoon die is ingelogd opslaan in de tabel AankoopOrder als men een record wil aanmaken.
Dus we moeten controle inbouwen dat gebruiker is ingelogd en we voegen ook nog een 2de controle toe dat enkel een persoon met een Aankoper rol deze acties mag uitvoeren.

```
public class AankoopOrder
{
        public int AankoopOrderId { get; set; }
        public int ArtikelId { get; set; }
        public int Hoeveelheid { get; set; }
        public string? IdentityUserId { get; set; }
        public Artikel? Artikel { get; set; }
        public IdentityUser? IdentityUser { get; set; }

}
```

Nuget Package Manager Console

```
Add-migration aankoop
Update-database
```