

Model ASP.NET MVC Core 3.1 Web App - Deel 5

1 Kortingen toevoegen

1.1 Aanmaken model voor Coupon

Onder de folder models, voeg een nieuwe class toe voor Coupon

```
public class Coupon
{
    [Key]
    public int Id { get; set; }

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

    [Required]
    [Display(Name = "Type Korting")]
    public string CouponType { get; set; }

    public enum ECouponType { Percent = 0, Euro = 1 }

    [Required]
    [Display(Name = "Korting")]
    public double Discount { get; set; }

    [Required]
    [Display(Name = "Minimum bedrag")]
    public double MinimumAmount { get; set; }
    [Display(Name = "Afbeelding")]
    public byte[] Picture { get; set; }

    [Display(Name = "Is Actief")]
    public bool IsActive { get; set; }
}
```

1.2 Aanpassen ApplicationDbContext

Voeg aan de class ApplicationDbContext DbSet public property toe voor Coupon

```
public DbSet<Coupon> Coupon { get; set; }
```

1.3 Database migraties toevoegen voor de nieuwe Coupon tabel en database updaten

Indien je applicatie goed werkt, kan je nu de tabellen in de database creëren.

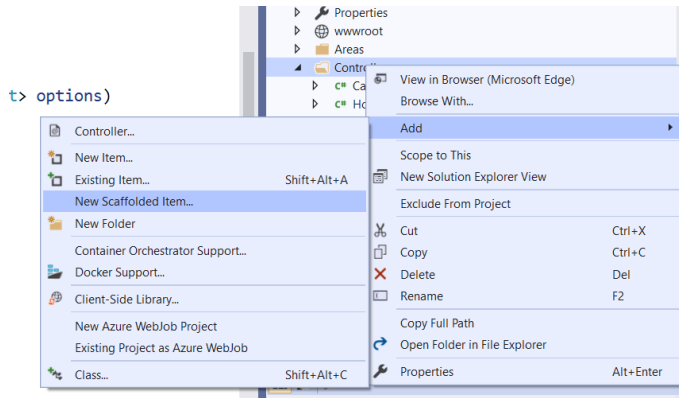
We doen dit via de **NuGet Package Manager Console** door eerst data-migraties op te zetten en daarna update-database commando

```
PM> add-migration AddedCouponToDb
```

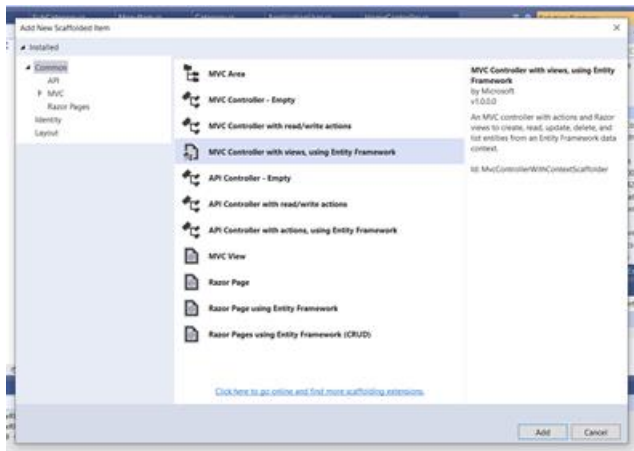
```
PM> update-database
```

1.4 Aanmaken Controller en Views voor CRUD operaties voor Coupon

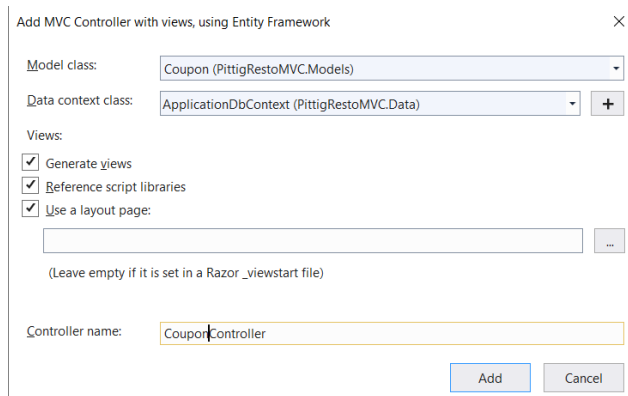
Rechtsklik op de folder Controller en kies Add/New Scaffolded Item...



En daarna Mvc Controller with views, using Entity Framework



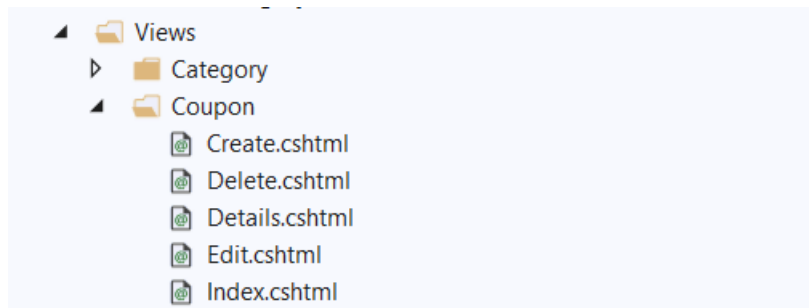
Kies in het volgende popup-venster bv als **model class Coupon**



Data class ApplicationDbContext

Controller name CouponController

Zorg dat de **Generate views aangevinkt** staat, deze zal dan onder de Views/Coupon folder Razor views genereren voor een Coupon (Create.cshtml, Delete.cshtml, Edit.cshtml en Index.cshtml, Details.cshtml) die een standaard UI voorzien waarin een gebruiker CRUD operaties kan verrichten voor Coupons.



Onder de folder controller is er eveneens een CouponController.cs toegevoegd

1.5 Aanpassen van Create en Edit action methods in de CouponController

Om een bestand in binair formaat te kunnen opslaan in de database (de coupon-image), moeten de Create en Edit action methods worden aangepast.

Open de CouponController en pas de methoden Create en Edit aan:

```
public IActionResult Create()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(Coupon coupons)
{
    if (ModelState.IsValid)
```

```

        {
            var files = HttpContext.Request.Form.Files;
            if (files.Count > 0)
            {
                byte[] p1 = null;
                using (var fs1 = files[0].OpenReadStream())
                {
                    using (var ms1 = new MemoryStream())
                    {
                        fs1.CopyTo(ms1);
                        p1 = ms1.ToArray();
                    }
                }
                coupons.Picture = p1;
            }
            _context.Coupon.Add(coupons);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(coupons);
    }
}

//GET Edit Coupon
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var coupon = await _context.Coupon.SingleOrDefaultAsync(m => m.Id == id);
    if (coupon == null)
    {
        return NotFound();
    }
    return View(coupon);
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(Coupon coupons)
{
    if (coupons.Id == 0)
    {
        return NotFound();
    }

    var couponFromDb = await _context.Coupon.Where(c => c.Id == coupons.Id).FirstOrDefaultAsync();

    if (ModelState.IsValid)
    {
        var files = HttpContext.Request.Form.Files;
        if (files.Count > 0)
        {
            byte[] p1 = null;
            using (var fs1 = files[0].OpenReadStream())
            {

```

```

        using (var ms1 = new MemoryStream())
        {
            fs1.CopyTo(ms1);
            p1 = ms1.ToArray();
        }
        couponFromDb.Picture = p1;
    }
    couponFromDb.MinimumAmount = coupons.MinimumAmount;
    couponFromDb.Name = coupons.Name;
    couponFromDb.Discount = coupons.Discount;
    couponFromDb.CouponType = coupons.CouponType;
    couponFromDb.IsActive = coupons.IsActive;

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
return View(coupons);
}

```

1.6 Aanpassen van Coupon Razor views Create en Edit

Om een image te kunnen uploaden in de Razor views, moeten er eveneens aanpassingen in deze views worden verricht:

1. Open Areas/Admin/Views/Coupon/Create.cshtml en pas de Razor code aan:

```

@model PittigRestoMVC.Models.Coupon
@{
    ViewData["Title"] = "Nieuw";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<br />
<h2 class="text-info">Aanmaken Korting</h2>
<br />

<form method="post" enctype="multipart/form-data">
    <div class="border backgroundWhite">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Name" class="col-form-label"></label>
            </div>
            <div class="col-5">
                <input asp-for="Name" class="form-control" />
            </div>
            <span asp-validation-for="Name" class="text-danger"></span>
        </div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Picture" class="col-form-label"></label>
            </div>
            <div class="col-5">
                <input type="file" id="projectImage" name="files" multiple
class="form-control" />
            </div>
        </div>
        <div class="form-group row">

```

```

        <div class="col-2">
            <label asp-for="CouponType" class="col-form-label"></label>
        </div>
        <div class="col-5">
            <select asp-for="CouponType" asp-
items="Html.GetEnumSelectList<Coupon.ECouponType>()" class="form-
control"></select>
        </div>
    </div>
    <div class="form-group row">
        <div class="col-2">
            <label asp-for="Discount" class="col-form-label"></label>
        </div>
        <div class="col-5">
            <input asp-for="Discount" class="form-control" />
        </div>
        <span asp-validation-for="Discount" class="text-danger"></span>
    </div>
    <div class="form-group row">
        <div class="col-2">
            <label asp-for="MinimumAmount" class="col-form-label"></label>
        </div>
        <div class="col-5">
            <input asp-for="MinimumAmount" class="form-control" />
        </div>
        <span asp-validation-for="MinimumAmount" class="text-danger"></span>
    </div>
    <div class="form-group row">
        <div class="col-2">
            <label class="form-check-label" asp-for="IsActive"></label>
        </div>
        <div class="col-5">
            <input class="form-check-input mx-1" type="checkbox" asp-
for="IsActive">
        </div>
    </div>
    <div class="form-group row">
        <div class="col-5 offset-2">
            <partial name="_CreateAndBackToListButton" />
        </div>
    </div>
</div>
</form>

@section Scripts{
    @{ await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

2. Open Areas/Admin/Views/Coupon/Edit. cshtml en pas de Razor code aan:

```

@model PittigRestoMVC.Models.Coupon
@{
    ViewData["Title"] = "Wijzig";
}

<br />
<h2 class="text-info">Wijzig korting</h2>
<br />

```

```

<form asp-action="Edit" method="post" enctype="multipart/form-data">
  <div class="row borderWidth">

    <br />
    <div class="col-12 border">
      @{
        var base64 = Convert.ToBase64String(Model.Picture);
        var imgsrc = string.Format("data:image/jpg;base64,{0}", base64);
      }
      
    </div>
    <div class="col-12 pt-4">
      <input hidden asp-for="Id" />
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-group row">
        <div class="col-2">
          <label asp-for="Name" class="custom-label"></label>
        </div>
        <div class="col-5">
          <input asp-for="Name" class="form-control" />
        </div>
        <span asp-validation-for="Name" class="text-danger"></span>
      </div>
      <div class="form-group row">
        <div class="col-2">
          <label asp-for="Picture" class="custom-label"></label>
        </div>
        <div class="col-5">
          <input type="file" id="projectImage" name="files" multiple
class="form-control" />
        </div>
      </div>
      <div class="form-group row">
        <div class="col-2">
          <label asp-for="CouponType" class="custom-label"></label>
        </div>
        <div class="col-5">
          <select asp-for="CouponType" asp-
items="Html.GetEnumSelectList<Coupon.ECouponType>()" class="form-
control"></select>
        </div>
      </div>
      <div class="form-group row">
        <div class="col-2">
          <label asp-for="Discount" class="custom-label"></label>
        </div>
        <div class="col-5">
          <input asp-for="Discount" class="form-control" />
        </div>
        <span asp-validation-for="Discount" class="text-danger"></span>
      </div>
      <div class="form-group row">
        <div class="col-2">
          <label asp-for="MinimumAmount" class="custom-label"></label>
        </div>
        <div class="col-5">
          <input asp-for="MinimumAmount" class="form-control" />
        </div>
      </div>
    </div>
  </div>

```

```

        <span asp-validation-for="MinimumAmount" class="text-
danger"></span>
    </div>
    <div class="form-group row">
        <div class="col-2">
            <label asp-for="IsActive" class="custom-label"></label>
        </div>
        <div class="col-5">
            <input type="checkbox" asp-for="IsActive" class="form-control"
/>
        </div>
    </div>
    <br />
    <div class="form-group row">
        <div class="col-5 offset-2">
            <partial name="_EditAndBackToListButton" model="Model.Id" />
        </div>
    </div>
</div>
</form>

```

3. Open als laatste Areas/Admin/Coupons/Index.cshtml en pas eveneens de Razor view met lijst van kortingen aan:

```

@model IEnumerable<Coupon>

@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<br />
<br />
<div class="border backgroundWhite">
    <div class="row">
        <div class="col-6">
            <h2 class="text-info"> Lijst van kortingen</h2>
        </div>
        <div class="col-6 text-right">
            <p>
                <a asp-action="Create" class="btn btn-info"><i class="fas fa-
plus"></i> &nbsp; Nieuw </a>
            </p>
        </div>
    </div>
    <br />
    <div>
        @if (Model.Count() > 0)
        {
            <table class="table table-striped border">
                <tr class="table-secondary">
                    <th>
                        @Html.DisplayNameFor(m => m.Name)
                    </th>
                    <th>
                        @Html.DisplayNameFor(m => m.Discount)

```



```

        </th>
        <th>
            @Html.DisplayNameFor(m => m.MinimumAmount)
        </th>
        <th>
            @Html.DisplayNameFor(m => m.IsActive)
        </th>
        <th></th>
        <th></th>
    </tr>
    @foreach (var item in Model)
    {
    <tr>
        <td>
            @Html.DisplayFor(m => item.Name)
        </td>
        <td>
            @Html.DisplayFor(m => item.Discount)
        </td>
        <td>
            @Html.DisplayFor(m => item.MinimumAmount)
        </td>
        <td>
            @Html.DisplayFor(m => item.IsActive)
        </td>
        <td>
            <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
            <a asp-action="Details" asp-route-id="@item.Id">Details</a> |
            <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
        </td>
    </tr>
    }
    </table>
    }
    else
    {
        <p>Er bestaan nog geen kortingen...</p>
    }
    </div>
</div>

```

Start de applicatie en test nu de applicatie met deze url:

<https://localhost:xxxxx/Admin/Coupon/index>

2 Tonen van Categorieën,coupons en menuitems op de homepagina

We moeten op de homepagina gegevens uit verschillende entiteiten en tonen en dus uit tabellen van de database halen.

Een goede manier om dit te doen is om een ViewModel class te maken die de gegevens uit verschillende tabellen zal bijhouden. In de HomeController vullen we de properties van een object van deze ViewModel in en halen de gegevens op vanuit. Ten slotte geeft de Controller dit ViewModel object door aan de View.

We maken een nieuwe folder Models/ViewModels om onze ViewModel class te plaatsen.

Maak een folder ViewModels onder de folder Models

Maak onder ViewModels folder een nieuwe class IndexViewModel

```
public class IndexViewModel
{
    public IEnumerable<MenuItem> MenuItem { get; set; }
    public IEnumerable<Category> Category { get; set; }
    public IEnumerable<Coupon> Coupon { get; set; }
}
```

Open de Controller Areas\Customer\Controllers\HomeController.cs

Verwijder eerst deze code:

```
private readonly ILogger<HomeController> _logger;
public HomeController(ILogger<HomeController> logger)
{
    _logger = logger;
}
```

We hebben **gegevens nodig uit de ApplicationDbContext**. Deze is reeds in de ConfigureServices in startup.cs geregistreerd als service in de DI Container van ASP.Net Core

We kunnen deze service **nu injecteren in onze HomeController** en we gaan dit doen via **Constructor-injection**:

Maak hiervoor een private field aan waarin we een ApplicationDbContext object kunnen bijhouden.

```
private readonly ApplicationDbContext _db;
```

En een constructor die deze als parameter aanneemt aan de private field initialiseert:

```
public HomeController(ApplicationDbContext db)
{
    _db = db;
}
```

Pas nu de Index() methode aan. Deze wordt uitgevoerd bij een HttpGet Request naar de Home-pagina

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PittigRestoMVC.Data;
using PittigRestoMVC.Models;
using PittigRestoMVC.Models.ViewModels;

namespace Pittig.Controllers
{
    [Area("Customer")]
    public class HomeController : Controller
    {
        private readonly ApplicationDbContext _db;

        public HomeController(ApplicationDbContext db)
        {
            _db = db;
        }

        public async Task<IActionResult> Index()
        {
            IndexViewModel IndexVM = new IndexViewModel()
            {
                MenuItem = await _db.MenuItem.Include(m =>
                    m.Category).Include(m => m.SubCategory).ToListAsync(),
                Category = await _db.Category.ToListAsync(),
                Coupon = await _db.Coupon.Where(c => c.IsActive == true).ToListAsync()
            };

            return View(IndexVM);
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

Opmerking: Aangezien we nu asynchrone calls maken in de Index methode, moet deze ook asynchroon worden

```
public async Task<IActionResult> Index()
```

Hier wordt een object van IndexViewModel aangemaakt en de Menus, Category en Coupons worden door asynchrone aanroepen uit de database gehaald en ingevuld in de public properties van het object.

Ten slotte wordt het object meegegeven aan de View (standaard heeft de view dezelfde naam als de action method, dus in dit geval zal dit dus de view Index.cshtml zijn onder de folder Area/Customr/Home/

Opmerking: De .Include() in LINQ to Entities bij bv_db.MenuItem, Category en SubCategory

Is nodig omdat **EF Core Lazy Loading** gebruikt en de gegevens uit gerefereerde tabellen niet automatisch zal opladen. Dus indien je eveneens gegevens wil ophalen van de tabel die gelinkt is met een FK relatie, moet je dit expliciet vermelden in de LINQ Query. Indien je dit niet doet, kan je null waarden terugkrijgen voor de gelinkte data.

3 Pas de View van de homepagina aan

1. We maken eerst een partial Razor view “_ThumbnailAreaPartial” onder de folder Views/Shared

Deze krijgt een Model van type IEnumerable<MenuItem> binnen en deze wordt afgebeeld

```
@model IEnumerable<MenuItem>
@*
    For more information on enabling MVC for empty projects, visit
    http://go.microsoft.com/fwlink/?LinkID=397860
*@
@if (Model.Count() > 0)
{
    <div class="col-12 post @Model.FirstOrDefault().Category.Name.Replace(
    ",string.Empty) menu-restaurant">
        <div class="row">
            <h3 class="text-success"> @Model.FirstOrDefault().Category.Name </h3>
        </div>

        @foreach(var item in Model)
        {
            <div class="border border-info rounded col-12" style="margin-
            bottom:10px; margin-top:10px; padding:10px">
                <div class="row">
                    <div class="col-md-3 col-sm-12">
                        
                    </div>
                    <div class="col-md-9 col-sm-12">
                        <div class="row pr-3">
                            <div class="col-8">
```

```

<label class="text-primary" style="font-
size:21px;color:maroon">@item.Name</label>
@if (item.Spicyness == "1")
{
    <img srcset="/images/mild.png" title="Mild" />
}
@if (item.Spicyness == "2")
{
    <img srcset="/images/spicy.png" title="Pittig"
/>
}
@if (item.Spicyness == "3")
{
    <img srcset="/images/veryspicy.png"
title="Zeer Pittig" />
}
</div>
<div class="col-4 text-right" style="color:maroon">
    <h4>@item.Price €</h4>
</div>
</div>

<div class="row col-12 text-justify d-none d-md-block">
    <p>@Html.Raw(item.Description)</p>
</div>
<div class="col-md-3 col-sm-12 offset-md-9 text-center">
    <a asp-action="Details" class="btn btn-success form-
control" asp-route-id="@item.Id">Details</a>
</div>
</div>
</div>
</div>
}
<div class="p-4"></div>
</div>
}

```

2. Open de Index.cshtml onder Area/Customer/Home/ Deze zal een 'carousel' tonen van de images van actieve kortingen, een lijst van categorieën tonen en we refereren de partial view die we in de vorige stap hebben aangemaakt die voor een categorie de lijst van menuitems toont.

De Index View ontvangt een object van Type IndexViewModel van de HomeController's Index() action method. We specificëren bovenaan deze pagina het type van het object dat binnenkomt dat ons Model zal zijn in deze view:

```

@model PittigRestoMVC.Models.ViewModels.IndexViewModel

<br />

@if (Model.Coupon.ToList().Count > 0)
{
    <div class="border">
        <div class="carousel" data-ride="carousel" data-interval="2500">
            @for (int i = 0; i < Model.Coupon.Count(); i++)

```

```

        {
            if (i == 0)
            {
                <div class="carousel-item active">
                    @{
                        var base64 =
Convert.ToBase64String(Model.Coupon.ToList()[i].Picture);
                        var imgsrc =
string.Format("data:image/jpg;base64,{0}", base64);
                    }

                    
                </div>
            }
            else
            {
                <div class="carousel-item">
                    @{
                        var base64 =
Convert.ToBase64String(Model.Coupon.ToList()[i].Picture);
                        var imgsrc =
string.Format("data:image/jpg;base64,{0}", base64);
                    }

                    
                </div>
            }
        }
    </div>
</div>
}

<br />
<br />

<div class="backgroundWhite container">

    <ul id="menu-filters" class="menu-filter-list list-inline text-center">
        <li class="active btn btn-secondary ml-1 mr-1" data-filter=".menu-
restaurant">Alles</li>

        @foreach (var item in Model.Category)
        {
            <li class="ml-1 mr-1" data-filter=".@item.Name.Replace("
",string.Empty)">@item.Name</li>
        }
    </ul>

    @foreach (var category in Model.Category)
    {
        <div class="row" id="menu-wrapper">
            <partial name="_ThumbnailAreaPartial"
model="@Model.MenuItem.Where(u=>u.Category.Name.Equals(category.Name))" />
        </div>
    }
}

```

```

    }
</div>

@section Scripts{
    <script src="https://code.jquery.com/jquery-3.3.1.js"
        integrity="sha256-2Kok7Mb0yxpqUVvAk/HJ2jig0SYS2auK4Pfzbm7uH60="
        crossorigin="anonymous"></script>

    <script>var posts = $('.post');

    (function ($) {

        $("#menu-filters li").click(function () {
            $("#menu-filters li").removeClass('active btn btn-secondary');
            $(this).addClass('active btn btn-secondary');

            var selectedFilter = $(this).data("filter");

            $(".menu-restaurant").fadeOut();

            setTimeout(function () {
                $(selectedFilter).slideDown();
            }, 300);
        });

    })(jQuery);</script>
}

```

4 GDPR: wetgeving i.v.m. de bescherming van persoonsgegevens

Om de toestemming te vragen aan de gebruiker om cookies te gebruiken (GDPR):

Voeg een Partial Razor view “_CookieConsentPartial” toe onder de folder Views/Shared

```

@using Microsoft.AspNetCore.Http.Features

@{
    var consentFeature = Context.Features.Get<ITrackingConsentFeature>();
    var showBanner = !consentFeature?.CanTrack ?? false;
    var cookieString = consentFeature?.CreateConsentCookie();
}

@if (showBanner)
{
    <div id="cookieConsent" class="alert alert-info alert-dismissible fade show"
        role="alert">
        Use this space to summarize your privacy and cookie use policy. <a asp-area=""
        asp-controller="Home" asp-action="Privacy">Learn More</a>.
        <button type="button" class="accept-policy close" data-dismiss="alert" aria-
        label="Close" data-cookie-string="@cookieString">
            <span aria-hidden="true">Accept</span>
        </button>
    </div>
}

```

```

<script>
  (function () {
    var button = document.querySelector("#cookieConsent button[data-cookie-
string]");
    button.addEventListener("click", function (event) {
      document.cookie = button.dataset.cookieString;
    }, false);
  })();
</script>
}

```

Als laatste layout wijzigingen toevoegen van class styles aan wwwroot/css/site.css:

```

....
.backgroundWhiteBorder {
  background-color: white;
  padding: 30px;
  border-radius: 10px;
  margin-bottom: 30px;
  border: 1px solid #ddd;
}

.backgroundWhiteBorder10Padding {
  background-color: white;
  padding: 10px;
  border-radius: 10px;
  border: 1px solid #ddd;
}

.menu-filter-list li {
  display: inline-block;
  cursor: pointer;
  padding: 10px 20px 10px;
  text-transform: uppercase;
  background: #f5f5f5;
  border-radius: 5px;
  font-weight: 700;
  font-size: 13px;
  -moz-transition: all 0.3s;
  -o-transition: all 0.3s;
  -webkit-transition: all 0.3s;
  transition: all 0.3s;
  font-family: "Poppins", sans-serif;
}

.menu-filter-list li.is-checked, .menu-filter-list li:hover {
  background-color: #545b62;
  color: #fff;
}

```

- aanpassen van css styles in wwwroot/css/site.css



site.css

5 Authenticatie en Autorisatie

5.1 Aanmaken model voor ApplicationUser

Onder de folder models, voeg een nieuwe class toe voor ApplicationUser

```
public class ApplicationUser : IdentityUser
{
    public string Name { get; set; }
    public string StreetAddress { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string PostalCode { get; set; }
}
```

5.2 Aanpassen ApplicationDbContext

Voeg aan de class ApplicationDbContext DbSet public properties toe voor SubCategory, MenuItem, Coupon en ApplicationUser

```
public DbSet<ApplicationUser> ApplicationUser { get; set; }
```

5.3 Aanpassen Startup.cs

Vervang in ConfigureServices IdentityUser door ApplicationIdentityUser. We speciëren eveneens IdentityRole bij AddIdentity service omdat we rollen zullen gebruiken in onze app

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));

    services.AddIdentity<IdentityUser, IdentityRole>()
        .AddDefaultTokenProviders()
        .AddEntityFrameworkStores<ApplicationDbContext>();

    services.AddControllersWithViews();
    services.AddRazorPages().AddRazorRuntimeCompilation();
}
```

5.4 Database migraties toevoegen voor de aangepaste AspNetUsers tabel en database updaten

Indien je applicatie goed werkt, kan je nu de tabellen in de database creëren.

We doen dit via de **NuGet Package Manager Console** door eerst data-migraties op te zetten en daarna update-database commando

```
PM> add-migration ChangedAspNetUsersToDb
```

PM> update-database

5.5 Aanpassen van de Register Razor Page en Code-behind

1. Open `Areas/Identity/Pages/Account/Register.cshtml` en voeg extra form fields toe voor de properties van de `ApplicationUser` (afgeleide class van `IdentityUser`):

```
@page
@model RegisterModel

@{
    ViewData["Title"] = "Registreen";
}

<br />
<h2 class="text-info">Een nieuwe account aanmaken</h2>
<br />

<form method="post" asp-route-returnUrl="@Model.ReturnUrl">
    <div class="border backgroundWhite">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Input.Name" class="col-form-label"></label>
            </div>
            <div class="col-5">
                <input asp-for="Input.Name" class="form-control" />
            </div>
            <span asp-validation-for="Input.Name" class="text-danger"></span>
        </div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Input.Email" class="col-form-label"></label>
            </div>
            <div class="col-5">
                <input asp-for="Input.Email" class="form-control" />
            </div>
            <span asp-validation-for="Input.Email" class="text-danger"></span>
        </div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Input.PhoneNumber" class="col-form-label"></label>
            </div>
            <div class="col-5">
                <input asp-for="Input.PhoneNumber" class="form-control" />
            </div>
            <span asp-validation-for="Input.PhoneNumber" class="text-danger"></span>
        </div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Input.StreetAddress" class="col-form-label"></label>
            </div>
            <div class="col-5">
```

```

        <input asp-for="Input.StreetAddress" class="form-control" />
    </div>
    <span asp-validation-for="Input.StreetAddress" class="text-
danger"></span>
</div>
<div class="form-group row">
    <div class="col-2">
        <label asp-for="Input.City" class="col-form-label"></label>
    </div>
    <div class="col-5">
        <input asp-for="Input.City" class="form-control" />
    </div>
    <span asp-validation-for="Input.City" class="text-danger"></span>
</div>
<div class="form-group row">
    <div class="col-2">
        <label asp-for="Input.State" class="col-form-label"></label>
    </div>
    <div class="col-5">
        <input asp-for="Input.State" class="form-control" />
    </div>
    <span asp-validation-for="Input.State" class="text-danger"></span>
</div>
<div class="form-group row">
    <div class="col-2">
        <label asp-for="Input.PostalCode" class="col-form-
label"></label>
    </div>
    <div class="col-5">
        <input asp-for="Input.PostalCode" class="form-control" />
    </div>
    <span asp-validation-for="Input.PostalCode" class="text-
danger"></span>
</div>

    <div class="form-group row">
        <div class="col-2">
            <label asp-for="Input.Password" class="col-form-label"></label>
        </div>
        <div class="col-5">
            <input asp-for="Input.Password" class="form-control" />
        </div>
        <span asp-validation-for="Input.Password" class="text-
danger"></span>
    </div>

    <div class="form-group row">
        <div class="col-2">
            <label asp-for="Input.ConfirmPassword" class="col-form-
label"></label>
        </div>
        <div class="col-5">
            <input asp-for="Input.ConfirmPassword" class="form-control" />
        </div>
        <span asp-validation-for="Input.ConfirmPassword" class="text-
danger"></span>
    </div>

```

```

        <div class="form-group row">
            <div class="col-5 offset-2">
                <button type="submit" class="btn btn-primary form-
control">Registreer</button>
            </div>
        </div>
    </div>
</form>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

```

2. Open Areas/Identity/Pages/Account/Register.cshtml.cs en voeg de extra properties toe aan het Input model:

```

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

    [Required]
    [StringLength(100, ErrorMessage = "Het {0} moet minstens {2} en maximaal
{1} tekens bevatten.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Wachtwoord")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Bevestig Wachtwoord")]
    [Compare("Password", ErrorMessage = "Het wachtwoord en bevestiging van
wachtwoord komen niet overeen")]
    public string ConfirmPassword { get; set; }

    [Required]
    [Display(Name = "Naam")]
    public string Name { get; set; }
    [Display(Name = "Straat")]
    public string StreetAddress { get; set; }
    [Display(Name = "TelNr")]
    public string PhoneNumber { get; set; }
    [Display(Name = "Gemeente")]
    public string City { get; set; }
    [Display(Name = "Provincie")]
    public string State { get; set; }
    [Display(Name = "Postcode")]
    public string PostalCode { get; set; }
}

```

3. Wijzig in de Register.cshtml.cs eveneens de creatie van IdentityUser object naar een ApplicationUser (in methode OnPostAsync)

```

public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{

```

```

        if (ModelState.IsValid)
        {
            //var user = new IdentityUser { UserName = Input.Email, Email =
Input.Email };
            var user = new ApplicationUser
            {
                UserName = Input.Email,
                Email = Input.Email,
                Name = Input.Name,
                City = Input.City,
                StreetAddress = Input.StreetAddress,
                State = Input.State,
                PostalCode = Input.PostalCode,
                PhoneNumber = Input.PhoneNumber
            };
            var result = await _userManager.CreateAsync(user, Input.Password);
            if (result.Succeeded)
            {
                ...
            }
        }
    }
}

```

6 Authorisatie

- 6.1 De namen van de verschillende rollen gaan we bijhouden in constanten in een Utility class.

Maak een nieuwe folder Utility aan en maak daarin een nieuwe static class SD:

```

namespace PittigRestoMVC.Utility
{
    namespace Pittig.Utility
    {
        public static class SD
        {
            public const string DefaultFoodImage = "default_food.png";

            public const string ManagerUser = "Manager";
            public const string KitchenUser = "Kitchen";
            public const string FrontDeskUser = "FrontDesk";
            public const string CustomerEndUser = "Customer";
        }
    }
}

```

- 6.2 Database initialiseren met rollen

We willen deze rollen in de database plaatsen (Table ASPNetRoles).

Er zijn verschillende manieren om dit te doen.

We kunnen dit doen via een Initializer te gebruiken, die de rollen zal creëren indien deze nog niet bestaan:

Maak onder de folder Data een Interface IDbInitializer en class DbInitializer:

```

namespace PittigRestMVC.Data
{
    public interface IDbInitializer
    {
        void Initialize();
    }
}

```

```

}

namespace PittigRestoMVC.Data
{
    public class DbInitializer : IDbInitializer
    {
        private readonly ApplicationDbContext _db;
        private readonly UserManager<IdentityUser> _userManager;
        private readonly RoleManager<IdentityRole> _roleManager;

        public DbInitializer(ApplicationDbContext db, UserManager<IdentityUser>
userManager, RoleManager<IdentityRole> roleManager)
        {
            _db = db;
            _roleManager = roleManager;
            _userManager = userManager;
        }

        public async void Initialize()
        {
            try
            {
                if(_db.Database.GetPendingMigrations().Count()>0)
                {
                    _db.Database.Migrate();
                }
            }
            catch (Exception ex)
            {
                throw(ex);
            }

            if (_db.Roles.Any(r => r.Name == SD.ManagerUser)) return;

            _roleManager.CreateAsync(new
IdentityRole(SD.ManagerUser)).GetAwaiter().GetResult();
            _roleManager.CreateAsync(new
IdentityRole(SD.FrontDeskUser)).GetAwaiter().GetResult();
            _roleManager.CreateAsync(new
IdentityRole(SD.KitchenUser)).GetAwaiter().GetResult();
            _roleManager.CreateAsync(new
IdentityRole(SD.CustomerEndUser)).GetAwaiter().GetResult();

            _userManager.CreateAsync(new ApplicationUser
{
                UserName = "admin@gmail.com",
                Email = "admin@gmail.com",
                Name = "Admin GMAIL",
                EmailConfirmed = true,
                PhoneNumber = "1112223333"
            }, "Admin123*").GetAwaiter().GetResult();

            IdentityUser user = await _db.Users.FirstOrDefaultAsync(u => u.Email ==
"admin@gmail.com");

            await _userManager.AddToRoleAsync(user, SD.ManagerUser);
        }
    }
}

```

```

    }
}
}

```

6.3 Pas Startup.cs aan:

1. In ConfigureServices voegen we de DbInitializer class toe via AddScoped

```

public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));

    services.AddIdentity<IdentityUser, IdentityRole>()
        .AddDefaultTokenProviders()
        .AddEntityFrameworkStores<ApplicationDbContext>();

    services.AddScoped<IDbInitializer, DbInitializer>();
    services.AddControllersWithViews();
    services.AddRazorPages().AddRazorRuntimeCompilation();
}

```

2. In Configure roepen we deze aan:

```

public void Configure(IApplicationBuilder app, IWebHostEnvironment env,
    IDbInitializer dbInitializer)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseDatabaseErrorPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this for
        production scenarios, see https://aka.ms/aspnetcore-hsts.
        app.UseHsts();
    }
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();
    dbInitializer.Initialize();
    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern:
                "{area=Customer}/{controller=Home}/{action=Index}/{id?}");
        endpoints.MapRazorPages();
    });
}

```

```
    }
}
```

6.4 Roles toevoegen aan Register Razor page en Code Behind

Open Areas/Identity/Pages/Account/Register.cshtml.cs (Razor Page Code behind van Register.cshtml)

En voeg RoleManager toe via DI:

```
public class RegisterModel : PageModel
{
    private readonly SignInManager<IdentityUser> _signInManager;
    private readonly UserManager<IdentityUser> _userManager;
    private readonly ILogger<RegisterModel> _logger;
    private readonly IEmailSender _emailSender;
    private readonly RoleManager<IdentityRole> _roleManager;

    public RegisterModel(
        UserManager<IdentityUser> userManager,
        SignInManager<IdentityUser> signInManager,
        ILogger<RegisterModel> logger,
        IEmailSender emailSender,
        RoleManager<IdentityRole> roleManager)
    {
        _userManager = userManager;
        _signInManager = signInManager;
        _logger = logger;
        _emailSender = emailSender;
        _roleManager = roleManager;
    }

    ...
}
```

We gaan op basis van de rol bepaalde delen van de Razor Views al dan niet tonen:
Open Shared/_Layout.cshtml en pas deze aan

- aanpassen van Views/Shared/_Layout.cshtml



_Layout.cshtml