

Model ASP.NET MVC Core 3.1 Web App - Deel 3

1 Start project

In deel 2 zijn we gekomen tot deze code:

<https://github.com/CSharpSyntraWest/PittigRestoMVC>

We starten vanaf deze code

2 Aanmaken models voor MenuItem

1. Maak een nieuwe class MenuItem onder de folder Models:

```
public class MenuItem
{
    public int Id { get; set; }

    [Required]
    [Display(Name = "Naam")]
    public string Name { get; set; }
    [Display(Name = "Beschrijving")]
    public string Description { get; set; }
    [Display(Name = "Pittigheid")]
    public string Spicyness { get; set; }
    public enum ESpicy { NA=0, Geen=1, Weinig=2, Sterk=3}

    [Display(Name = "Foto")]
    public string Image { get; set; }

    [Display(Name="Categorie")]
    public int CategoryId { get; set; }

    [ForeignKey("CategoryId")]
    public virtual Category Category { get; set; }

    [Display(Name = "Sub categorie")]
    public int SubCategoryId { get; set; }

    [ForeignKey("SubCategoryId")]
    public virtual SubCategory SubCategory { get; set; }

    [Range(1,int.MaxValue, ErrorMessage = " Prijs moet hoger zijn dan €{1}")]
    [Display(Name = "Prijs (EUR)")]
    public double Price { get; set; }
}
```

We gaan onze CRUD pagina's binden aan een nieuwe MenuItemViewModel class

2. Maak onder de folder Models/ViewModels een nieuwe class MenuItemViewModel:

```
public class MenuItemViewModel
{
    public MenuItem MenuItem { get; set; }
    public IEnumerable<Category> Category { get; set; }
    public IEnumerable<SubCategory> SubCategory { get; set; }

}
```

2.1 Aanpassen ApplicationDbContext

Voeg aan de class ApplicationDbContext DbSet public property toe voor MenuItem:

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

2.2 Database migraties toevoegen voor de nieuwe MenuItem 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 AddedMenuItemToDb
```

```
PM> update-database
```

Indien je bij update-database deze foutmelding krijgt:

```
Introducing FOREIGN KEY constraint 'FK_MenuItem_SubCategory_SubCategoryId' on table 'MenuItem' may cause cycles or multiple cascade paths. Specify ON DELETE NO ACTION or ON UPDATE NO ACTION, or modify other FOREIGN KEY constraints.
Could not create constraint or index. See previous errors.
PM>
```

Wijzig dan in de class onder de migrations folder waar dat de tabel MenuItem via fluent API wordt aangemaakt, de

```
onDelete: ReferentialAction.Cascade;
```

naar

```
onDelete: ReferentialAction.NoAction;
```

```
constraints: table =>
{
    table.PrimaryKey("PK_MenuItem", x => x.Id);
    table.ForeignKey(
        name: "FK_MenuItem_Category_CategoryId",
        column: x => x.CategoryId,
        principalTable: "Category",
        principalColumn: "Id",
        onDelete: ReferentialAction.NoAction);
    table.ForeignKey(
        name: "FK_MenuItem_SubCategory_SubCategoryId",
        column: x => x.SubCategoryId,
        principalTable: "SubCategory",
        principalColumn: "Id",
        onDelete: ReferentialAction.NoAction);
});
```

3 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; }
}

```

3.1 Aanpassen ApplicationDbContext

Voeg aan de class ApplicationDbContext DbSet public property toe voor Coupon

```

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

```

3.2 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.

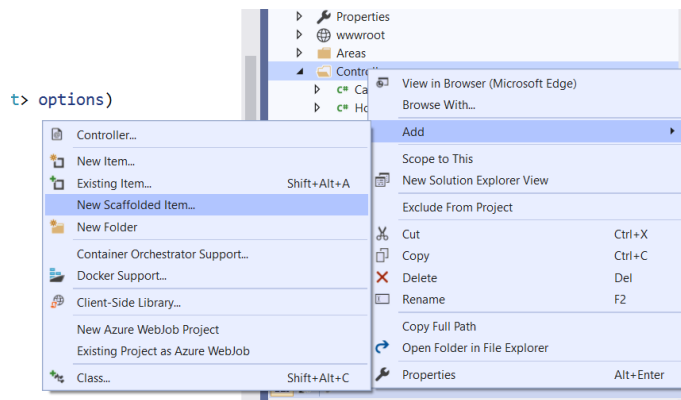
We doen dit via de **NuGet Package Manager Console** door eerst data-migraties op te zetten en daarna update-database commando

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

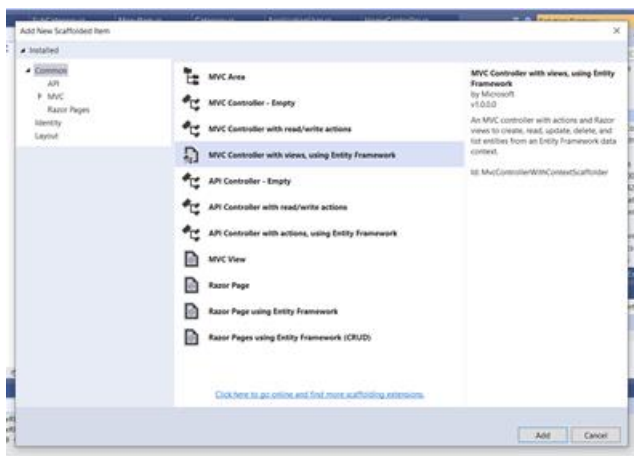
```
PM> update-database
```

3.3 Aanmaken Controllers en Views voor CRUD operaties voor MenuItem en Coupon

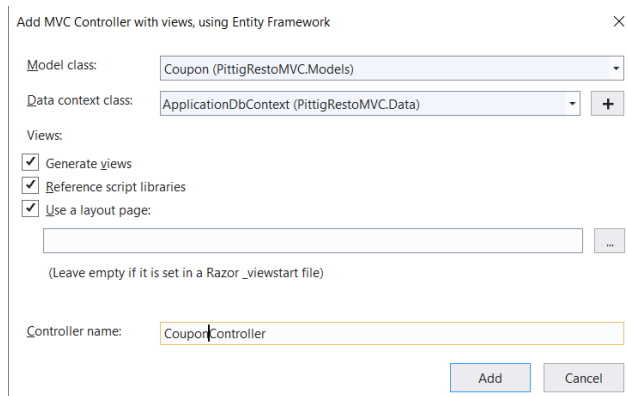
Rechtsklik op de folder Controller en kies Add/New Scaffolded Item...



En daarna Mvc Controller with views, using Entity Framework



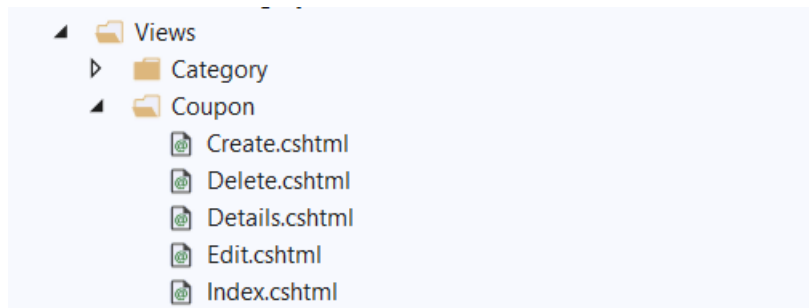
Kies in het volgende popup-venster bv als **model class Coupon**



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

Doe nu hetzelfde voor de MenuItem:

Maak een MenuItemController aan en views via Add/New scaffolded item

Test de applicatie via deze urls:

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

<https://localhost:44365/Customer/Home/index>

<https://localhost:44365/Admin/Category/index>

<https://localhost:44365/Admin/SubCategory/index>

<https://localhost:44365/Admin/Coupon/index>

<https://localhost:44365/Admin/MenuItem/index>

4 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; }
}
```

4.1 Aanpassen ApplicationDbContext

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

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

4.2 Database migraties toevoegen voor de aangepasteAspNetUsers 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 Aanpassen van MenuItem Controller en CRUD views om images te kunnen uploaden en bewaren op de webserver

Om images te kunnen uploaden en te bewaren onder de wwwroot/images folder vanaf de Create MenuItem razor view, moeten we de Create action methoden aanpassen van de **MenuItemController**:

1. Open MenuItemController onder de folder Areas/Admin/Controllers en wijzig de Create action methoden:

```
[Area("Admin")]
public class MenuItemController : Controller
{
    private readonly ApplicationDbContext _context;
    private readonly IWebHostEnvironment _hostingEnvironment;

    [BindProperty]
    public MenuItemViewModel MenuItemVM { get; set; }

    public MenuItemController(ApplicationDbContext db, IWebHostEnvironment
hostingEnvironment)
    {
```

```

        _context = db;
        _hostingEnvironment = hostingEnvironment;
        MenuItemVM = new MenuItemViewModel()
        {
            Category = _context.Category,
            MenuItem = new MenuItem()
        };
    }

    // GET: Admin/MenuItem
    public async Task<IActionResult> Index()
    {
        var applicationDbContext = _context.MenuItem.Include(m =>
m.Category).Include(m => m.SubCategory);
        return View(await applicationDbContext.ToListAsync());
    }

    //GET - CREATE
    public IActionResult Create()
    {
        return View(MenuItemVM);
    }

    [HttpPost, ActionName("Create")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> CreatePOST()
    {
        MenuItemVM.MenuItem.SubCategoryId =
Convert.ToInt32(Request.Form["SubCategoryId"].ToString());

        if (!ModelState.IsValid)
        {
            return View(MenuItemVM);
        }

        _context.MenuItem.Add(MenuItemVM.MenuItem);
        await _context.SaveChangesAsync();

        //Work on the image saving section

        string webRootPath = _hostingEnvironment.WebRootPath;
        var files = HttpContext.Request.Form.Files;

        var menuItemFromDb = await
_context.MenuItem.FindAsync(MenuItemVM.MenuItem.Id);

        if (files.Count > 0)
        {
            //files has been uploaded
            var uploads = Path.Combine(webRootPath, "images");
            var extension = Path.GetExtension(files[0].FileName);

            using (var fileStream = new FileStream(Path.Combine(uploads,
MenuItemVM.MenuItem.Id + extension), FileMode.Create))
            {
                files[0].CopyTo(fileStream);
            }
            menuItemFromDb.Image = @"\"images\" + MenuItemVM.MenuItem.Id + extension;
        }
    }

```

```

        else
        {
            //no file was uploaded, so use default
            var uploads = Path.Combine(webRootPath, @"images\" +
"default_food.png");//SD.DefaultFoodImage);
            System.IO.File.Copy(uploads, webRootPath + @"\images\" +
MenuItemVM.MenuItem.Id + ".png");
            menuItemFromDb.Image = @"\images\" + MenuItemVM.MenuItem.Id + ".png";
        }

        await _context.SaveChangesAsync();

        return RedirectToAction(nameof(Index));
    }

    ...
}

```

2. Open create.cshtml razor view onder de folder Areas/Admin/Views/MenuItem en wijzig de inhoud:

```

@model PittigRestoMVC.Models.ViewModels.MenuItemViewModel
@using PittigRestoMVC.Extensions

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

<br />
<h2 class="text-info">Create MenuItem</h2>
<br />

<form method="post" asp-action="Create" 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="MenuItem.Name" class="col-form-label"></label>
            </div>
            <div class="col-5">
                <input asp-for="MenuItem.Name" class="form-control" />
            </div>
            <span asp-validation-for="MenuItem.Name" class="text-danger"></span>
        </div>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="MenuItem.Description" class="col-form-label"></label>
            </div>
            <div class="col-5">
                @Html.TextAreaFor(m => m.MenuItem.Description)
            </div>
            <span asp-validation-for="MenuItem.Description" class="text-
danger"></span>
        </div>
        <div class="form-group row">
            <div class="col-2">

```



```

        <label asp-for="MenuItem.Price" class="col-form-label"></label>
    </div>
    <div class="col-5">
        <input asp-for="MenuItem.Price" class="form-control" />
    </div>
    <span asp-validation-for="MenuItem.Price" class="text-danger"></span>
</div>
<div class="form-group row">
    <div class="col-2">
        <label asp-for="MenuItem.Image" class="col-form-label"></label>
    </div>
    <div class="col-5">
        <input type="file" name="files" multiple class="form-control" />
    </div>
</div>

<div class="form-group row">
    <div class="col-2">
        <label asp-for="MenuItem.CategoryId" class="col-form-label"></label>
    </div>
    <div class="col-5">
        <select asp-for="@Model.MenuItem.CategoryId" id="CategoryId" asp-
items="Model.Category.ToSelectListItem(Model.MenuItem.CategoryId)" class="form-
control"></select>
    </div>
</div>
<div class="form-group row">
    <div class="col-2">
        <label asp-for="MenuItem.SubCategoryId" class="col-form-
label"></label>
    </div>
    <div class="col-5">
        <select asp-for="@Model.MenuItem.SubCategoryId" name="SubCategoryId"
id="SubCategoryId" asp-items="@((new SelectList(string.Empty, "Id", "Name")))"
class="form-control"></select>
    </div>
</div>
<div class="form-group row">
    <div class="col-2">
        <label asp-for="MenuItem.Spicyness" class="col-form-label"></label>
    </div>
    <div class="col-5">
        <select asp-for="MenuItem.Spicyness" asp-
items="Html.GetEnumSelectList<MenuItem.ESpicy>()" class="form-control"></select>
    </div>
</div>

<div class="form-group row">
    <div class="col-5 offset-2">
        <partial name="_CreateAndBackToListButton" />
    </div>
</div>
</div>
</form>

```

@section Scripts{

```

<script type="text/javascript">
    $(document).ready(function () {
        getSubCategory();
        $('#CategoryId').change(function () {
            getSubCategory();
        });
    });

    function getSubCategory() {
        var url = '@Url.Content("~/")' + "Admin/SubCategory/GetSubCategory";
        var ddlsource = '#CategoryId';
        $.getJSON(url, { id: $(ddlsource).val() }, function (data) {
            var items = '';
            $('#SubCategoryId').empty();
            $.each(data, function (i, subcategory) {
                items += "<option value='" + subcategory.value + "'>" +
subcategory.text + "</option>";
            });

            $('#SubCategoryId').html(items);
        })
    }
</script>

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

```

3. We passen hetzelfde aan voor de Edit action methods in de MenuItemController:

```

//GET - EDIT
public async Task<IActionResult> Edit(int? id)
{
    if(id==null)
    {
        return NotFound();
    }

    MenuItemVM.MenuItem = await _db.MenuItem.Include(m => m.Category).Include(m
=> m.SubCategory).SingleOrDefaultAsync(m => m.Id == id);
    MenuItemVM.SubCategory = await _db.SubCategory.Where(s => s.CategoryId ==
MenuItemVM.MenuItem.CategoryId).ToListAsync();

    if(MenuItemVM.MenuItem ==null)
    {
        return NotFound();
    }
    return View(MenuItemVM);
}

[HttpPost, ActionName("Edit")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> EditPOST(int? id)
{

```

```

        if(id==null)
        {
            return NotFound();
        }
        MenuItemVM.MenuItem.SubCategoryId =
Convert.ToInt32(Request.Form["SubCategoryId"].ToString());

        if (!ModelState.IsValid)
        {
            MenuItemVM.SubCategory = await _db.SubCategory.Where(s => s.CategoryId ==
MenuItemVM.MenuItem.CategoryId).ToListAsync();
            return View(MenuItemVM);
        }

        //Work on the image saving section

        string webRootPath = _hostingEnvironment.WebRootPath;
        var files = HttpContext.Request.Form.Files;

        var menuItemFromDb = await _db.MenuItem.FindAsync(MenuItemVM.MenuItem.Id);

        if (files.Count > 0)
        {
            //New Image has been uploaded
            var uploads = Path.Combine(webRootPath, "images");
            var extension_new = Path.GetExtension(files[0].FileName);

            //Delete the original file
            var imagePath = Path.Combine(webRootPath,
menuItemFromDb.Image.TrimStart('\\'));

            if(System.IO.File.Exists(imagePath))
            {
                System.IO.File.Delete(imagePath);
            }

            //we will upload the new file
            using (var fileStream = new FileStream(Path.Combine(uploads,
MenuItemVM.MenuItem.Id + extension_new), FileMode.Create))
            {
                files[0].CopyTo(fileStream);
            }
            menuItemFromDb.Image = @"images\" + MenuItemVM.MenuItem.Id +
extension_new;
        }

        menuItemFromDb.Name = MenuItemVM.MenuItem.Name;
        menuItemFromDb.Description = MenuItemVM.MenuItem.Description;
        menuItemFromDb.Price = MenuItemVM.MenuItem.Price;
        menuItemFromDb.Spicyiness = MenuItemVM.MenuItem.Spicyiness;
        menuItemFromDb.CategoryId = MenuItemVM.MenuItem.CategoryId;
        menuItemFromDb.SubCategoryId = MenuItemVM.MenuItem.SubCategoryId;

        await _db.SaveChangesAsync();

        return RedirectToAction(nameof(Index));
    }

```

4. En passen eveneens de Edit.cshtml aan onder Areas/Views/MenuItem:

```
@model PittigRestoMVC.Models.ViewModels.MenuItemViewModel
@using PittigRestoMVC.Extensions

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

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

<form method="post" asp-action="Create" enctype="multipart/form-data">
    <div class="border backgroundWhite row">
        <input type="hidden" asp-for="MenuItem.Id" />

        <div asp-validation-summary="ModelOnly" class="text-danger"></div>

        <div class="col-8">
            <div class="form-group row">
                <div class="col-4">
                    <label asp-for="MenuItem.Name" class="col-form-label"></label>
                </div>
                <div class="col-8">
                    <input asp-for="MenuItem.Name" class="form-control" />
                </div>
                <span asp-validation-for="MenuItem.Name" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <div class="col-4">
                    <label asp-for="MenuItem.Description" class="col-form-label"></label>
                </div>
                <div class="col-8">
                    @Html.TextAreaFor(m => m.MenuItem.Description)
                </div>
                <span asp-validation-for="MenuItem.Description" class="text-
danger"></span>
            </div>
            <div class="form-group row">
                <div class="col-4">
                    <label asp-for="MenuItem.Price" class="col-form-label"></label>
                </div>
                <div class="col-8">
                    <input asp-for="MenuItem.Price" class="form-control" />
                </div>
                <span asp-validation-for="MenuItem.Price" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <div class="col-4">
                    <label asp-for="MenuItem.Image" class="col-form-label"></label>
                </div>
                <div class="col-8">
                    <input type="file" name="files" multiple class="form-control" />
                </div>
            </div>
        </div>
    </div>
</form>
```

```

        <div class="form-group row">
            <div class="col-4">
                <label asp-for="MenuItem.CategoryId" class="col-form-label"></label>
            </div>
            <div class="col-8">
                <select asp-for="@Model.MenuItem.CategoryId" id="CategoryId" asp-
items="Model.Category.ToSelectListItem(Model.MenuItem.CategoryId)" class="form-
control"></select>
            </div>
        </div>
        <div class="form-group row">
            <div class="col-4">
                <label asp-for="MenuItem.SubCategoryId" class="col-form-
label"></label>
            </div>
            <div class="col-8">
                <select asp-for="@Model.MenuItem.SubCategoryId" name="SubCategoryId"
id="SubCategoryId" asp-items="@((new SelectList(string.Empty, "Id", "Name")))" class="form-
control"></select>
            </div>
        </div>
        <div class="form-group row">
            <div class="col-4">
                <label asp-for="MenuItem.Spicyness" class="col-form-label"></label>
            </div>
            <div class="col-8">
                <select asp-for="MenuItem.Spicyness" asp-
items="Html.GetEnumSelectList<MenuItem.ESpicy>()" class="form-control"></select>
            </div>
        </div>
        </div>
        <div class="col-3 offset-1">
            
        </div>

        <div class="col-8">
            <div class="col-8 offset-4">
                <partial name="_EditAndBackToListButton" model="Model.MenuItem.Id" />
            </div>
        </div>
    </div>
</form>

@section Scripts{
    <script type="text/javascript">
        $(document).ready(function () {
            getSubCategory();
            $('#CategoryId').change(function () {
                getSubCategory();
            });
        });

        function getSubCategory() {

```

```

var url = '@Url.Content("~/")' + "Admin/SubCategory/GetSubCategory";
var ddlsource = '#CategoryId';
$.getJSON(url, { id: $(ddlsource).val() }, function (data) {
    var items = '';
    $('#SubCategoryId').empty();
    $.each(data, function (i, subcategory) {
        items += "<option value='" + subcategory.value + "'>" +
subcategory.text + "</option>";
    });

    $('#SubCategoryId').html(items);
})
}
</script>

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

```

Deze Edit Razor view maakt gebruik van een Partial View _EditAndBackToListButton.cshtml

5. Maak een Partial View _EditAndBackToListButton.cshtml deze aan onder Views/Shared:

```

@model int

<div class="row">
    <div class="col-6">
        <input type="submit" class="btn btn-info form-control" asp-route-id="@Model"
value="Update" />
    </div>
    <div class="col-6">
        <a asp-action="Index" class="btn btn-success form-control">Terug naar
lijst</a>
    </div>
</div>

```

6 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 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

```
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);
}
```

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/Customer/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.

7 Pas View van homepagina aan

Open de Index.cshtml onder Area/Customer/Home/

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

```
@model Pittig.Models.ViewModels.IndexViewModel
```

In de View gaan we een lijst van Categorieën tonen:

```
<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>
```