Project schoolartikelen

Basistabellen :

1. Brand : Id, BrandName
2. Category: Id, CategoryName
3. Article : Id, ArticleName, BrandId, CategoryId, Price, Description, Score

Bijkomende tabellen :

4. User : Id, Email, Name, FirstName, Password, IsAdmin
5. Basket : Id, UserId, ArticleId, Count
6. Order : Id, UserId, DateTimeStamp
7. OderDetail : Id, OrderId, ArticleId, Count, SalesPrice

Stap 1 : applicatie aanmaken

- Type = ASP.Net Core Web app
- Project name = SchoolShop
  Solution name = SchoolShop
- Target framework = .NET 5.0
  Configure for HTTPS

Stap 2 : modellen maken

- Maak in je solution de map **Models** aan
- Maak in de map Models volgende klassen aan :
    - Brand

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

        [Display(Name = "Merk")]
        [Required(ErrorMessage = "Merknaam is vereist")]
        [StringLength(50, MinimumLength = 2, ErrorMessage = "Minimaal 2, maximaal 50
letters")]
        public string BrandName { get; set; }
    }
```

    - Category

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

        [Display(Name = "Categorie")]
        [Required(ErrorMessage = "Categorienaam is vereist")]
        [StringLength(50, MinimumLength = 2, ErrorMessage = "Minimaal 2, maximaal 50
letters")]
        public string CategoryName { get; set; }
        public string ImagePath { get; set; }
    }
```

    - Article

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

        [ForeignKey("Brand")]
        public int BrandId { get; set; }
        [Display(Name = "Merk")]
        public Brand Brand { get; set; }

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

        [Display(Name = "Prijs")]
        [DisplayFormat(DataFormatString = "€ {0:#,##0.00}", ApplyFormatInEditMode =
false)]
        [Range(1, 20000, ErrorMessage = "Kies een waarde tussen 1 en 20.000")]
        [Column(TypeName = "decimal(8, 2)")]
        public decimal Price { get; set; }

        [Display(Name = "Artikelnaam")]
        [Required(ErrorMessage = "Geef een artikel(naam) op !")]
        [StringLength(250, ErrorMessage = "Modelnaam maximaal 250 letters")]
        public string ArticleName { get; set; }


        [Display(Name = "Omschrijving")]
        [Required(ErrorMessage = "Geef een omschrijving op !")]
        public string Description { get; set; }

        [Display(Name = "Score")]
        [DisplayFormat(DataFormatString = "€ {0:#,##0.00}", ApplyFormatInEditMode =
false)]
        [Range(1, 5, ErrorMessage = "Kies een waarde tussen 1 en 5")]
        [Column(TypeName = "decimal(4, 2)")]
        public decimal Score { get; set; }
```

```
        public string ImagePath { get; set; }
    }
```

- User

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

        [Display(Name = "E-mail")]
        [Required(ErrorMessage = "E-mail is vereist")]
        [StringLength(500)]
        [DataType(DataType.EmailAddress, ErrorMessage = "E-mail is niet geldig")]
        public string Email { get; set; }

        [Display(Name = "Acthernaam")]
        [Required(ErrorMessage = "Acthernaam is vereist")]
        [StringLength(50, MinimumLength = 2, ErrorMessage = "Minimaal 2, maximaal 50
letters")]
        public string Name { get; set; }

        [Display(Name = "Voornaam")]
        [Required(ErrorMessage = "Voornaam is vereist")]
        [StringLength(50, MinimumLength = 2, ErrorMessage = "Minimaal 2, maximaal 50
letters")]
        public string FirstName { get; set; }

        [Display(Name = "Paswoord")]
        [Required(ErrorMessage = "Paswoord is vereist")]
        [StringLength(250, MinimumLength = 8, ErrorMessage = "Minimaal 8 karakters")]
        [RegularExpression(@"^((?=.*[a-z])(?=.*[A-Z])(?=.*\d)).+$", ErrorMessage =
"Minstens 1 kleine letter, 1 hoofdletter en 1 cijfer")]
        [DataType(DataType.Password)]
        public string Password { get; set; }
        public bool IsAdmin { get; set; } = false;
    }
```

- Basket

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

        [ForeignKey("User")]
        public int UserId { get; set; }
        [Display(Name = "Gebruiker")]
        public User User { get; set; }

        [ForeignKey("Article")]
        public int ArticleId { get; set; }
        [Display(Name = "Artikel")]
        public Article Article { get; set; }

        [Display(Name = "Aantal")]
        [Range(1, 100, ErrorMessage = "Kies een waarde tussen 1 en 100")]
        public int Count { get; set; } = 1;

    }
```

- OrderDetail

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

        [ForeignKey("Order")]
        public int OrderId { get; set; }
```

```
        [ForeignKey("Article")]
        public int ArticleId { get; set; }
        [Display(Name = "Artikel")]
        public Article Article { get; set; }

        [Display(Name = "Aantal")]
        public int Count { get; set; }

        [Display(Name = "Prijs")]
        [DisplayFormat(DataFormatString = "€ {0:#,##0.00}", ApplyFormatInEditMode =
false)]
        [Column(TypeName = "decimal(8, 2)")]
        public decimal SalesPrice { get; set; }

    }
```

- Order

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

        [ForeignKey("User")]
        public int UserId { get; set; }
        [Display(Name = "Klant")]
        public User User { get; set; }

        [Column(TypeName = "Date")]
        public DateTime DateTimeStamp { get; set; }

        public List<OrderDetail> OrderDetails { get; set; }

    }
```

Stap 3 : helper klassen

- Maak in je solution de map Helpers aan.
- Voeg via de Package Manager het pakket BCrypt.Net-Core toe aan je solution
- Voeg volgende klassen toe aan deze map :

    - Encoding

```
    public class Encoding
    {
                private static string GetRandomSalt()
                {
                        return BCrypt.Net.BCrypt.GenerateSalt(12);
                }
                public static string HashPassword(string password)
                {
                        return BCrypt.Net.BCrypt.HashPassword(password, GetRandomSalt());
                }
                public static bool ValidatePassword(string password, string correctHash)
                {
                        return BCrypt.Net.BCrypt.Verify(password, correctHash);
                }
        public static string EncryptString(string InputText, string Password)
        {
            RijndaelManaged RijndaelCipher = new RijndaelManaged();
            byte[] PlainText = System.Text.Encoding.Unicode.GetBytes(InputText);
            byte[] Salt = System.Text.Encoding.ASCII.GetBytes(Password.Length.ToString());
            PasswordDeriveBytes SecretKey = new PasswordDeriveBytes(Password, Salt);
            ICryptoTransform Encryptor =
RijndaelCipher.CreateEncryptor(SecretKey.GetBytes(32), SecretKey.GetBytes(16));
            System.IO.MemoryStream memoryStream = new System.IO.MemoryStream();
            CryptoStream cryptoStream = new CryptoStream(memoryStream, Encryptor,
CryptoStreamMode.Write);
            cryptoStream.Write(PlainText, 0, PlainText.Length);
            cryptoStream.FlushFinalBlock();
            byte[] CipherBytes = memoryStream.ToArray();
            memoryStream.Close();
            cryptoStream.Close();
            string EncryptedData = Convert.ToBase64String(CipherBytes);
            return EncryptedData;
        }
        public static string DecryptString(string InputText, string Password)
        {
            try
            {
                RijndaelManaged RijndaelCipher = new RijndaelManaged();
                byte[] EncryptedData = Convert.FromBase64String(InputText);
                byte[] Salt =
System.Text.Encoding.ASCII.GetBytes(Password.Length.ToString());
                PasswordDeriveBytes SecretKey = new PasswordDeriveBytes(Password, Salt);
                ICryptoTransform Decryptor =
RijndaelCipher.CreateDecryptor(SecretKey.GetBytes(32), SecretKey.GetBytes(16));
                System.IO.MemoryStream memoryStream = new
System.IO.MemoryStream(EncryptedData);
                CryptoStream cryptoStream = new CryptoStream(memoryStream, Decryptor,
CryptoStreamMode.Read);
                byte[] PlainText = new byte[EncryptedData.Length];
                int DecryptedCount = cryptoStream.Read(PlainText, 0, PlainText.Length);
                memoryStream.Close();
                cryptoStream.Close();
                string DecryptedData = System.Text.Encoding.Unicode.GetString(PlainText,
0, DecryptedCount);
                return DecryptedData;
            }
            catch
            {
                return InputText;
            }
        }

    }
```

- Availability : we laten nog even de constructor in commentaar staan omdat die op dit moment nog een fout gaat opleveren (we beschikken nog niet over een context), en we bij scaffolding (zometeen) geen fouten mogen hebben in onze bestaande code.

```
    public class Availability
    {
        public string UserId { get; } = "";
        public bool IsAdmin { get; } = false;
        public string Email { get; } = "";
        public string ConfigButtonStyle { get; } = "visibility:hidden;";
        public string BasketCount { get; } = "";

        //public Availability(SchoolShopContext context, HttpContext httpContext)
        //{
        //    string userId = httpContext.Request.Cookies["UserID"];
        //    if (!string.IsNullOrEmpty(userId))
        //    {
        //        userId = Encoding.DecryptString(userId, "P@sw00rd");
        //        User user = context.User.FirstOrDefault(m => m.Id == int.Parse(userId));
        //        if (user != null)
        //        {
        //            UserId = userId;
        //            IsAdmin = user.IsAdmin;
        //            Email = user.Email;
        //            if (IsAdmin)
        //            {
        //                ConfigButtonStyle = "visibility:visible;";
        //            }
        //            int findId = int.Parse(userId);
        //            BasketCount = context.Basket.Where(b => b.UserId ==
findId).Count().ToString();
        //        }
        //    }
        //}

    }
```

- Pagination

```
    public class Pagination
    {
        public int PageIndex { get; set; }
        public int FirstPageIndex { get; set; }
        public int LastPageIndex { get; set; }
        public int PreviousPageIndex { get; set; }
        public int NextPageIndex { get; set; }
        public int FirstObjectIndex { get; set; }
        public List<int> PageIndexes { get; set; }

        public Pagination(IQueryable<object> query, int? pageIndex, int itemsPerPage = 6)
        {
            int numberOfObjects = query.Count();
            if(pageIndex == null)
            {
                pageIndex = 0;
            }
            PageIndex = (int)pageIndex;
            int totalPages = (int)Math.Ceiling(1.0 * numberOfObjects / itemsPerPage);
            FirstPageIndex = 0;
            LastPageIndex = totalPages - 1;
            PreviousPageIndex = PageIndex - 1;
            if (PreviousPageIndex < 0)
            {
                PreviousPageIndex = 0;
            }
            NextPageIndex = PageIndex + 1;
            if (NextPageIndex > LastPageIndex)
            {
                NextPageIndex = LastPageIndex;
            }
```

```
            PageIndexes = new List<int>();
            for (int p = 0; p <= LastPageIndex; p++)
            {
                PageIndexes.Add(p);
            }
            FirstObjectIndex = PageIndex * itemsPerPage;
        }

    }
```

Stap 4 : scaffolding

- Maak in de map Pages onderstaande nieuwe submappen aan zodanig dat we straks de scaffolding tool zijn werk kunnen laten doen :
    - /Pages/Articles
    - /pages/Baskets
    - /pages/Brands
    - /pages/Categories
    - /pages/Orders
    - /pages/Users

- Klik met de rechtermuisknop op de submap Articles (het maakt eigenlijk niet zoveel uit met welke map je begint) en kies voor ADD -> New Scaffolded Item -> Razor Pages -> Razor Pages using Entity Framework (CRUD).
  Bij Model Class kies je voor **Article**
  Bij Data Context Class klik je op het + teken en maak je een nieuwe context aan :
  SchoolShop.Data.SchoolShopContext

  De context dien/mag je maar 1 keer aanmaken, en je doet dit bij het eerste model waarvoor je de scaffolding laat lopen. Bij onze volgende modellen (Baskets, Brands …) zullen we deze context (SchoolShop.Data.SchoolShopContext) hergebruiken.

- Doe nu hetzelfde voor de overige mappen :
    - Baskets ➔ Model Class = Basket
    - Brands ➔ Model Class = Brand
    - Categories ➔ Model Class = Category
    - Orders ➔ Model Class = Order
    - Users ➔ Model Class = User

- Sluit alle tabbladen
- Open de klasse Availibility en haal de constructor uit commentaar : er zou zich nu geen fout meer mogen voordoen (eventueel extra Using directieve voorzien).

Stap 5 : Database

- Open /Data/SchoolShopContext.cs.
  Voeg een extra DbSet eigenschap toe :

```
public DbSet<SchoolShop.Models.OrderDetail> OrderDetail { get; set; }
```

- Hoewel dit niet vereist is gaan we toch wat seeding voorzien zodat we straks meteen aan de slag kunnen met onze website.
  Voeg aan de Models map een nieuwe klasse toe met de naam **SeedData.cs**

```
    public class SeedData
    {
        public static void Initialize(IServiceProvider serviceProvider)
        {
            using (var context = new SchoolShopContext(
                serviceProvider.GetRequiredService<
                    DbContextOptions<SchoolShopContext>>()))
            {
                if (!context.Brand.Any() && !context.Category.Any() && !context.Article.Any())
                {
                    context.Category.AddRange(
                        new Category { CategoryName = "Schooltassen" },
                        new Category { CategoryName = "Pennenzakken" },
                        new Category { CategoryName = "Schriften" },
                        new Category { CategoryName = "Mappen" }
                        );
                    context.SaveChanges();

                    context.Brand.AddRange(
                        new Brand { BrandName = "Atoma" },
                        new Brand { BrandName = "Oxford" },
                        new Brand { BrandName = "Kangaro" },
                        new Brand { BrandName = "Leitz" }
                        );
                    context.SaveChanges();
                    context.Article.AddRange(
                        new Article
                        {
                            BrandId = 2,
                            CategoryId = 1,
                            ArticleName = "Eastpak",
                            Price = 40.5M,
                            Description = "De Eastpak Out of Office is een populaire laptoprugzak
uit de Authentic serie van Eastpak. Eastpak. Deze rugzak is geschikt voor dagelijks gebruik en
zeer geschikt als boekentas. De tas heeft een groot hoofdvak met daarin een laptopvak van 29 x 34
x 4 cm geschikt voor een 14” laptop. Daarnaast is er voldoende ruimte voor A4 formaat documenten.
Op de voorzijde van de tas zit een groot ritsvak voor kleinere spullen, zoals een etui,
rekenmachine, telefoon etc. De tas heeft een groot draagcomfort door de dik gefoamde (verstelbare)
schouderbanden en het zacht gevoerde rugpaneel.",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 2,
                            CategoryId = 1,
                            ArticleName = "Micmacbags Friendship Rugtas Camel",
                            Price = 109.55M,
                            Description = "Rugtas uit de Friendship serie van Micmacbags. Ruime
tas met intern rits- en steekvak zeer geschikt voor dagelijks gebruik. Mee naar je werk of gewoon
een dag weg. De Friendship serie heeft een leuke tashanger die doet denken aan de
vriendschapsbandjes die je vroeger zelf knoopte voor je vriendinnen. De tas sluit met een overslag
met gesp die je gemakkelijk in het slot schuift. Het hoofdcompartiment sluit nog extra met een
rits en de opening kun je groter of kleiner maken met drukknopen aan beide zijkanten. ",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 3,
                            CategoryId = 1,
                            ArticleName = "Bagbase Vintage ",
                            Price = 32M,
```

```
                            Description = "100% polyester (600D). Handgreep, gevoerde verstelbare
schouderriemen. PU accenten in contrasterende kleur bruin(behalve black / black). Gevoerde
achterkant. Koordsluiting. Gevoerd laptop compartiment, laptop tot 17. Klep met magneetsluiting.
",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 1,
                            CategoryId = 2,
                            ArticleName = "Eberhard Faber ",
                            Price = 28.9M,
                            Description = "Mooie glitter etui voor school met 3 lagen. Gevuld met
alle belangrijke schoolbenodigdheden: van puntenslijper tot kleurpotlood in uw favoriete kleur. 34
delig; Inhoud: 12 kleurpotloden, 2 grafietpotloden HB, 4 inktcartridges, 1 puntenslijper, 1
liniaal, lengte: 17 cm, 1 tijdschema, 1 driehoek liniaal, 1 notitieblok, 2 paperclips, 2 gum
doppen, 1 UHU-lijmstift 8,2 g. Kleur: Rosé goud.",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 1,
                            CategoryId = 2,
                            ArticleName = "Miss Melody ",
                            Price = 36.95M,
                            Description = "Om bij weg te dromen! De paarse etui van Miss Melody
heeft een LED-functie: met een druk op de knop gaan de veren in de manen van Miss Melody en ook
sommige vuurvliegjes in het gras branden (circa 18 seconden). De drie aparte vakken zijn gevuld
met kleurpotloden, viltstiften, potloden, een liniaal, puntenslijper, gum, schaar, plakstift en
een geheim vakje.",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 2,
                            CategoryId = 2,
                            ArticleName = "Avocado",
                            Price = 36.95M,
                            Description = "Een etui om van te smullen: Hip Etui/Pennenzak met veel
ruimte voor je schrijfgerief en een avocadomotief om van te watertanden",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 3,
                            CategoryId = 3,
                            ArticleName = "Kangaro Schrift A4 Gelinieerd",
                            Price = 3.59M,
                            Description = "Dit zijn ideale schriften voor op kantoor, op school of
voor thuis. Dit Kangaro schrift is gelinieerd, spiraalgebonden en is voorzien van 80 pagina's.",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 2,
                            CategoryId = 3,
                            ArticleName = "Oxford School - Schoolschrift - A4 - Lijn",
                            Price = 3.59M,
                            Description = "Oxford Schoolschriften, voor al je huiswerk,
aantekeningen of samenvattingen! Een voordeelpak met daarin drie Oxford A4 schriften gelijnd.
Ideaal voor het maken van aantekeningen tijdens de les en je huiswerk. Dit pak bevat drie
schriften, twee zwarte en een turquoise. De schriften bevatten 36 vellen met een kantlijn en het
90 grams Optik Paper. Door het Oxford Optik Paper is het papier gegarandeerd van goede kwaliteit,
hierdoor kun je op beide zijdes van het papier schrijven zonder doordrukken.",
                            Score = 5M
                        },
                        new Article
                        {
                            BrandId = 4,
                            CategoryId = 4,
                            ArticleName = "Leitz Active Leitz WOW 4-rings ringband - blauw",
                            Price = 11.49M,
                            Description = "Zeer duurzaam en robuust. Ideaal voor in schooltassen.
Gepatenteerd SoftClick-mechanisme voor eenvoudig openen en veilig afsluiten. Binnenhoezen voor het
bewaren van papier, cd's en visitekaartjes. Ronde rug en pennenhouder voor gebruiksvriendelijkheid
onderweg. Capaciteit: 280 A4 - vellen 80 gram. 3 jaar garantie op het mechanisme.",
```

```
                            Score = 5M
                        }
                    );
                    context.SaveChanges();

                }
            }
        }
    }
```

- Pas het bestand appsettings.json in de root als volgt aan (= de naam van de nog te maken database aanpassen) :

```
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "SchoolShopContext":
"Server=(localdb)\\mssqllocaldb;Database=SchoolShop;Trusted_Connection=True;MultipleActiveResultSe
ts=true"
  }
```

- Open de Package Manager Console en voer onderstaande 2 instructies uit :

```
add-migration initialMigration

update-database
```

- Open het bestand Program.cs op de root en vervang de Main methode met onderstaande code :

```
        public static void Main(string[] args)
        {
            var host = CreateHostBuilder(args).Build();
            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
                    SeedData.Initialize(services);
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "Het seeden van de DB is mislukt.");
                }
            }

            host.Run();
        }
```

- Open het bestand Startup.cs op de root en voeg helemaal onderaan de methode Configure onderstaande 3 regels code toe :

```
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapRazorPages();
            });

            var cultureInfo = new CultureInfo("nl-BE");
            cultureInfo.NumberFormat.NumberDecimalSeparator = ",";
            CultureInfo.DefaultThreadCurrentCulture = cultureInfo;

        }
    }
}
```

- Run even je app zodat de seeding uitgevoerd wordt.
  Check de paginas /brands, /categories en /articles

Stap 6 : De _layout pagina aanpassen

Vervang de volledige code van de /Pages/Shared/_Layout.cshtml pagina door onderstaande code. Deze is identiek (mits wat kleine naamsveranderingen) aan de _Layout pagina van onze StovePalace applicatie :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Little School Shop</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css"
integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-
shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-page="/Index">The Little School Shop</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-
collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Articles/Index">Ons
aanbod</a>
                        </li>
                        <li class="nav-item dropdown" style="@ViewData["Configstyle"]">
                            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                                Configuratie
                            </a>
                            <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                                <a class="dropdown-item" href="/brands/index">Merken</a>
                                <a class="dropdown-item" href="/Categories/index">Categoriën</a>
                            </div>
                        </li>
                    </ul>
                    <ul class="nav navbar-nav navbar-right" style="@ViewData["Loginstyle"]">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
page="/Users/Register">Registreer</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Users/Login">Login</a>
                        </li>
                    </ul>
                    <ul class="nav navbar-nav navbar-right" style="@ViewData["Logoutstyle"]">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Users/Account">
                                <i class="fas fa-cog"></i> @ViewData["Email"]
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Users/Logout">
                                <i class="fas fa-sign-out-alt"></i> Logout
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Baskets/Index">
```

```
                            <i class="fas fa-shopping-cart"></i>
                            <span class="badge badge-dark">@ViewData["BasketCount"]</span>
                        </a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
</header>
<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>

<footer class="border-top footer text-muted">
    <div class="container">
        &copy; 2021 - The little school shop - <a asp-area="" asp-page="/Index">Home</a>
    </div>
</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Test je applicatie even uit.

Stap 7 : De /Pages/Index.cshtml pagina aanpassen (Home)

Maak in de map **wwwroot** een submap aan met de naam **images**.

Zoek op het internet een afbeelding die je kan gebruiken op je Home page, en bewaar deze in de zonet gemaakte map Images onder de haam schoolshop.jpg (indien je een andere naam kiest dan dien je straks de code in deze zin aan te passen).

De volledige inhoud van Index.cshtml vervang je met onderstaande code :

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}

<div class="text-center">
    <h1 class="display-3">Little School Shop</h1>
    <h1 class="display-4">Welkom</h1>
    <img class="img-fluid rounded" src="~/images/schoolshop.jpg" />
</div>
<p></p>
<div class="card">
    <div class="card-header">
        <h5>Contactgegevens</h5>
    </div>
    <div class="card-body">
        <p>Adres : Schoolstraat 1, 8000 Brugge</p>
        <p>Telefoon : 0497/66.77.88</p>
        <p>E-mail : info@littleschoolshop.be</p>
    </div>
</div>
```

In Index.cshtml.cs vervang je de IndexModel klasse met onderstaande code :

```
    public class IndexModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        public Availability Availability { get; set; }

        public IndexModel(SchoolShop.Data.SchoolShopContext context)
        {
            _context = context;
        }
        public void OnGet()
        {
            Availability = new Availability(_context, HttpContext);
        }
    }
```

Stap 8 : de Availablity klasse overal implementeren.

Voeg onderstaande lijn (het aanmaken van een property Availability van het type Availabity) (voeg deze telkens toe onder het readonly veld _context) :

```
        public Availability Availability { get; set; }
```

Toe aan de volgende pagina's :

- Articles
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Baskets
  - Index.cshtml.cs
- Brands
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Categories
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Users
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Orders
  - Details.cshtml.cs
  - Index.cshtml.cs

In al deze pagina's ga je eveneens in alle OnGet() of OnGetAsync() methoden, en indien aanwezig ook in de OnPost of OnPostAsync() methoden, deze eigenschap gaan instantieren (plaats deze telkens gemakshalve als eerste regel) :

```
            Availability = new Availability(_context, HttpContext);
```

Tenslotte ga je ook nog in al deze CSHTML pagina's bovenaan onderstaande code gaan toevoegen (net onder de @model instructie, +/- 3° regel) waar de ViewData waarden ingevuld worden die door de _Layout page worden uitgelezen (Title pas je uiteraard bij elke pagina aan met een aangepaste waarde) :

```
@{
    ViewData["Title"] = "…";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}
```

Stap 9: de Pages/Users/Register pagina aanmaken.

In de map Pages/Users voeg je een nieuwe Razor Page (Empty) toe met de naam Register.

De Register.cshtml pagina bevat volgende code (deze code is identiek aan wat we deden bij onze kachel app) :

```
@page
@model SchoolShop.Pages.Users.RegisterModel
@{
    ViewData["Title"] = "Registreren";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}

<h1>Registratieformulier</h1>

<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="User.Email" class="control-label"></label>
                <input asp-for="User.Email" class="form-control" />
                <span asp-validation-for="User.Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="User.Name" class="control-label"></label>
                <input asp-for="User.Name" class="form-control" />
                <span asp-validation-for="User.Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="User.FirstName" class="control-label"></label>
                <input asp-for="User.FirstName" class="form-control" />
                <span asp-validation-for="User.FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="User.Password" class="control-label"></label>
                <input type="password" asp-for="User.Password" class="form-control" />
                <span asp-validation-for="User.Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Registreer" class="btn btn-primary" />
            </div>
        </form>
        <p>
            <a class="btn btn-success" asp-page="/Users/Login">
                Reeds een account?  Ga naar de login pagina ...
            </a>
        </p>
    </div>
</div>
```

De Register.cshtml.cs code ziet er als volgt uit (deze code is identiek aan wat we deden bij onze kachel app) :

```
public class RegisterModel : PageModel
{
    private readonly SchoolShop.Data.SchoolShopContext _context;
    [BindProperty]
    public User User { get; set; }
    public Availability Availability { get; set; }
    public RegisterModel(SchoolShop.Data.SchoolShopContext context)
    {
        _context = context;
    }
    public IActionResult OnGet()
    {
        Availability = new Availability(_context, HttpContext);
        return Page();
    }
    public IActionResult OnPost()
    {
        Availability = new Availability(_context, HttpContext);
        if (!ModelState.IsValid)
        {
            return Page();
        }
        User.Password = Encoding.HashPassword(User.Password);
        _context.User.Add(User);
        try
        {
            _context.SaveChanges();
        }
        catch
        {
            return Page();
        }

        string IdCookie = Encoding.EncryptString(User.Id.ToString(), "P@sw00rd");
        CookieOptions cookieOptions = new CookieOptions();
        cookieOptions.Expires = new DateTimeOffset(DateTime.Now.AddDays(365));
        HttpContext.Response.Cookies.Append("UserId", IdCookie, cookieOptions);
        return RedirectToPage("../Articles/Index");

    }
}
```

Stap 10: de Pages/Users/Login pagina aanmaken.

Opnieuw wordt alle code die je hieronder kan terugvinden gewoon overgenomen van onze kachel app.

Maak in de map Pages/Users/ een nieuwe pagina aan met de naam Login (kies voor Razor Page Empty)

Login.cshtml :

```
@page
@model SchoolShop.Pages.Users.LoginModel
@{
    ViewData["Title"] = "Inloggen";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}
<h1>Login</h1>

<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label class="control-label">E-mail adres</label>
                <input asp-for="@Model.Email" class="form-control" />
            </div>
            <div class="form-group">
                <label class="control-label">Paswoord</label>
                <input type="password" asp-for="@Model.Password" class="form-control" />
            </div>
            <div class="form-group">
                <input type="submit" value="Login" class="btn btn-primary" />
            </div>
        </form>
        <p>
            <a class="btn btn-success" asp-page="/Users/Register">
                Nog geen account?  Registreer je dan snel ...
            </a>
        </p>
    </div>
</div>
```

Login.cshtml.cs :

```
    public class LoginModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;

        public LoginModel(SchoolShop.Data.SchoolShopContext context)
        {
            _context = context;
        }

        public string Email { get; set; }
        public string Password { get; set; }
        public Availability Availability { get; set; }
        public object Hashing { get; private set; }
```

```
        public IActionResult OnGet()
        {
            Availability = new Availability(_context, HttpContext);
            return Page();
        }
        public IActionResult OnPost(string email, string password)
        {
            Availability = new Availability(_context, HttpContext);
            Email = email;
            User user = _context.User.FirstOrDefault(u => u.Email == Email);
            if (user == null)
            {
                return RedirectToPage("./Login");
            }
            else
            {
                if (!Encoding.ValidatePassword(password, user.Password))
                {
                    return RedirectToPage("./Login");
                }
                string IdCookie = Encoding.EncryptString(user.Id.ToString(), "P@sw00rd");
                CookieOptions cookieOptions = new CookieOptions();
                cookieOptions.Expires = new DateTimeOffset(DateTime.Now.AddDays(7));
                HttpContext.Response.Cookies.Append("UserId", IdCookie, cookieOptions);
                return RedirectToPage("../Articles/Index");
            }
        }

    }
```

Stap 11: de Pages/Users/Logout pagina aanmaken.

Opnieuw wordt alle code die je hieronder kan terugvinden gewoon overgenomen van onze kachel app.

Maak in de map Pages/Users/ een nieuwe pagina aan met de naam Logout (kies voor Razor Page Empty)

Aan de Logout.cshtml pagina dien je niets te veranderen.

Aan de Logout.cshtml.cs pagina voeg je volgende code toe :

```
public class LogoutModel : PageModel
{
    public void OnGet()
    {
        CookieOptions cookieOptions = new CookieOptions();
        cookieOptions.Expires = new DateTimeOffset(DateTime.Now.AddDays(-1));
        HttpContext.Response.Cookies.Append("UserId", "", cookieOptions);
        Response.Redirect("../Articles/Index");
    }
}
```

Test je app even uit.
Registreer een gebruiker die geen admin rechten mag hebben en log terug uit.
Registreer een gebruiker die wel admin rechten mag hebben.

Open in Visual studio de SQL Server Object Explorer en open in de database SchoolShop de tabel User (View data) en wijzig bij de tweede gebruiker de waarde in de kolom IsAdmin (op True ipv False).

Stap 12: de pagina's beveiligen

In deze stap gaan we alle pagina's waar een gewone bezoeker geen toegang toe heeft beveiligen.
We gaan dit uiteraard doen m.b.v. de availability klasse die ons kan vertellen of een gebruiker wel degelijk ingelogd is en indien dat zo is of dat wel degelijk een beheerder is. Is dat niet zo, dan sturen we hem steevast naar de /Pages/Index pagina.

In al deze pagina's ga je, net onder het instantiëren van het Availability object in de OnGet/OnGetAsync en de OnPost/OnPostAsync methoden, die je eerder maakte, onderstaande code overnemen :

```
if (!Availability.IsAdmin)
{
    return RedirectToPage("../Articles/Index");
}
```

- Articles
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Edit.cshtml.cs
- Brands
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Categories
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Users
  - Create.cshtml.cs
  - Delete.cshtml.cs
  - Details.cshtml.cs
  - Edit.cshtml.cs
  - Index.cshtml.cs
- Orders
  - Details.cshtml.cs
  - Index.cshtml.cs

Opgepast : in sommige pagina's (zoals in de Brands/Index.cshtml.cs pagina) krijg je fouten op deze code. Dit komt omdat de OnGet/OnGetAsync methode hier VOID methoden zijn en dus geen return-value vragen. In deze situaties pas je de code dan als volgt aan :

```
if (!Availability.IsAdmin)
{
    RedirectToPage("../Articles/Index");
}
```

Stap 13: De Pages/Articles/Create : afbeelding

In deze pagina gaan we

- In de Pages/Articles/Create pagina de layout aanpassen
- De mogelijkheid voorzien een afbeelding toe te voegen

We beelden de volledige code van de pagina af (vervang dus je eigen code hiermee) en we markeren in het groen waar we wijzigingen hebben aangebracht t.o.v. de originele code :

```
@page
@model SchoolShop.Pages.Articles.CreateModel

@{
    ViewData["Title"] = "Een nieuw artikel";
    string ImagePath = "~/images/noimage.jpg";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}


<h1>Een nieuw artikel</h1>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post"    enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Merk</span>
                <div class="col-sm-10">
                    <select asp-for="Article.BrandId" class="form-control" asp-
items="ViewBag.BrandId"></select>
                </div>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Categorie</span>
                <div class="col-sm-10">
                    <select asp-for="Article.CategoryId" class="form-control" asp-
items="ViewBag.CategoryId"></select>
                </div>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Artikel</span>
                <div class="col-sm-10">
                    <input asp-for="Article.ArticleName" class="form-control" />
                </div><span asp-validation-for="Article.ArticleName" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Prijs</span>
                <div class="col-sm-10">
                    <input asp-for="Article.Price" class="form-control" />
                </div><span asp-validation-for="Article.Price" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Omschrijving</span>
                <div class="col-sm-10">
```

```
                    <textarea asp-for="Article.Description" class="form-control" rows="5"
></textarea>
                </div><span asp-validation-for="Article.Description" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Score</span>
                <div class="col-sm-10">
                    <input asp-for="Article.Score" class="form-control" value="5" />
                </div><span asp-validation-for="Article.Score" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label font-weight-bold">Afbeelding</label>
                <div class="col-sm-10">
                    <div class="custom-file">
                        <input asp-for="PhotoUpload" class="custom-file-input form-control" />
                        <label class="custom-file-label">Klik hier om foto te wijzigen</label>
                    </div>
                </div>
            </div>

            <div class="form-group row col-sm-4 offset-4">
                <img class="imageThumbnail" src="@ImagePath" asp-append-version="true" />
            </div>

            <div class="form-group">
                <div class="btn-group">
                    <button type="submit" class="btn  btn-success" title="Opslaan">
                        <i class="far fa-save"></i>
                    </button>
                    <a asp-page="Index" class="btn btn-danger" title="Annuleren">
                        <i class="fas fa-undo-alt"></i>
                    </a>
                </div>
            </div>

            @section Scripts {
                <script>
                    $(document).ready(function () {
                        $('.custom-file-input').on("change", function () {
                            var fileName = $(this).val().split("\\").pop();
                            $(this).next('.custom-file-label').html(fileName);
                        });
                    });
                </script>
            }
        </form>
    </div>
</div>
```

In Pages/Articles/Create.cshtml.cs dienen er uiteraard ook een aantal aanpassingen te gebeuren.
De volledige code van deze pagina :

```
    public class CreateModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        private readonly IWebHostEnvironment webhostEnvironment;
        public Availability Availability { get; set; }

        [BindProperty]
        public Article Article { get; set; }
        public IFormFile PhotoUpload { get; set; }
        public CreateModel(SchoolShop.Data.SchoolShopContext context,
            IWebHostEnvironment webhostEnvironment)
        {
            _context = context;
            this.webhostEnvironment = webhostEnvironment;
        }

        public IActionResult OnGet()
        {
            Availability = new Availability(_context, HttpContext);
            if (!Availability.IsAdmin)
            {
                return RedirectToPage("../Articles/Index");
```

```
            }
            ViewData["BrandId"] = new SelectList(_context.Brand, "Id", "BrandName");
            ViewData["CategoryId"] = new SelectList(_context.Category, "Id", "CategoryName");
            return Page();
        }



        // To protect from overposting attacks, see https://aka.ms/RazorPagesCRUD
        public async Task<IActionResult> OnPostAsync()
        {
            Availability = new Availability(_context, HttpContext);
            if (!Availability.IsAdmin)
            {
                return RedirectToPage("../Articles/Index");
            }
            if (!ModelState.IsValid)
            {
                return Page();
            }
            if (PhotoUpload != null)
            {
                if (Article.ImagePath != null)
                {
                    string filePath = Path.Combine(webhostEnvironment.WebRootPath, "images",
Article.ImagePath);
                    System.IO.File.Delete(filePath);
                }
                Article.ImagePath = ProcessUploadedFile();
            }
            _context.Article.Add(Article);
            await _context.SaveChangesAsync();

            return RedirectToPage("./Index");
        }
        private string ProcessUploadedFile()
        {
            string uniqueFileName = null;
            if (PhotoUpload != null)
            {
                string uploadFolder = Path.Combine(webhostEnvironment.WebRootPath, "images");
                uniqueFileName = Guid.NewGuid().ToString() + "_" + PhotoUpload.FileName;
                string filePath = Path.Combine(uploadFolder, uniqueFileName);
                using (var fileStream = new FileStream(filePath, FileMode.Create))
                {
                    PhotoUpload.CopyTo(fileStream);
                }
            }

            return uniqueFileName;
        }
    }
```

Stap 14: De Pages/Articles/Edit : afbeelding

In deze pagina gaan we

- In de Pages/Articles/Edit pagina de layout aanpassen
- De mogelijkheid voorzien een afbeelding te wijzigen

We beelden de volledige code van de pagina af (vervang dus je eigen code hiermee) en we markeren in het groen waar we wijzigingen hebben aangebracht t.o.v. de originele code :

```
@page  "{id:int?}"
@model SchoolShop.Pages.Articles.EditModel

@{
    ViewData["Title"] = "Artikel wijzigen";
    string ImagePath = "~/images/" + (Model.Article.ImagePath ?? "noimage.jpg");
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}


<h1>Een artikel wijzigen</h1>
<hr />
<div class="row">
    <div class="col-md-10">
        <form method="post" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Article.Id" />
            <input type="hidden" asp-for="Article.ImagePath" />
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Merk</span>
                <div class="col-sm-10">
                    <select asp-for="Article.BrandId" class="form-control" asp-
items="ViewBag.BrandId"></select>
                </div>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Categorie</span>
                <div class="col-sm-10">
                    <select asp-for="Article.CategoryId" class="form-control" asp-
items="ViewBag.CategoryId"></select>
                </div>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Artikel</span>
                <div class="col-sm-10">
                    <input asp-for="Article.ArticleName" class="form-control" />
                    <span asp-validation-for="Article.ArticleName" class="text-danger"></span>
                </div>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Prijs</span>
                <div class="col-sm-10">
                    <input asp-for="Article.Price" class="form-control" />
                    <span asp-validation-for="Article.Price" class="text-danger"></span>
                </div>
            </div>
            <div class="form-group row">
```

```
                <span class="col-sm-2 col-form-label font-weight-bold">Omschrijving</span>
                <div class="col-sm-10">
                    <textarea asp-for="Article.Description" class="form-control"
rows="5"></textarea>
                    <span asp-validation-for="Article.Description" class="text-danger"></span>
                </div>
            </div>
            <div class="form-group row">
                <span class="col-sm-2 col-form-label font-weight-bold">Score</span>
                <div class="col-sm-10">
                    <input asp-for="Article.Score" class="form-control" />
                    <span asp-validation-for="Article.Score" class="text-danger"></span>
                </div>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label font-weight-bold">Afbeelding</label>
                <div class="col-sm-10">
                    <div class="custom-file">
                        <input asp-for="PhotoUpload" class="custom-file-input form-control" />
                        <label class="custom-file-label">Klik hier om foto te wijzigen</label>
                    </div>
                </div>
            </div>

            <div class="form-group row col-sm-4 offset-6">
                <img class="imageThumbnail" src="@ImagePath" asp-append-version="true" />
            </div>

            <div class="form-group">
                <div class="btn-group">
                    <button type="submit" class="btn  btn-success" title="Opslaan">
                        <i class="far fa-save"></i>
                    </button>
                    <a asp-page="Index" class="btn btn-danger" title="Annuleren">
                        <i class="fas fa-undo-alt"></i>
                    </a>
                </div>
            </div>

            @section Scripts {
                <script>
                    $(document).ready(function () {
                        $('.custom-file-input').on("change", function () {
                            var fileName = $(this).val().split("\\").pop();
                            $(this).next('.custom-file-label').html(fileName);
                        });
                    });
                </script>
            }
        </form>
    </div>
</div>
```

In Edit.cshtml.cs vervang je de volledige code met onderstaande :

```
    public class EditModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        private readonly IWebHostEnvironment webhostEnvironment;
        public Availability Availability { get; set; }
        public EditModel(SchoolShop.Data.SchoolShopContext context,
            IWebHostEnvironment webhostEnvironment)
        {
            _context = context;
            this.webhostEnvironment = webhostEnvironment;
        }

        [BindProperty]
        public Article Article { get; set; }
        [BindProperty]
        public IFormFile PhotoUpload { get; set; }
        public async Task<IActionResult> OnGetAsync(int? id)
        {
            Availability = new Availability(_context, HttpContext);
            if (!Availability.IsAdmin)
```

```
            {
                return RedirectToPage("../Articles/Index");
            }
            if (id == null)
            {
                return NotFound();
            }

            Article = await _context.Article
                .Include(a => a.Brand)
                .Include(a => a.Category).FirstOrDefaultAsync(m => m.Id == id);

            if (Article == null)
            {
                return NotFound();
            }
           ViewData["BrandId"] = new SelectList(_context.Brand, "Id", "BrandName");
           ViewData["CategoryId"] = new SelectList(_context.Category, "Id", "CategoryName");
            return Page();
        }

        // To protect from overposting attacks, enable the specific properties you want to bind
to.
        // For more details, see https://aka.ms/RazorPagesCRUD.
        public async Task<IActionResult> OnPostAsync()
        {
            Availability = new Availability(_context, HttpContext);
            if (!Availability.IsAdmin)
            {
                return RedirectToPage("../Articles/Index");
            }
            if (!ModelState.IsValid)
            {
                return Page();
            }
            if (PhotoUpload != null)
            {
                if (Article.ImagePath != null)
                {
                    string filePath = Path.Combine(webhostEnvironment.WebRootPath, "images",
Article.ImagePath);
                    System.IO.File.Delete(filePath);
                }
                Article.ImagePath = ProcessUploadedFile();
            }
            _context.Attach(Article).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ArticleExists(Article.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return RedirectToPage("./Index");
        }

        private bool ArticleExists(int id)
        {
            return _context.Article.Any(e => e.Id == id);
        }
        private string ProcessUploadedFile()
        {
            string uniqueFileName = null;
            if (PhotoUpload != null)
            {
                string uploadFolder = Path.Combine(webhostEnvironment.WebRootPath, "images");
```

```
                uniqueFileName = Guid.NewGuid().ToString() + "_" + PhotoUpload.FileName;
                string filePath = Path.Combine(uploadFolder, uniqueFileName);
                using (var fileStream = new FileStream(filePath, FileMode.Create))
                {
                    PhotoUpload.CopyTo(fileStream);
                }
            }
            return uniqueFileName;
        }
    }
```

Stap 15: De Pages/Articles/Details : afbeelding

In deze pagina gaan we

- In de Pages/Articles/Details pagina de layout aanpassen alsook de afbeelding tonen

Opmerkingen :

- We gaan verderop deze pagina nog een keer aanpassen omwille van de paginatie
- Het item Score werken we voorlopig nog niet uit (we tonen gewoon een getal) : we doen dit helemaal op het einde

Details.cshtml :

```
@page
@model SchoolShop.Pages.Articles.DetailsModel

@{
    ViewData["Title"] = "Details artikel";
    string imagePath = "~/images/" + (Model.Article.ImagePath ?? "noimage.jpg");
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}
<h1>@Html.DisplayFor(model => model.Article.ArticleName)</h1>
<h2>@Html.DisplayFor(model => model.Article.Brand.BrandName)</h2>
<h2>@Html.DisplayFor(model => model.Article.Category.CategoryName)</h2>
<hr />
<div class="container-fluid">
    <img class="imageThumbnail" src="@imagePath" asp-append-version="true" />
</div>
<hr />
<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Prijs</span>
    <div class="col-sm-10">
        @Html.DisplayFor(model => model.Article.Price)
    </div>
</div>
<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Omschrijving</span>
    <div class="col-sm-10">
        @Html.DisplayFor(model => model.Article.Description)
    </div>
</div>
<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Score</span>
    <div class="col-sm-10">
        @Html.DisplayFor(model => model.Article.Score)
    </div>
</div>


<div>
    <div class="btn-group">
        <a style="@Model.Availability.ConfigButtonStyle" asp-page="./Edit" asp-route-
id="@Model.Article.Id" class="btn  btn-warning" title="Wijzigen">
            <i class="fas fa-pencil-alt"></i>
        </a>
```

```
        <a asp-page="./Index" class="btn btn-light">
            Terug naar overzicht
        </a>
    </div>
</div>
```

Details.cshtml.cs :

```
    public class DetailsModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        public Availability Availability { get; set; }
        public DetailsModel(SchoolShop.Data.SchoolShopContext context)
        {
            _context = context;
        }
        public Article Article { get; set; }
        public async Task<IActionResult> OnGetAsync(int? id)
        {
            Availability = new Availability(_context, HttpContext);
            if (id == null)
            {
                return NotFound();
            }

            Article = await _context.Article
                .Include(a => a.Brand)
                .Include(a => a.Category).FirstOrDefaultAsync(m => m.Id == id);

            if (Article == null)
            {
                return NotFound();
            }
            return Page();
        }
    }
```

Stap 16: De Pages/Articles/Index : layout (deel 1)

Deze pagina is uiteraard belangrijk … hier zal onze bezoeker alle artikels te zien krijgen.
We gaan deze pagina in een aantal stappen aanpassen :

- We zorgen dat bij elk artikel de afbeelding getoond wordt
- We zorgen er voor dat er met paginatie kan gewerkt worden
- We zorgen er voor dat er gefilterd kan worden

In dit eerste deel passen we de layout van de pagina aan.

Voeg eerst aan wwwroot/css/sites.cs onderstaande css klasse toe :

```
.longtext {
    overflow: hidden;
    white-space: nowrap; /* Don't forget this one */
    text-overflow: ellipsis;
}
```

/Articles/Index

```
@page
@model SchoolShop.Pages.Articles.IndexModel

@{
    ViewData["Title"] = "Alle artikels";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}
<form method="post">

    <h1>Ons aanbod</h1>

    <p style="margin-top:10px;">
        <a style="@Model.Availability.ConfigButtonStyle" asp-page="Create" class="btn btn-
primary"><i class="fa fa-plus"></i> Nieuw artikel</a>
    </p>
    <hr />
    <div class="row">
        @foreach (var item in Model.Article)
        {
            string imagePath = "~/images/" + (item.ImagePath ?? "noimage.jpg");

            <div class="col-6 col-lg-4 p-3">
                <div class="card">
                    <div class="card-header">
                        <h3 class="longtext">@Html.DisplayFor(modelItem => item.ArticleName)</h3>
                        <h4>@Html.DisplayFor(modelItem => item.Category.CategoryName)</h4>
                    </div>
                    <a asp-page="./Details" asp-route-id="@item.Id" class="btn">
                        <div class="card-body text-left" style="min-height:400px;">
                            <h5>Merk : @Html.DisplayFor(modelItem => item.Brand.BrandName)</h5>
                            <h5>Prijs : @Html.DisplayFor(modelItem => item.Price)</h5>
                            <h5>Score : @Html.DisplayFor(modelItem => item.Score)</h5>
```

```
                            <hr />
                            <div class="text-center">
                                <img class="imageThumbnail" src="@imagePath" asp-append-
version="true" />
                            </div>
                        </div>
                    </a>
                    <div class="card-footer">
                        <div class="btn-group btn-group-sm  float-right">
                            <a style="@Model.Availability.ConfigButtonStyle" asp-page="./Edit"
asp-route-id="@item.Id" class="btn btn-sm btn-warning" title="Wijzig"><i class="fas fa-pencil-
alt"></i></a>
                            <a style="@Model.Availability.ConfigButtonStyle" asp-page="./Delete"
asp-route-id="@item.Id" class="btn btn-sm btn-danger" title="Wissen"><i class="fas fa-trash-
alt"></i></a>
                            <a asp-page="./Basket" asp-route-id="@item.Id" class="btn btn-sm btn-
success" title="Toevoegen aan winkelmandje"><i class="fas fa-cart-plus"></i></a>
                        </div>
                    </div>
                </div>
            </div>

        }
        </div>


    </form>
```

Stap 17: De Pages/Articles/Index : filters (deel 2)

We voorzien nu in onze Index pagina de mogelijkheid om te filteren op merk en/of categorie

Index.cshtml.cs

- We voorzien 2 eigenschappen van het type List<SelectedListItem> om de filter comboboxen aan te maken
- We voorzien 2 eigenschappen om de geselecteerde filterwaarden (id van Brand en id van Category) op te vangen
- We veranderen de OnGetAsync methode in een OnGet methode en we roepen in deze methode de nog te maken PopulateCollection methode op.
- In de PopulateCollection methode gaan we :
    - M.b.v. IQueryable de lijst met artikels gaan opbouwen volgens de eventueel ingestelde filters die we achteraf converteren naar een list
    - De comboboxen vullen we op met de beschikbare merken en categorieën
- We voegen een OnPost methode toe die opnieuw de PopulateCollection methode gaat oproepen.

```
    public class IndexModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        public Availability Availability { get; set; }
        public IndexModel(SchoolShop.Data.SchoolShopContext context)
        {
            _context = context;
        }

        public IList<Article> Article { get;set; }
        public List<SelectListItem> AvailableBrands { get; set; }
        public List<SelectListItem> AvailableCategories { get; set; }
        public int? SelectedBrandId { get; set; }
        public int? SelectedCategoryId { get; set; }
        public void  OnGet()
        {
            Availability = new Availability(_context, HttpContext);
            PopulateCollections(null, null);

        }
        private void PopulateCollections(int? brandFilter, int? categoryFilter)
        {
            // artikels uit database halen volgens ingestelde filters
            IQueryable<Article> artquery = _context.Article
                .Include(b => b.Brand)
                .Include(c => c.Category)
                .OrderBy(a => a.Category.CategoryName)
                .ThenBy(a => a.Price);
            if (brandFilter != null && categoryFilter == null)
            {
                artquery = artquery.Where(b => b.BrandId.Equals(brandFilter));
            }
            if (brandFilter == null && categoryFilter != null)
            {
                artquery = artquery.Where(b => b.CategoryId.Equals(categoryFilter));
            }
            if (brandFilter != null && categoryFilter != null)
            {
                artquery = artquery.Where(b => b.BrandId.Equals(brandFilter) &&
b.CategoryId.Equals(categoryFilter));
            }
            Article = artquery.ToList();

            // merken ophalen om combobox te vullen
            AvailableBrands = _context.Brand.Select(a =>
                new SelectListItem
```

```
                {
                    Value = a.Id.ToString(),
                    Text = a.BrandName
                }).OrderBy(b => b.Text).ToList();
            AvailableBrands.Insert(0, new SelectListItem()
            {
                Value = null,
                Text = "--- Alle merken ---"
            });

            // categorien ophalen om combobox te vullen
            AvailableCategories = _context.Category.Select(a =>
                new SelectListItem
                {
                    Value = a.Id.ToString(),
                    Text = a.CategoryName
                }).OrderBy(b => b.Text).ToList();
            AvailableCategories.Insert(0, new SelectListItem()
            {
                Value = null,
                Text = "--- Alle categoriën ---"
            });

        }

        public void OnPost(int? brandFilter, int? categoryFilter)
        {
            Availability = new Availability(_context, HttpContext);
            SelectedBrandId = brandFilter;
            SelectedCategoryId = categoryFilter;
            PopulateCollections(brandFilter, categoryFilter);
        }
    }
```

Index.cshtml

Plaats onderstaande code net onder <h1>Ons aanbod</h1> (+/- regel 25)

```
    <div class="row">
        <div class="col-md-4">
            <p>Filter merken : </p>
            <select name="brandFilter" class="form-control"
                    asp-for="SelectedBrandId"
                    asp-items="Model.AvailableBrands" onchange="this.form.submit()"></select>
        </div>
        <div class="col-md-4">
            <p>Filter categorieën : </p>
            <select name="categoryFilter" class="form-control"
                    asp-for="SelectedCategoryId"
                    asp-items="Model.AvailableCategories" onchange="this.form.submit()"></select>
        </div>
    </div>
```

Stap 18: De Pages/Articles/Index : paginatie (deel 3)

Index.cshtml.cs

- We voegen een nieuwe private variabele toe (ItemsPerPage). Deze waarde zal bepalen hoeveel artikels we per pagina gaan afbeelden.
- We voegen een nieuwe eigenschap van het type Pagination toe (uit onze helper klassen)
- We passen de OnGet methode aan zodat deze een argument (het paginanummer) kan ontvangen
- We passen de PopulateCollections aan zodat deze dat paginanummer kan ontvangen en dat de klasse Pagination gebruikt wordt in het filteren van de gegevens.
- We passen de OnPost methode aan zodat deze een extra argument (het paginanummer) kan ontvangen

```
    public class IndexModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        private readonly int ItemsPerPage = 3;

        public Availability Availability { get; set; }
        public Pagination Pagination { get; private set; }
        public IList<Article> Article { get;set; }
        public List<SelectListItem> AvailableBrands { get; set; }
        public List<SelectListItem> AvailableCategories { get; set; }
        public int? SelectedBrandId { get; set; }
        public int? SelectedCategoryId { get; set; }
        public IndexModel(SchoolShop.Data.SchoolShopContext context)
        {
            _context = context;
        }
        public void  OnGet(int? pageIndex)
        {
            Availability = new Availability(_context, HttpContext);
            PopulateCollections(null, null, pageIndex);

        }
        private void PopulateCollections(int? brandFilter, int? categoryFilter, int? pageIndex)
        {
            // artikels uit database halen volgens ingestelde filters
            IQueryable<Article> artquery = _context.Article
                .Include(b => b.Brand)
                .Include(c => c.Category)
                .OrderBy(a => a.Category.CategoryName)
                .ThenBy(a => a.Price);
            if (brandFilter != null && categoryFilter == null)
            {
                artquery = artquery.Where(b => b.BrandId.Equals(brandFilter));
            }
            if (brandFilter == null && categoryFilter != null)
            {
                artquery = artquery.Where(b => b.CategoryId.Equals(categoryFilter));
            }
            if (brandFilter != null && categoryFilter != null)
            {
                artquery = artquery.Where(b => b.BrandId.Equals(brandFilter) &&
b.CategoryId.Equals(categoryFilter));
            }

            // paginatie regelen
            Pagination = new Pagination(artquery, pageIndex, ItemsPerPage);
            if (pageIndex > Pagination.LastPageIndex)
            {
                Pagination.PageIndex = 0;
                Pagination.FirstObjectIndex = 0;
            }
```

```
            // IQueryable filteren op artikels van de geselecteerde
            // pagina en omzetten in een list

            Article = artquery.Skip(Pagination.FirstObjectIndex)
                              .Take(ItemsPerPage)
                              .ToList();

            // merken ophalen om combobox te vullen
            AvailableBrands = _context.Brand.Select(a =>
                new SelectListItem
                {
                    Value = a.Id.ToString(),
                    Text = a.BrandName
                }).OrderBy(b => b.Text).ToList();
            AvailableBrands.Insert(0, new SelectListItem()
            {
                Value = null,
                Text = "--- Alle merken ---"
            });

            // categorien ophalen om combobox te vullen
            AvailableCategories = _context.Category.Select(a =>
                new SelectListItem
                {
                    Value = a.Id.ToString(),
                    Text = a.CategoryName
                }).OrderBy(b => b.Text).ToList();
            AvailableCategories.Insert(0, new SelectListItem()
            {
                Value = null,
                Text = "--- Alle categoriën ---"
            });

        }

        public void OnPost(int? brandFilter, int? categoryFilter,int? pageIndex)
        {
            Availability = new Availability(_context, HttpContext);
            SelectedBrandId = brandFilter;
            SelectedCategoryId = categoryFilter;
            PopulateCollections(brandFilter, categoryFilter, pageIndex);
        }
    }
```

Index.cshtml

Voeg onderstaande code toe net onder de CREATE button (+/- regel 44)

```
    <hr />
    <button type="submit" class="btn btn-light"
            formaction="?pageIndex=@(Model.Pagination.FirstPageIndex)">
        &lt;&lt;
    </button>
    <button type="submit" class="btn btn-light"
            formaction="?pageIndex=@(Model.Pagination.PreviousPageIndex)">
        &lt;
    </button>

    @foreach (int item in Model.Pagination.PageIndexes)
    {
        if (item == Model.Pagination.PageIndex)
        {
            <button type="submit" class="btn btn-primary"
                    formaction="?pageIndex=@item">
                @((item + 1).ToString())
            </button>
        }
        else
        {
            <button type="submit" class="btn btn-light"
                    formaction="?pageIndex=@item">
                @((item + 1).ToString())
            </button>
```

```
        }
    }
    <button type="submit" class="btn btn-light"
            formaction="?pageIndex=@(Model.Pagination.NextPageIndex)">
        &gt;
    </button>
    <button type="submit" class="btn btn-light"
            formaction="?pageIndex=@(Model.Pagination.LastPageIndex)">
        &gt;&gt;
    </button>
```

Stap 19: De Pages/Articles/Details : navigatie (deel 3)

We zorgen er nu ook voor dat in de details page kan genavigeerd worden zodat niet telkens dient teruggekeerd te worden naar de Index pagina om een volgend artikel in detail te bekijken.

Details.cshtml.cs

- We voegen 2 eigenschappen toe waarin we het ID van het voorgaande en het volgende artikel gaan bijhouden
- We voegen een eigenschap toe (List<Articles>) waarin we alle ID's van alle artikels zullen bijhouden zodat we te weten kunnen komen wat het Id is van de voorgaande artikel en van het volgende artikel
- In de OnGetAsync methode gaan we deze List vullen met alle artikels zodat we de betrokken ID's kunnen opzoeken

```
public class DetailsModel : PageModel
{
    private readonly SchoolShop.Data.SchoolShopContext _context;
    public Availability Availability { get; set; }
    public Article Article { get; set; }
    public int? PreviousId { get; set; }
    public int? NextId { get; set; }
    private List<Article> Articles { get; set; }
    public DetailsModel(SchoolShop.Data.SchoolShopContext context)
    {
        _context = context;
    }

    public async Task<IActionResult> OnGetAsync(int? id)
    {
        Availability = new Availability(_context, HttpContext);
        if (id == null)
        {
            return NotFound();
        }

        Article = await _context.Article
            .Include(a => a.Brand)
            .Include(a => a.Category).FirstOrDefaultAsync(m => m.Id == id);

        if (Article == null)
        {
            return NotFound();
        }

        // vraag alle artikels op
        // sorteer ze op dezelfde manier als de index pagina
        Articles = _context.Article
            .Include(x => x.Category)
            .OrderBy(x=> x.Category.CategoryName)
            .ThenBy(x => x.Price)
            .ToList();

        PreviousId = null;
        NextId = null;
        // ga op zoek naar het id van vorige en volgende
        for (int i = 0; i < Articles.Count; i++)
        {
            if (((Article)Articles[i]).Id == id)
            {
                if (i > 0)
                    PreviousId = ((Article)Articles[i - 1]).Id;
                if (i < Articles.Count - 1)
                    NextId = ((Article)Articles[i + 1]).Id;
                break;
            }
        }
        if (PreviousId == null) PreviousId = id;
```

```
            if (NextId == null) NextId = id;

            return Page();
        }
    }
```

Details.cshtml

```
@page
@model SchoolShop.Pages.Articles.DetailsModel

@{
    ViewData["Title"] = "Details artikel";
    string imagePath = "~/images/" + (Model.Article.ImagePath ?? "noimage.jpg");
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}
<h1>@Html.DisplayFor(model => model.Article.ArticleName)</h1>
<h2>@Html.DisplayFor(model => model.Article.Brand.BrandName)</h2>
<h2>@Html.DisplayFor(model => model.Article.Category.CategoryName)</h2>
<hr />
<div class="btn-group">
    <a asp-page="./Details" asp-route-id="@Model.PreviousId" class="btn btn-light">&lt; Vorige</a>
    <a asp-page="./Index" class="btn btn-light">
        Terug naar overzicht
    </a>
    <a asp-page="./Details" asp-route-id="@Model.NextId" class="btn btn-light">Volgende &gt;</a>
</div>
<hr />
<div class="container-fluid">
    <div class="row">
        <div class="col-10">
            <img class="imageThumbnail" src="@imagePath" asp-append-version="true" />
        </div>
        <div class="col-2">
            <a asp-page="./Basket" asp-route-id="@Model.Article.Id" class="btn btn-lg btn-success"
title="Toevoegen aan winkelmandje"><i class="fas fa-cart-plus"></i></a>
        </div>
    </div>
</div>

<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Prijs</span>
    <div class="col-sm-10">
        @Html.DisplayFor(model => model.Article.Price)
    </div>
</div>
<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Omschrijving</span>
    <div class="col-sm-10">
        @Html.DisplayFor(model => model.Article.Description)
    </div>
</div>
<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Score</span>
    <div class="col-sm-10">
        @Html.DisplayFor(model => model.Article.Score)
    </div>
</div>
```

```
<div>
    <div class="btn-group">
        <a style="@Model.Availability.ConfigButtonStyle" asp-page="./Edit" asp-route-
id="@Model.Article.Id" class="btn  btn-warning" title="Wijzigen">
            <i class="fas fa-pencil-alt"></i>
        </a>

    </div>
</div>
```

Stap 19: De Pages/Categories/Edit

We voorzien in de pagina Pages/Categories/Edit eveneens de mogelijkheid om algemene afbeeldingen per categorie toe te voegen.

Pages/Categories/Edit.cshtml

```
@page
@model SchoolShop.Pages.Categories.EditModel

@{
    ViewData["Title"] = "Categorie wijzigen";
    string ImagePath = "~/images/" + (Model.Category.ImagePath ?? "noimage.jpg");
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}


<h1>Edit</h1>

<h4>Category</h4>
<hr />
<div class="row">
    <div class="col-md-10">
        <form method="post"  enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Category.Id" />
            <input type="hidden" asp-for="Category.ImagePath" />
            <div class="form-group">
                <label asp-for="Category.CategoryName" class="control-label"></label>
                <input asp-for="Category.CategoryName" class="form-control" />
                <span asp-validation-for="Category.CategoryName" class="text-danger"></span>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label font-weight-bold">Afbeelding</label>
                <div class="col-sm-10">
                    <div class="custom-file">
                        <input asp-for="PhotoUpload" class="custom-file-input form-control" />
                        <label class="custom-file-label">Klik hier om foto te wijzigen</label>
                    </div>
                </div>
            </div>
            <div class="form-group row col-sm-4 offset-6">
                <img class="imageThumbnail" src="@ImagePath" asp-append-version="true" />
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
            @section Scripts {
                <script>
                    $(document).ready(function () {
                        $('.custom-file-input').on("change", function () {
                            var fileName = $(this).val().split("\\").pop();
                            $(this).next('.custom-file-label').html(fileName);
                        });
                    });
```

```
                </script>
            }
        </form>
    </div>
</div>

<div>
    <a asp-page="./Index">Back to List</a>
</div>
```

Pages/Categories/Edit.cshtml.cs

```
    public class EditModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        private readonly IWebHostEnvironment webhostEnvironment;
        public Availability Availability { get; set; }
        public EditModel(SchoolShop.Data.SchoolShopContext context,
            IWebHostEnvironment webhostEnvironment)
        {
            _context = context;
            this.webhostEnvironment = webhostEnvironment;
        }

        [BindProperty]
        public Category Category { get; set; }
        [BindProperty]
        public IFormFile PhotoUpload { get; set; }
        public async Task<IActionResult> OnGetAsync(int? id)
        {
            Availability = new Availability(_context, HttpContext);
            if (!Availability.IsAdmin)
            {
                return RedirectToPage("../Articles/Index");
            }
            if (id == null)
            {
                return NotFound();
            }

            Category = await _context.Category.FirstOrDefaultAsync(m => m.Id == id);

            if (Category == null)
            {
                return NotFound();
            }
            return Page();
        }

        // To protect from overposting attacks, enable the specific properties you want to bind
to.
        // For more details, see https://aka.ms/RazorPagesCRUD.
        public async Task<IActionResult> OnPostAsync()
        {
            Availability = new Availability(_context, HttpContext);
            if (!Availability.IsAdmin)
            {
                return RedirectToPage("../Articles/Index");
            }

            if (!ModelState.IsValid)
            {
                return Page();
            }
            if (PhotoUpload != null)
            {
                if (Category.ImagePath != null)
                {
                    string filePath = Path.Combine(webhostEnvironment.WebRootPath, "images",
Category.ImagePath);
                    System.IO.File.Delete(filePath);
                }
                Category.ImagePath = ProcessUploadedFile();
            }
```

```
            _context.Attach(Category).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!CategoryExists(Category.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return RedirectToPage("./Index");
        }

        private bool CategoryExists(int id)
        {
            return _context.Category.Any(e => e.Id == id);
        }
        private string ProcessUploadedFile()
        {
            string uniqueFileName = null;
            if (PhotoUpload != null)
            {
                string uploadFolder = Path.Combine(webhostEnvironment.WebRootPath, "images");
                uniqueFileName = Guid.NewGuid().ToString() + "_" + PhotoUpload.FileName;
                string filePath = Path.Combine(uploadFolder, uniqueFileName);
                using (var fileStream = new FileStream(filePath, FileMode.Create))
                {
                    PhotoUpload.CopyTo(fileStream);
                }
            }
            return uniqueFileName;
        }
    }
```

Stap 20: een nieuwe pagina Pages/Categories/Filter

We gaan onze bezoeker de kans geven om via een nieuw te maken pagina de lijst met al onze artikels meteen te beperken tot één bepaalde categorie.
We maken eerst deze pagina aan, en gaan er straks voor zorgen dat deze via de _Layout pagina beschikbaar zal zijn.

- Klik met de rechtermuisknop op de map /Pages/Categories
- Voeg een nieuwe – Empty – Razor Page toe aan deze map en geef deze de naam Filter

De Filter.cshtml pagina :

```
@page
@model SchoolShop.Pages.Categories.FilterModel
@{
    ViewData["Title"] = "Alle artikels";
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
    ViewData["Logoutstyle"] = "visibility: hidden; ";
    ViewData["Configstyle"] = "visibility: hidden; ";
    if (!string.IsNullOrEmpty(Model.Availability.Email))
    {
        ViewData["Email"] = Model.Availability.Email;
        ViewData["BasketCount"] = Model.Availability.BasketCount;
        ViewData["Loginstyle"] = "visibility: hidden; ";
        ViewData["Logoutstyle"] = "visibility: visible; ";
        if (Model.Availability.IsAdmin)
        {
            ViewData["Configstyle"] = "visibility: visible; ";
        }
    }
}

<h1>Categorieën</h1>
<div class="row">
    @foreach (var item in Model.Categories)
    {
        string imagePath = "~/images/" + (item.ImagePath ?? "noimage.jpg");

        <div class="col-6 col-lg-4 p-3">
            <div class="card">
                <div class="card-header">
                    <h3>@Html.DisplayFor(modelItem => item.CategoryName)</h3>
                </div>
                <a asp-page="../Articles/Index" asp-route-categoryFilter="@item.Id" class="btn">
                    <div class="card-body" style="min-height:300px;">
                        <div class="text-center">
                            <img class="imageThumbnail" src="@imagePath" asp-append-version="true"
/>
                        </div>
                    </div>
                </a>

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

De filter.cshtml.cs pagina :

```
    public class FilterModel : PageModel
    {
        private readonly SchoolShop.Data.SchoolShopContext _context;
        public Availability Availability { get; set; }
        public IList<Category> Categories { get; set; }
```

```
        public FilterModel(SchoolShop.Data.SchoolShopContext context)
        {
            _context = context;
        }

        public void OnGet()
        {
            Availability = new Availability(_context, HttpContext);
            IQueryable<Category> catquery = _context.Category
                .OrderBy(a => a.CategoryName);
            Categories = catquery.ToList();
        }
    }
```

Wanneer op een categorie wordt geklikt zullen we de /Pages/Articles/Index pagina openen en er meteen voor zorgen dat de categorie filter ingesteld wordt op dezelfde waarde. We moeten hiervoor in de OnGet() methode van deze pagina nog een kleine aanpassing doen :

In /Pages/Articles/Index.cshtml

```
        public void  OnGet(int? pageIndex, int? categoryFilter)
        {
            Availability = new Availability(_context, HttpContext);
            SelectedCategoryId = categoryFilter;
            PopulateCollections(null, categoryFilter, pageIndex);
        }
```

Tenslotte passen we nog onze /Pages/Shared/_Layout.cshtml aan. Plak onderstaande code net onder de link die naar “Ons aanbod” verwijst (+/- regel 28) :

```
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
page="/Categories/Filter">Categorieën</a>
                        </li>
```

Mogelijks valt je menu systeem nu wat door elkaar … Indien dat zo is, verander dan op regel 14 de klasse van de DIV van “container” naar “conainer-fluid”.

Stap 21: de overige pagina’s bijwerken

Je gaat nu zelf alle overige pagina’s gaan bijwerken wat layout betreft.
Het gaat om deze pagina’s :

- Articles
  - Delete
- Brands
  - Create
  - Delete
  - Details
  - Edit
  - Index
- Categories
  - Create
  - Delete
  - Details
  - Edit
  - Index

Stap 22: sterretjes plaatsen i.p.v. scores

Opgepast : we gaan enkel de huidige score die een artikel heeft afbeelden. De bezoeker een score laten selecteren doen we later.

/Pages/Articles/Details.cshtml

- Voeg onder de variabele imagePath een nieuwe string variabele toe die de score uit het model (Model.Article.Score) opvraagt en afrond (afronden kan je met aan de ToString() methode een aangepast formaat – 0 – te geven) :

```
@{
    ViewData["Title"] = "Details artikel";
    string imagePath = "~/images/" + (Model.Article.ImagePath ?? "noimage.jpg");
    string roundScore = Model.Article.Score.ToString("0");
    ViewData["Email"] = "";
    ViewData["Loginstyle"] = "visibility: visible; ";
```

- Op (+/-) regel 59 vervang je de weergave van de score (nu een gewoon getal) door onderstaande code :

```
<div class="form-group row">
    <span class="col-sm-2 col-form-label font-weight-bold">Score</span>
    <div class="col-sm-10">
        @for (int i = 0; i < Int32.Parse(roundScore); i++)
        {
            <i class="fas fa-star" style="color:gold;"></i>
        }
        @for (int i = Int32.Parse(roundScore); i < 5; i++)
        {
            <i class="far fa-star" style="color:gold;"></i>
        }
    </div>
</div>
```

/Pages/Articles/Index.cshtml

- In de Index pagina tonen we meerdere artikels : we realiseren dit door een foreach lus te gebruiken.
  Omdat de score van elk artikel verschillend kan zijn moeten we onze variabele “roundScore” nu uiteraard aanmaken binnen deze foreach lus.
  Op +/- regel 84 plaats je onderstaande declaratie van “roundScore” :

```
        @foreach (var item in Model.Article)
        {
            string imagePath = "~/images/" + (item.ImagePath ?? "noimage.jpg");
            string roundScore = item.Score.ToString("0");

            <div class="col-6 col-lg-4 p-3">
```

- In de DIV met bootstrap klasse card-body vervang je het afbeelden van de score door onderstaande code :

```
                    <a asp-page="./Details" asp-route-id="@item.Id" class="btn">
                        <div class="card-body text-left" style="min-height:400px;">
                            <h5>Merk : @Html.DisplayFor(modelItem =>
item.Brand.BrandName)</h5>
                            <h5>Prijs : @Html.DisplayFor(modelItem => item.Price)</h5>
                            <h5>
                                Score :
                                @for (int i = 0; i < Int32.Parse(roundScore); i++)
                                {
                                    <i class="fas fa-star" style="color:gold;"></i>
                                }
                                @for (int i = Int32.Parse(roundScore); i < 5; i++)
                                {
                                    <i class="far fa-star" style="color:gold;"></i>
                                }
                            </h5>
                            <hr />
                            <div class="text-center">
                                <img class="imageThumbnail" src="@imagePath" asp-append-
version="true" />
                            </div>
                        </div>
```

Stap 23: de bezoeker artikels laten beoordelen

Eerst dienen we nog een nieuw model aan te maken.
We wensen een tabel te voorzien waar elke gebruiker een individueel oordeel kan vellen over elk individueel artikel, en er zelfs nog commentaar zal aan toevoegen.

De gebruiker zal dit straks kunnen doen in de detail pagina van onze artikels, dus we zullen voor dit model straks geen afzonderlijke pagina's laten aanmaken (lees via scaffolding).

Voeg aan de map Models een nieuwe klasse toe :

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

        [ForeignKey("User")]
        public int UserId { get; set; }
        [Display(Name = "Gebruiker")]
        public User User { get; set; }

        [ForeignKey("Article")]
        public int ArticleId { get; set; }
        [Display(Name = "Artikel")]
        public Article Article { get; set; }

        [Display(Name = "Score")]
        [Range(1, 5, ErrorMessage = "Kies een waarde tussen 1 en 5")]
        public int Stars { get; set; } = 5;

        [Display(Name = "Opmerking")]
        public string Comment { get; set; }
    }
```

Voeg manueel aan de klasse /Data/SchoolShopContext.cs een extra DBSet toe :

```
        …
        public DbSet<SchoolShop.Models.Article> Article { get; set; }
        public DbSet<SchoolShop.Models.Score> Scores { get; set; }
    }
}
```

Voeg een migratie-bestand toe een laat je database updaten : de tabel Scores zal nu gemaakt worden voor jou.