



Eindproject

Pieterjan Blomme

20/9/2020

C# - Programmeren

Syntra – West Brugge

ASK-IT

Ondersteuning IT: helpdesk en infotheek

Als IT'er in een scholengemeenschap van 7 scholen met ongeveer 400 personeelsleden en 3500 leerlingen weet je altijd wel wat gedaan. We proberen met ons team een echt professionele omgeving uit te bouwen op vlak van infrastructuur, software, communicatieplatformen, automatisatie, veiligheid,... Met beperkte tijd en middelen vormen we toch de machinekamer van onze organisatie.

Het opzet van dit werk is het opzetten van een goed werkend IT-platform (communicatieplatform) waar leerkrachten en administratieve medewerkers op terecht kunnen met hun vragen, problemen en opmerkingen. Naast een helpdesk willen we het personeel ook zelfredzamer maken. Het is de bedoeling dat ze naast het vragen van hulp, ook efficient handleidingen, instructiefilmpjes en weblinks kunnen terugvinden.

De naam ASK-IT zegt alvast alles... als iemand ons een vraag stelt dan helpen wij die persoon graag verder. Als dat wat efficiënt en geordend kan gebeuren maken we het voor beide partijen een stuk gemakkelijker!



ASK-IT

We helpen je graag verder!

Inhoud

Vooropgestelde doelen	4
Authenticatie via Azure AD	7
Model classes	9
ViewModel class	12
De views	13
Sorteren en filteren in tickets	17
Mailing	20
Screenshots uploaden	21
Tickets exporteren	23
Infotheek	24
Publiceren	26

Vooropgestelde doelen

Voor ik aan het werk ging, schreef ik op papier enkele doelen uit. Zowel voor de gebruiker als voor de IT'ers.

1. De gebruiker moet...

- **efficiënt kunnen inloggen.**

Ik wilde niet het zoveelste platform creëren waar bij de gebruiker een nieuwe gebruikersnaam en paswoord voor moet aanmaken. In onze scholengemeenschap heeft iedereen een Azure-account. We streven er naar om hen op zoveel mogelijk platformen via SSO te laten inloggen.

- **heel makkelijk de weg vinden.**

Als een iemand onze hulp nodig heeft is het niet de bedoeling dat hij/zij nogmaals op een platform terechtkomt met een drukke en uitgebreide interface vol knoppen en linken. Het moet dus heel intuïtief en eenvoudig aanvoelen.

- **zich veilig voelen om een vraag te stellen.**

Een gebruiker mag zich niet geremd voelen een vraag te stellen. De omgeving moet afgeschermd worden zodat er niemand anders dan de IT'er het aangemaakte ticket kan bekijken.

- **snel geholpen kunnen worden.**

Eenmaal een ticket is aangemaakt verwacht de gebruiker ook een antwoord. Het is omslachtig om altijd maar weer naar het platform te surfen om te kijken of de IT'er het ticket al heeft opgevolgd. Een efficiënt mailsysteem achter het systeem lijkt me dan ook een meerwaarde.

- **zijn eigen tickets kunnen beheren.**

Een gebruiker moet een ticket kunnen aanmaken en bewerken. Bovendien moet hij zelf kunnen kiezen welke hij bewaard en welke hij wist.

- **de mogelijkheid hebben gemakkelijk informatie te vinden.**

Voorkomen is beter dan genezen... Met instructiefilmpjes en handleidingen kunnen we onszelf al heel wat werk besparen. Ook hier moet de gebruiker in het aanbod door de bomen het bos kunnen zien.

2. De IT'er (opvolger) moet...

- **een overzicht kunnen terugvinden van alle aangemaakte tickets.**

De opvolgers moeten heel snel een overzicht kunnen oproepen van alle op te volgen tickets. Dit zijn de tickets die open staan of in opvolging.

- **kunnen sorteren en filteren in de lijst met tickets.**

In een lijst van tientallen tickets moet de opvolger kunnen filteren. Welke tickets staan open? Welke staan in opvolging? Wat zijn de meest prioritaire tickets? Snel een specifiek ticket oproepen moet hij ook vlot kunnen.

- **afgewerkte tickets kunnen raadplegen.**

Het is handig om een archief aan te leggen van afgewerkte tickets. Zo kan er teruggegrepen worden naar afgewerkte tickets indien nodig.

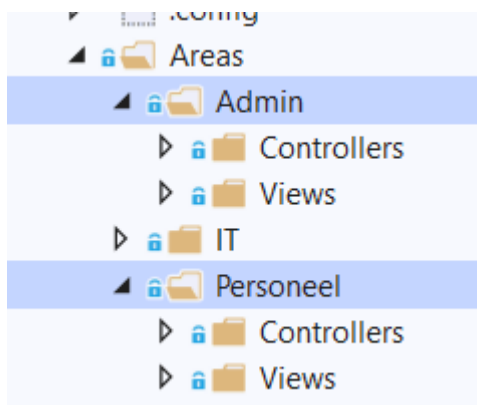
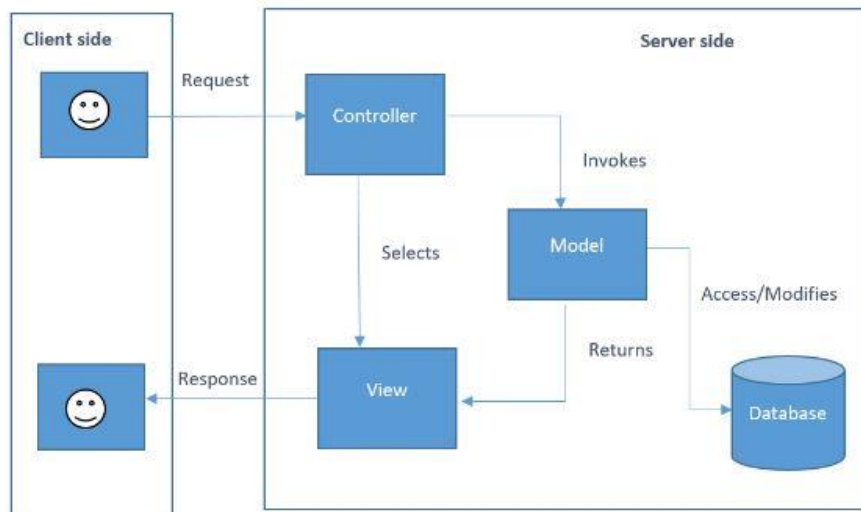
- **snel op de hoogte gebracht worden van een ticket die door hem opgevolgd moet worden.**

Als een ticket wordt aangemaakt moet de IT'er een melding krijgen. Zo kan hij kort op de bal spelen. De melding mag enkel binnenkomen als het een ticket betreft binnen zijn specialiteit.

- **makkelijk info en handleidingen op het platform kunnen plaatsen**

Naast het opvolgen van tickets, moet de IT'er ook instructiefilmpjes, handleidingen en weblinks ter beschikking kunnen stellen.

De architectuur



Het project is een ASP.NET Core Web application met design pattern MVC. Door models, views en controllers van elkaar te scheiden krijg je een duidelijkere structuur waarin elke component zijn functie heeft. Alles in verband met de weergave hoort in de 'view', de invoerlogica hoort in de 'controller' en de 'businesslogica' hoort in de 'model'.

Om het nog wat overzichtelijker te houden heb ik ook gebruikt gemaakt van 'Areas'. Ik heb meerdere

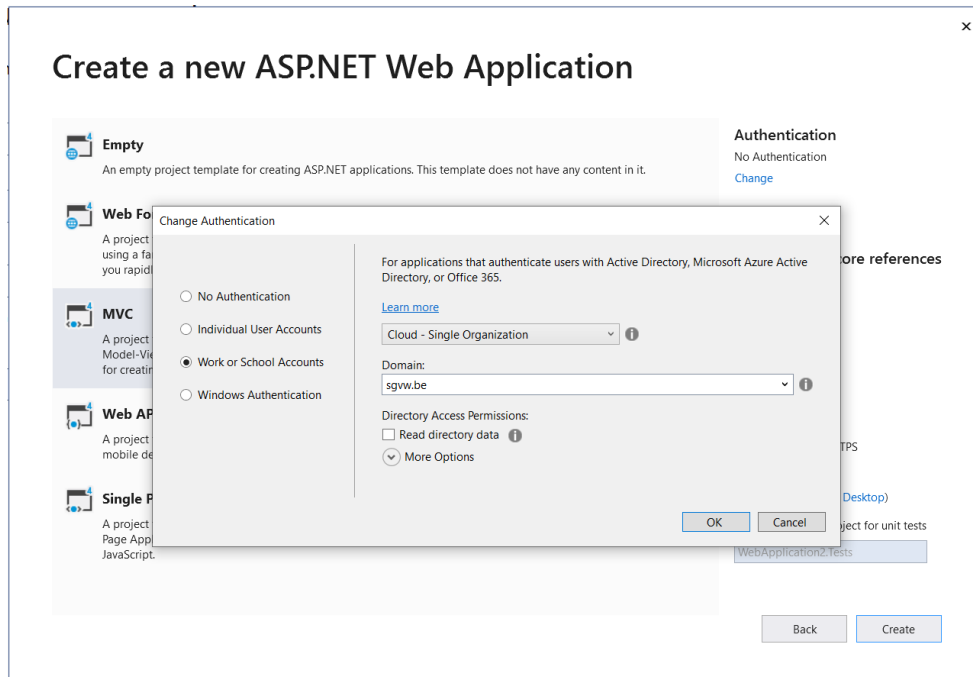
models, views en model classes waardoor dit moeilijker in het standaard MVC-patroon te gieten is. Met Areas kan ik het project in kleinere deeltjes opdelen. Zo onderscheidt ik het beheergedeelte van het gebruikersgedeelte.

De database is opgebouwd in SQL server en wordt aangesproken met de hulp van Entity Framework Core.

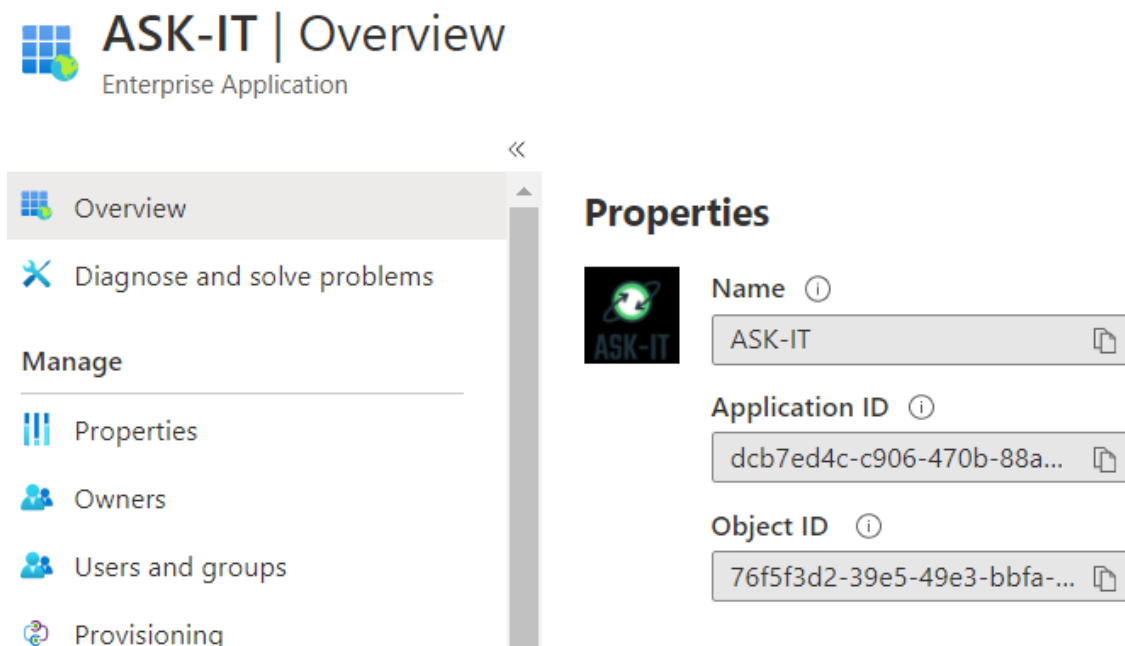
Authenticatie via Azure AD

Zoals eerder aangegeven was één van de doelstellingen dat de gebruikers zich kunnen aanmelden met hun office365-account.

Daarom heb ik bij de opstart gekozen om de authenticatie via Azure AD te doen verlopen :



Vervolgens heb ik het project geregistreerd in Azure AD. Het volledige stappenplan daarvoor doe ik hier niet uit de doeken. Dit is het resultaat:



De application ID heb ik vervolgens moeten importeren in de appsettings.json:

```
{
  "AzureAd": {
    "Instance": "https://login.microsoftonline.com/",
    "Domain": "sgvw.be",
    "TenantId": "5faf5315-8f04-4319-8aa2-e97a543ac975",
    // "ClientId": "efbb6b01-c5a4-4c57-92ef-dfef84b26d58",
    "ClientId": "dcb7ed4c-c906-470b-88a8-3bb7f9a0dcb2",
    "CallbackPath": "/signin-oidc"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

- Instance: url van de login-pagina;
- Domain: domain waarin AD zit;
- TenantId: ID van de tenant;
- ClientId: ID dat we meekregen bij de registratie van de app;
- Callback Path: route om in te loggen.

Nu kan de **connectie** gemaakt worden Azure AD.

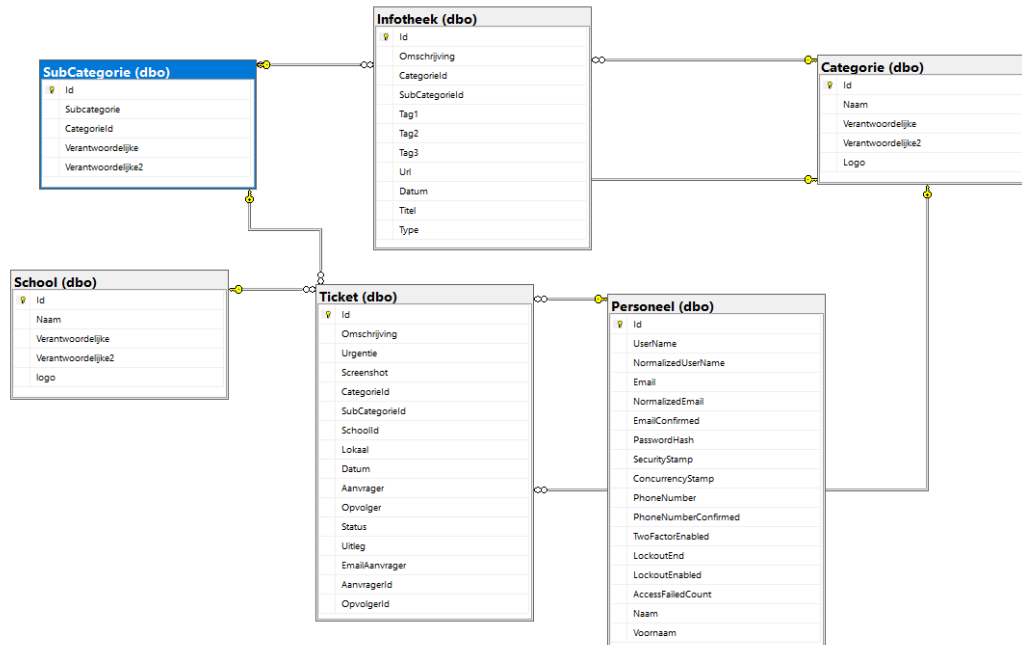
We geven nog mee in startup.cs - meer bepaald in de methode ConfigureServices() - dat ook de **authenticatie** via Azure AD moet gebeuren:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(AzureADDefaults.AuthenticationScheme)
        .AddAzureAD(options => Configuration.Bind("AzureAd", options));
}
```


Model classes

Dit is de structuur van de database achter het project. Deze is gemaakt in SQL server.

Er wordt gewerkt met 6 tabellen.



- **Tbl Ticket**

Bevat alle gegevens in verband met het aangemaakte ticket.

Maakt een **one-to-many** relatie met de tabel Personeel, de tabel school, de tabel categorie en de tabel subcategorie. Een ticket kan immers maar aan 1 gebruiker, school, categorie of subcategorie gelinkt zijn. Elke gebruiker, school, categorie of subcategorie kan in meerdere tickets voorkomen.

- **Tbl categorie/ Tbl subcategorie**

Bevat de (sub)categorieën waarin gebruikers hun tickets kunnen aanmaken.

We zien ook hier een one-to-many relatie. Een categorie kan meerde subcategorieën bevatten. Een subcategorie kan maar aan 1 categorie gekoppeld worden.

- **Tbl school**

Elke school van onze scholengemeenschap zit in deze tabel.

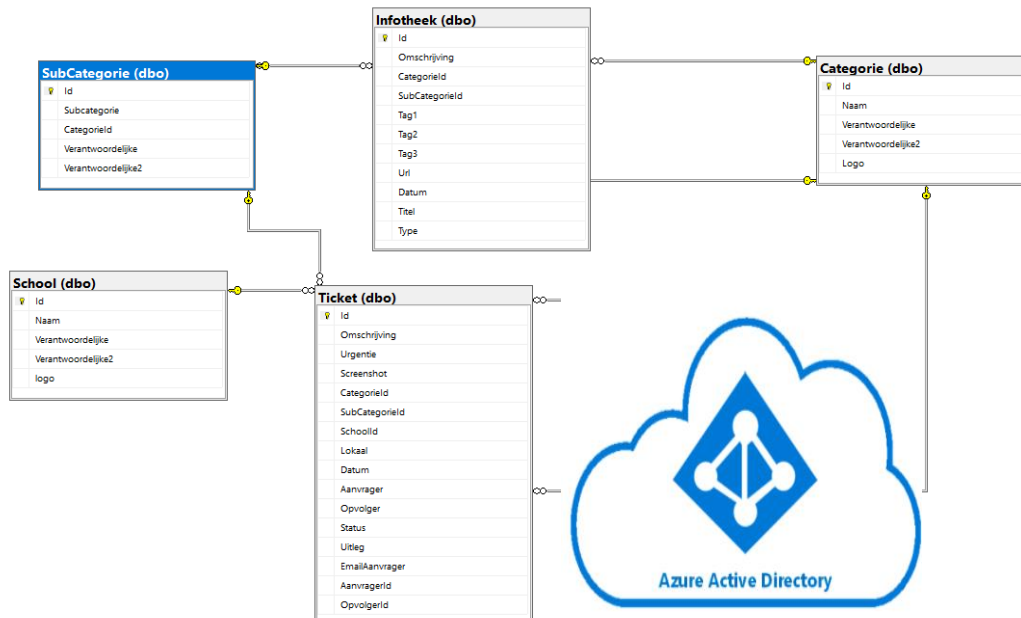
- **Tbl infotheek**

Dit is de tabel met alle url's naar handleidingen, tutorials en instructiefilmpjes.

De tabel is ook gelinkt aan de tabel categorieën en subcategorieën.

- **Tbl personeel**

De tabel personeel had ik oorspronkelijk aangemaakt (eigenlijk is die automatisch gegenereerd) en gelinkt aan de tabel tickets. Gaandeweg werd duidelijk dat ik deze tabel eigenlijk niet aanspreek aangezien ik gebruikersgegevens rechtstreeks uit Azure AD haal. Eigenlijk moet ik het databasediagram dus zo voorstellen:



Aanmaken model

- Nieuwe classes aanmaken in de map models (met key's en foreign key's)

```

namespace Helpdesk_SGVW.Models
{
    22 references
    public class Infotheek
    {
        [Key]
        14 references
        public int Id { get; set; }

        [Display(Name = "Titel")]
        10 references
        public string Titel { get; set; }

        [Display(Name = "Omschrijving")]
        16 references
        public string Omschrijving { get; set; }

        [Display(Name = "Datum upload")]
        2 references
        public DateTime Datum { get; set; } = DateTime.Now;

        [Display(Name = "Type")]
        21 references
        public string Type { get; set; }

        2 references
        public enum EType { Instructiefilmpje = 1, Webinar = 2, Handleiding = 3, Site = 4 }

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

        [ForeignKey("CategorieId")]
        15 references
        public virtual Categorie Categorie { get; set; }

        [Display(Name = "Subcategorie")]
        8 references
        public int SubCategorieId { get; set; }
    }
}
    
```

- We voegen de public properties met de naam van de tabellen toe aan ApplicationDbContext:

```
public class ApplicationDbContext : DbContext
{
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }

    22 references
    public DbSet<Categorie> Categorie { get; set; }
    20 references
    public DbSet<SubCategorie> SubCategorie { get; set; }
    15 references
    public DbSet<Ticket> Ticket { get; set; }
    12 references
    public DbSet<School> School { get; set; }
    0 references
    public DbSet<Personeel> Personeel { get; set; }
    9 references
    public DbSet<Infotheek> Infotheek { get; set; }
```

- We voegen EF Core EntityFramework migrations toe voor de tabellen via de Package Manager Console:

```
PM> Add-Migration AddSubCategory
```

- Via het commando update-database voegen we de tabellen toe aan de database:

```
PM> update-database
```

ViewModel class

Een ViewModel is een model class die properties bevat die vereist zijn voor een view. Het kan ook properties bevatten van meer dan 1 entiteit (tabel) van de database. Dit model is specifiek gemaakt voor de view-vereisten.

```
namespace Helpdesk_SGVW.Models.ViewModel
{
    8 references
    public class TicketViewModel
    {
        99+ references
        public Ticket Ticket { get; set; }
        3 references
        public IEnumerable<Categorie> Categorie { get; set; }
        2 references
        public IEnumerable<SubCategorie> SubCategorie { get; set; }
        3 references
        public IEnumerable<School> School { get; set; }
    }
}
```

Om straks de views aan te maken moeten verschillende tabellen in 1 view terecht komen.

Via het viewmodel creëren we bijvoorbeeld voor de subcategoriën een list van items via een property. Het viewmodel geven we dan door aan de razor view.

```
namespace Helpdesk_SGVW.Models.ViewModel
{
    8 references
    public class SubCategorieEnCategorieViewModel
    {
        3 references
        public IEnumerable<Categorie> CategorieLijst { get; set; }
        18 references
        public SubCategorie SubCategorie { get; set; }
        3 references
        public List<string> SubCategorieLijst { get; set; }
        2 references
        public string StatusBericht { get; set; }
    }
}
```

Het omzetten van items van gelijk welk type naar SelectListItem type (voor de razor list items) maken we via een generieke extension method, zodat we dit kunnen hergebruiken in ons project.

Met deze extension method kunnen we een IEnumerable<T> items naar een IEnumerable<SelectListItem> omvormen:

```
public static class IEnumerableExtension
{
    6 references
    public static IEnumerable<SelectListItem> ToSelectListItem<T>(this IEnumerable<T> items, int selectedValue)
    {
        return from item in items
               select new SelectListItem
               {
                   Text = item.GetPropertyValue("Naam"),
                   Value = item.GetPropertyValue("Id"),
                   Selected = item.GetPropertyValue("Id").Equals(selectedValue.ToString())
               };
    }
}
```

Via de subcategorie**controller** gaan we het viewmodel doorgeven aan de **view**:

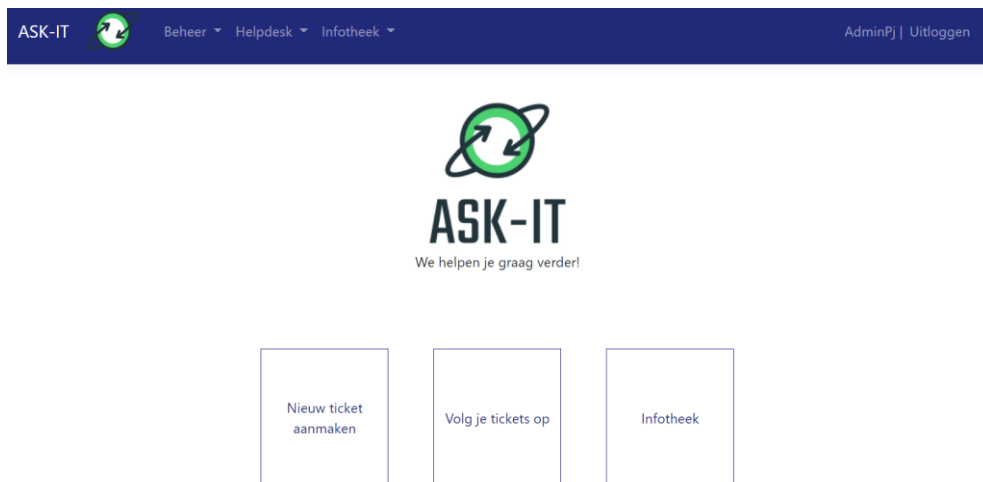
```
0 references
public async Task<IActionResult> Create()
{
    SubCategorieEnCategorieViewModel model = new SubCategorieEnCategorieViewModel()
    {
        CategorieLijst = await _context.Categorie.ToListAsync(),
        SubCategorie = new Models.SubCategorie(),
        SubCategorieLijst = await _context.SubCategorie.OrderBy(p => p.Subcategorie).Select(p => p.Subcategorie).Distinct().ToListAsync()
    };

    return View(model);
}
```

De views

Authorisatie in de views

De startpagina heb ik bewust zo eenvoudig mogelijk gehouden.



Het is niet de bedoeling dat een gebruiker dezelfde zaken ziet als een beheerder. Daarom gaan we bepaalde layout maar tonen als de gebruiker tot een bepaalde groep behoort. In dit geval zitten de IT'ers (opvolgers) in de groep "manager". Die rollen zijn gedefinieerd in Azure AD, waar alle gebruikersbeheer moet gebeuren.

ASK-IT | Users and groups
Enterprise Application

« + Add user Edit Remove Update Credentials Columns Got feedback?

The application will appear on the Access Panel for assigned users. Set 'visible to users?' to no in properties to prevent this. →

First 100 shown, to search all users & groups, enter a display name.

Display Name	Object Type	Role assigned
<input type="checkbox"/> admin1	User	Manager
<input type="checkbox"/> AD AdminPatrick	User	Manager
<input type="checkbox"/> AD AdminPj	User	Manager
<input type="checkbox"/> AD AdminSnd	User	Manager
<input type="checkbox"/> André Delanghe	User	Manager

```
<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
  <partial name="_LoginPartial" />
  <ul class="navbar-nav flex-grow-1">
    @if (User.IsInRole("Manager"))
    {
      <li class="nav-item dropdown text-white-50">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropDownMenuLink" role="button"
          data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Beheer
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropDownMenuLink">
          <a class="dropdown-item" asp-action="Index" asp-controller="Categorie" asp-area="Admin">Categorieën</a>
          <a class="dropdown-item" asp-action="Index" asp-controller="SubCategorie" asp-area="Admin">Subcategorieën</a>
          <a class="dropdown-item" asp-action="Index" asp-controller="School" asp-area="Admin">Scholen</a>
        </div>
      </li>
      <li class="nav-item dropdown text-white-50">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropDownMenuLink" role="button"
          data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Helpdesk
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropDownMenuLink">
          <a class="dropdown-item" asp-area="Personeel" asp-controller="Ticket" asp-action="Index">Op te volgen tickets</a>
          <a class="dropdown-item" asp-area="Personeel" asp-controller="Ticket" asp-action="IndexArchief">Archief tickets</a>
        </div>
      </li>
    }
  </ul>
</div>
```

De IT'ers uit de groep 'manager' zullen een andere edit-pagina zien dan gewone gebruikers. De velden 'status', 'opvolger' en 'uitleg' mogen immers enkel door de gebruiker gezien worden:

```
@if (User.IsInRole("Manager"))
{
  <div class="form-group row">
    <label asp-for="Ticket.Status" class="control-label"></label>
    <select asp-for="Ticket.Status" asp-items="Html.GetEnumSelectList<Ticket.EStatus>()" class="form-control"></select>
    <span asp-validation-for="Ticket.Status" class="text-danger"></span>
  </div>

  <div class="form-group row">
    <label asp-for="Ticket.Opvolger" class="control-label"></label>
    @*<select asp-for="Ticket.Opvolger" asp-items="Html.GetEnumSelectList<Ticket.EOpvolger>()" class="form-control"></select>*@
    <input asp-for="Ticket.Opvolger" value="@User.FindFirst("name").Value" class="form-control" />
    <span asp-validation-for="Ticket.Opvolger" class="text-danger"></span>
  </div>

  <div class="form-group row">
    <label asp-for="Ticket.Uitleg" class="col-form-label"></label>
    @Html.TextAreaFor(m => m.Ticket.Uitleg, new { @class = "form-control" })
    <span asp-validation-for="Ticket.Uitleg" class="text-danger"></span>
  </div>
}
```

Status

Opvolger

Uitleg

Partial razor view

Een Partial Razor View kan worden hergebruikt in andere views. Zo'n view heb ik bijvoorbeeld gebruikt de knoppen 'bewerken', 'wissen' en 'terug naar lijst' te kunnen oproepen. De navigatie zal ook hier afhankelijk zijn van de rol waarin je als gebruiker bent aangemeld.



Bekijk je ticket

Ticket ID: 26

Bewerk

Wis

Terug

```
<div class="row">
  <div class="col-3">
    @if (@Model.Status != "3")
    {
      <a asp-action="Edit" asp-area="Personeel" asp-controller="Ticket" asp-route-id="@Model.Id" class="btn btn-info form-control">Bewerk</a>
    }
  </div>

  <div class="col-3">
    <a asp-action="Delete" asp-area="Personeel" asp-controller="Ticket" asp-route-id="@Model.Id" class="btn btn-wis text-white form-control">Wis</a>
  </div>

  <div class="col-3">
    @if (User.IsInRole("Manager"))
    {
      @if (@Model.Status != "3")
      {
        <a asp-action="Index" asp-area="Personeel" asp-controller="Ticket" class="btn btn-success2 form-control">Terug</a>
      }
      @if (Model.Status == "3")
      {
        <a asp-action="IndexArchief" asp-area="Personeel" asp-controller="Ticket" class="btn btn-success2 form-control">Terug</a>
      }
    }
    else
    {
      <a asp-action="Overzicht" asp-area="Personeel" asp-controller="Home" class="btn btn-success2 form-control">Terug</a>
    }
  </div>
</div>
```

Ook het inloggen en uitloggen is gemaakt via een partial view:

```
@using System.Security.Principal

<ul class="navbar-nav">
  @if (User.Identity.IsAuthenticated)
  {
    <li class="nav-item">
      <span class="navbar-text"> @User.FindFirst("name").Value | </span>
    </li>
    <li class="nav-item">
      <a class="nav-link" asp-area="AzureAD" asp-controller="Account" asp-action="SignOut">Uitloggen</a>
    </li>
  }
  else
  {
    <li class="nav-item">
      <a class="nav-link" asp-area="AzureAD" asp-controller="Account" asp-action="SignIn">Inloggen</a>
    </li>
  }
</ul>
```

Thumbnails

Voor een gebruiker is overzicht en lay-out belangrijk. Ze moeten het aangenaam vinden om het platform te bezoeken. Ik wilde dat elke gebruiker (enkel) z'n eigen tickets kan raadplegen, aanpassen en wissen. Ik koos er voor om de tickets in als een soort thumbnails op het scherm te zetten:



Op 13/09/2020 10:24:56 stelde je een vraag over computers

Afgewerkt

Je vraag: *Mijn computer valt steeds uit na 5 minuten*

Ticket ID: 28

Urgentie: *Hoog*

Opvolger: AdminPj

Bedankt voor de melding. Ik heb de computer vervangen door een nieuw toestel.

Bekijk en bewerk

```

@model IEnumerable<Ticket>

@if (Model.Count() > 0)
{
    @foreach (var item in Model)
    {
        <div class="border border-info rounded col-12" style="margin-bottom:10px; margin-top:10px; padding:10px">
            <div class="row">
                <div class="col-md-3 col-sm-12">
                    <img srcset=@item.School.Logo title=@item.School.Naam />
                    <center>
                        <img srcset="/images/logo/askit3.png" title="Ask-IT" />
                    </center>
                </div>
                <div class="col-md-9 col-sm-12">
                    <div class="row pr-3">
                        <div class="col-12 text-right" style="color:#212a79">
                            <b>
                                @if (@item.Status == "1")
                                {<label style="color:red">Open</label>}
                                @if (@item.Status == "2")
                                {<label style="color:blue">Opgevolgd </label>}
                                @if (@item.Status == "3")
                                {<label style="color:green">Afgewerkt </label>}
                            </b>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    }
}

```

Ook voor de infotheek maakte ik de views op een gelijkaardige manier.

ALLES

HARDWARE


TEAMS

OFFICE365

SOFTWARE

BEAMERS

Teams



Aanmaken Teams | Intranet.sgvw.be

Zelf een team aanmaken

Kernwoorden: Teams | aanmaken | Office365

Type: Instructiefilmpje

Bekijk

Sorteren en filteren in tickets

Het is belangrijk voor de opvolgers om bepaalde tickets heel snel te kunnen oproepen. Daarom maakte ik boven de lijstweergave een vrij uitgebreide zoekfunctie:

Op te volgen tickets

Nieuw ticket

<input type="text" value="TicketID"/>	<input type="text" value="Aanvrager"/>	<input type="text" value="Selecteer status"/>
<input type="text" value="Omschrijving"/>	<input type="text" value="School"/>	<input type="text" value="Selecteer opvolger"/>
<input type="text" value="Categorie"/>	<input type="text" value="Zoek"/>	<div>21 op te volgen tickets</div>

Id	Datum	Omschrijving	Urgentie	Categorie	Subcategorie	School	Aanvrager	Status	Opvolger	
27	13/09/2020 10:06:48	gf	Laag	Hardware	computers	Immaculata De Panne	AdminPj	In opvolging	AdminPj	Bewerk Details Wis
26	12/09/2020 15:09:18	gf	Laag	Hardware	computers	Immaculata De Panne	AdminPj	Open	Niemand	Bewerk Details Wis
25	12/09/2020 15:09:07		Laag	Hardware	computers	Immaculata De Panne	AdminPj	Open	Niemand	Bewerk Details Wis

Sorteren

In de view heb ik de gewone kolomnaam vervangen door een link. Ik geef ook de *kolomtitel* mee, de aan te spreken *methode* in de controller en via een *ViewBag* geef ik mee welke gegevens moeten gesorteerd worden.

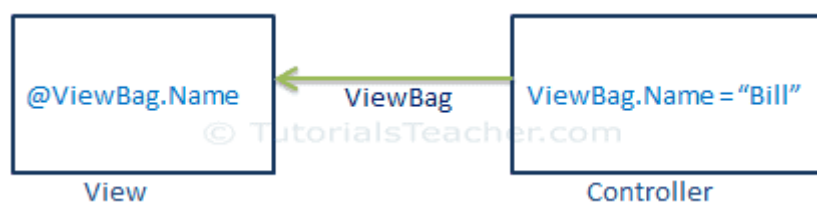
```

<th>
    @Html.ActionLink("Urgentie", "Index", new { sortOrder = ViewBag.UrgentieSortParm })
</th>

<th>
    @Html.ActionLink("Categorie", "Index", new { sortOrder = ViewBag.CategorieSortParm })
</th>
<th>
    @Html.ActionLink("Subcategorie", "Index", new { sortOrder = ViewBag.SubCategorieSortParm })
</th>
<th>
    @Html.ActionLink("School", "Index", new { sortOrder = ViewBag.SchoolSortParm })
</th>

```

De ViewBag is ASP.NET MVC wordt gebruikt om tijdelijke gegevens (die niet in het model zijn opgenomen) over te dragen van de controller naar de view.



Die gegevens geven we dus door aan de controller, meer bepaald aan de methode index:

```
1 reference
public async Task<ActionResult> Index(string sortOrder, int zoektermId, string zoektermOmschrijving, string zoektermAanvraag)
{
    var applicationDbContext = _context.Ticket.Include(m => m.Categorie).Include(m => m.SubCategorie).Include(m => m.School);

    ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "aanvrager_desc" : "";
    ViewBag.SchoolSortParm = String.IsNullOrEmpty(sortOrder) ? "school_desc" : "";
    ViewBag.UrgentieSortParm = String.IsNullOrEmpty(sortOrder) ? "urgentie_desc" : "";
    ViewBag.CategorieSortParm = String.IsNullOrEmpty(sortOrder) ? "categorie_desc" : "";
    ViewBag.SubCategorieSortParm = String.IsNullOrEmpty(sortOrder) ? "subcategorie_desc" : "";
    ViewBag.StatusSortParm = String.IsNullOrEmpty(sortOrder) ? "status_desc" : "";
    ViewBag.DateSortParm = sortOrder == "Date" ? "date_desc" : "Date";
    var tickets = from s in _context.Ticket
                  select s;
```

De methode gebruikt LINQ om de kolom op te geven waarop moet worden gesorteerd. De code maakt een IQueryable variabele 'tickets' aan voor het switch-statement, wijzigt het vervolgens in het switch-statement en roept dan de ToListAsync-methode aan

```
switch (sortOrder)
{
    case "aanvrager_desc":
        tickets = tickets.OrderByDescending(s => s.Aanvrager);
        break;

    case "school_desc":
        tickets = tickets.OrderByDescending(s => s.School);
        break;

    case "urgentie_desc":
        tickets = tickets.OrderByDescending(s => s.Urgentie);
        break;

    case "categorie_desc":
        tickets = tickets.OrderByDescending(s => s.Categorie);
        break;

    case "subcategorie_desc":
        tickets = tickets.OrderByDescending(s => s.SubCategorie);
        break;

    case "status_desc":
        tickets = tickets.OrderByDescending(s => s.Status);
        break;

    default:
        tickets = tickets.OrderByDescending(s => s.Datum);
        break;
};

return View(await tickets.ToListAsync());
```

Filteren

Om data te kunnen filteren heb ik een html-form aangemaakt. Het formulier verzamelt gegevens ("Get") in een textbox of keuzelijst en geeft die door als parameter aan de methode index in de controller.

```

<div class="row" style="background-color:#fff">
  <div class="col-md-3">
    @Html.TextBox("zoektermId", null, new { @placeholder = "TicketID", @class = "form-control" })<br />
    @Html.TextBox("zoektermOmschrijving", null, new { @placeholder = "Omschrijving", @class = "form-control" })<br />
    @Html.TextBox("zoektermCategorie", null, new { @placeholder = "Categorie", @class = "form-control" })<br />
  </div>
  <div class="col-md-1">
  </div>
  <div class="col-md-3">
    @Html.TextBox("zoektermAanvrager", null, new { @placeholder = "Aanvrager", @class = "form-control" })<br />
    @Html.TextBox("zoektermSchool", null, new { @placeholder = "School", @class = "form-control" })
    <label class="col-form-label" style="color:white">Zoek: </label><input type="submit" value="Zoek" class="btn btn-succes form-control" />
  </div>
  <div class="col-md-1">
  </div>
  <div class="col-md-3">
    @Html.DropDownList("zoektermStatus", new SelectList(Enum.GetValues(typeof(Ticket.EStatus))), "Selecteer status", new { @class = "form-control" })<br />
    @Html.DropDownList("zoektermOpvolger", new SelectList(Enum.GetValues(typeof(Ticket.EOpvolger))), "Selecteer opvolger", new { @class = "form-control" })<br />
    <label style="font-size:40px; border:2px solid #212a79;">@ViewBag.AantalTickets </label> <label> op te volgen tickets </label>
  </div>
  <div class="col-md-1">
  </div>
</div>
</div>
</p>
</div>

// GET: Personeel/Ticket
1 reference
public async Task<IActionResult> Index(string sortOrder, int zoektermId, string zoektermOmschrijving, string zoektermAanvrager, string zoektermCategorie, string zoektermSchool, :
{

```

Ook hier gaan we de variabele 'tickets' aanpassen. We selecteren de tickets die voldoen aan een bepaalde zoekterm:

```

if (!string.IsNullOrEmpty(zoektermAanvrager) || !string.IsNullOrEmpty(zoektermOmschrijving) || !string.IsNullOrEmpty(zoektermCategorie) || !string.IsNullOrEmpty(zoektermSchool))
{
    tickets =
        applicationDbContext
            .Where(s => s.Aanvrager.Contains(zoektermAanvrager))
            .Where(s => s.Omschrijving.Contains(zoektermOmschrijving))
            .Where(s => s.Categorie.Naam.Contains(zoektermCategorie))
            .Where(s => s.School.Naam.Contains(zoektermSchool))
            .Where(s => s.Opvolger.Contains(zoektermOpvolger))
            .Where(s => s.Status.Contains(zoektermStatus));
}
else if (zoektermId != 0)
{
    tickets =
        applicationDbContext
            .Where(s => s.Id.Equals(zoektermId));
}
else
{
    tickets = applicationDbContext;
}

```

Mailing

Het lijkt me belangrijk dat een gebruiker of IT'er op de hoogte wordt gebracht als er een ticket is aangemaakt of bewerkt. De mailing moet ook op maat gebeuren.

Daarom kan aan elke categorie of school het mailadres van een verantwoordelijke gekoppeld worden. Deze worden dus ook opgeslaan in de database:

Categorie	Logo	Verantwoordelijke	Verantwoordelijke 2	
Hardware	/images/logo/HW.jpg	blmpt@sgvw.be	blmpt@sgvw.be	Bewerk Details Wis
Teams	/images/logo/teams.png	blmpt@sgvw.be	blmpt@sgvw.be	Bewerk Details Wis

In de methode 'create' in de controller heb ik met 'mailkit' een mailfunctie aangemaakt:

```
//stuur een mail
var message = new MimeMessage();
message.From.Add(new MailboxAddress("HelpdeskIT", "ask-it@sgvw.be"));
message.To.Add(new MailboxAddress("Aanvrager", TicketVM.Ticket.EmailAanvrager));
message.To.Add(new MailboxAddress("VerantwoordelijkeSchool", _context.School.Where(u => u.Id == TicketVM.Ticket.SchoolId).
//message.To.Add(new MailboxAddress("VerantwoordelijkeSchool2", _context.School.Where(u => u.Id == TicketVM.Ticket.SchoolId).
message.To.Add(new MailboxAddress("VerantwoordelijkeCategorie", _context.Categorie.Where(u => u.Id == TicketVM.Ticket.Cate
//message.To.Add(new MailboxAddress("VerantwoordelijkeCategorie2", _context.Categorie.Where(u => u.Id == TicketVM.Ticket.C
message.To.Add(new MailboxAddress("VerantwoordelijkeSubCategorie", _context.SubCategorie.Where(u => u.Id == TicketVM.Ticke
//message.To.Add(new MailboxAddress("VerantwoordelijkeSubCategorie2", _context.SubCategorie.Where(u => u.Id == TicketVM.Ti

message.Subject = "nieuw helpdeskticket: " + TicketVM.Ticket.Id;
message.Body = new TextPart("html")
{
    Text = "Beste collega <p>" +
        "Er is een nieuwe helpdesk-ticket aangemaakt <br>" +
        "<br><b> Ticket-Id: </b>" + TicketVM.Ticket.Id +
        "<br><b> Aanvrager: </b>" + TicketVM.Ticket.Aanvrager +
        "<br><b> Categorie: </b>" + _context.Categorie.Where(u => u.Id == TicketVM.Ticket.CategorieId).FirstOrDefault().Naam +
        "<br><b> Subcategorie: </b>" + _context.SubCategorie.Where(u => u.Id == TicketVM.Ticket.SubCategorieId).FirstOrDefault()
        "<br><b> School: </b>" + _context.School.Where(u => u.Id == TicketVM.Ticket.SchoolId).FirstOrDefault().Naam +
        "<br><b> Omschrijving: </b><br>" + TicketVM.Ticket.Omschrijving +
        "<p> We volgen het voor u op" +
        "<p>Met vriendelijke groet" +
        "<br> Het IT-team"
};

using (var client = new SmtpClient())
{
    client.Connect("smtp.office365.com", 587, false);
    client.Authenticate("ask-it@sgvw.be", "Qav75583");
    client.Send(message);
    client.Disconnect(true);
}
```

Bij het aanmaken van een ticket zal dus een mail gestuurd worden naar de verantwoordelijke van de (sub)categorie, de schoolverantwoordelijke en de aanvrager.

Ook als een ticket bewerkt is zal op analoge manier een mail gestuurd worden naar dezelfde personen.

Mails worden verstuurd vanuit het account ask-it@sgvw.be

Screenshots uploaden

Als IT'er is het handig dat de gebruiker een screenshot kan meegeven met het ticket. Daarom heb ik het ook mogelijk gemaakt om een afbeelding te uploaden:

Screenshot

Bestanden kiezen Geen bestand gekozen

In de controller in de methode 'create' heb ik volgende code ingegeven:

```
//Bewaar afbeelding

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

var ticketFromDb = await _context.Ticket.FindAsync(TicketVM.Ticket.Id);

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

    using (var fileStream = new FileStream(Path.Combine(uploads, TicketVM.Ticket.Id + extension), FileMode.Create))
    {
        files[0].CopyTo(fileStream);
    }
    ticketFromDb.Screenshot = @"images\" + TicketVM.Ticket.Id + extension;
}
else
{
    //Als er geen afbeelding is meegegeven
    var uploads = Path.Combine(webRootPath, @"images\" + "askit.png"); ;
    System.IO.File.Copy(uploads, webRootPath + @"images\" + TicketVM.Ticket.Id + ".png");
    ticketFromDb.Screenshot = @"images\" + TicketVM.Ticket.Id + ".png";
}
```

Als er geen screenshot wordt meegegeven zal het logo van de site weergegeven worden als default.

De afbeelding moet ook bewerkt kunnen worden in de 'edit'-methode:

```
//Een afbeelding uploaden

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

var menuItemFromDb = await _context.Ticket.FindAsync(TicketVM.Ticket.Id);


if (files.Count > 0)
{
    //Nieuwe image
    var uploads = Path.Combine(webRootPath, "images");
    var extension_new = Path.GetExtension(files[0].FileName);

    //Wis originele afbeelding
    var imagePath = Path.Combine(webRootPath, menuItemFromDb.Screenshot.TrimStart('\'));

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

    //Nieuwe afbeelding uploaden
    using (var fileStream = new FileStream(Path.Combine(uploads, TicketVM.Ticket.Id + extension_new), FileMode.Create))
    {
        files[0].CopyTo(fileStream);
    }
    menuItemFromDb.Screenshot = @"images\" + TicketVM.Ticket.Id + extension_new;
}
```

Ik zorg het dan ook voor dat in de detail-weergave van het ticket de screenshot mooi te zien is:



Bekijk je ticket

Ticket ID: 28

Wis
Terug

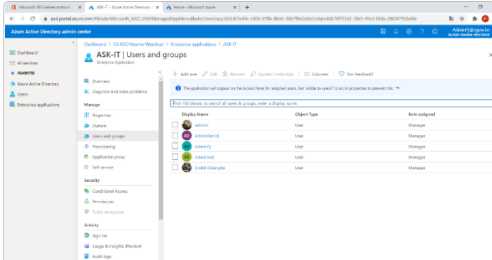
Datum
13/09/2020 10:24:56

Aanvrager
AdminPj

Omschrijving
Mijn computer valt steeds uit na 5 minuten

Urgentie
Hoog

Categorie
Hardware



Fragment uit de razor-view:

```

<div class="col-md-5">
  <div class="form-group row">
    <div>
      <p></p>
      
    </div>
  </div>
</div>
</div>

```

Tickets exporteren

Met ClosedXML kunnen we gemakkelijke gegevens exporteren naar Excel.

In deze methode genereer ik een generieke lijst met alle tickets:

```
IList<Ticket> TicketsEXP = await _context.Ticket.Include(m => m.Categorie).Include(m => m.SubCategorie).Include(m => m.School).ToListAsync();

using (var workbook = new XLWorkbook())
{
    var worksheet = workbook.Worksheets.Add("Ticket");
    var currentRow = 1;
    var currentCol = 1;
    foreach (var property in TicketsEXP.GetType().GetGenericArguments()[0].GetProperties())
    {
        worksheet.Cell(currentRow, currentCol++).Value = property.Name;
    }
    foreach (var item in TicketsEXP)
    {
        currentCol = 1;
        currentRow++;
        foreach (var property in item.GetType().GetProperties())
        {
            worksheet.Cell(currentRow, currentCol++).Value = property.GetValue(item);
        }
    }
    using (var stream = new MemoryStream())
    {
        workbook.SaveAs(stream);
        var content = stream.ToArray();

        return File(
            content,
            "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
            "Tickets.xlsx");
    }
}
```

We vragen dus van elk ticket de waarden op en streamen ze in een array door naar de cellen in Excel.

Op het moment van het verschijnen van dit document stond dit nog niet helemaal op punt. In de exportlijst staat de wel de verwijzingen naar ID's van andere tabellen (school, categorie, subcategorie,...) met niet de waarde zelf. Dit zal opgelost moeten worden met een viewmodel te werken.

Infotheek

Om de hoeveelheid tickets wat te beperken heb ik de infotheek aan het project gekoppeld. Dat is een databank aan instructiefilmpjes, opgenomen webinars, handleidingen en weblinks. Deze staan allemaal verzameld in MS streams, sharepoint, OneDrive,... Het leek me dan ook aangewezen de info in een databank te verzamelen en die geordend aan te bieden.

Ik heb er voor gekozen om de categorieën in een menubalk te gieten. Zo kan een gebruiker per categorie de gewenste info opzoeken. Ik koppelde aan elk item ook zoektags.

Nieuw item aanmaken

Id	Titel	Omschrijving	Datum upload	Categorie	SubCategorie	Tag1	Tag2	Tag3	Type	
1	Intranet.sgvw.be	Zelf een team aanmaken	9/19/2020 4:36:34 PM	Teams	Aanmaken Teams	Teams	aanmaken	Office365	Instructiefilmpje	Bewerk Details Wis
2	Een toets maken met MS Forms	In deze webinar maken we een toets met evaluatie in Forms	9/19/2020 3:26:44 PM	Office365	Forms	Forms	Toets	evaluatie	Webinar	Bewerk Details Wis
3	Inloggen op smartschool		9/19/2020 4:09:07 PM	Smartschool	computers	Inloggen	smartschool	eerste maal	Handleiding	Bewerk Details Wis

De items worden aan de gebruikers voorgesteld in een thumbnailview. U ziet ook de categorieën als menu voorgesteld.

ALLES HARDWARE TEAMS OFFICE365 SOFTWARE SMARTSCHOOL

Teams



Aanmaken Teams | **Intranet.sgvw.be**

Zelf een team aanmaken

Kernwoorden: Teams | aanmaken | Office365

Type: Instructiefilmpje

[Bekijk](#)

Office365



Forms | **Een toets maken met MS Forms**

In deze webinar maken we een toets met evaluatie in Forms

Kernwoorden: Forms | Toets | evaluatie

Type: Webinar

[Bekijk](#)

Met dit script kon ik de menubalk aanmaken:

```
<div class="backgroundWhite container">

    <ul id="menu-filters" class="menu-filter-list list-inline text-center">
        <li class="active btn btn-secondary ml-1 mr-1" data-filter=".menu-infotheek">Alles</li>
        @foreach (var item in Model.Categorie)
        {
            <li class="ml-1 mr-1" data-filter=".@item.Naam.Replace(" ",string.Empty)">@item.Naam</li>
        }
    </ul>

    @foreach (var categorie in Model.Categorie)
    {
        <div class="row" id="menu-wrapper">
            <partial name="ThumbnailAreaInfotheek" model="@Model.Infotheek.Where(u=>u.Categorie.Naam.Equals(categorie.Naam))" />
        </div>
    }
</div>

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

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

        (function ($) {

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

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


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

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

        })(jQuery);</script>
    </section>
```

In de views kon ik instructiefilmpjes (die in MS streams opgeslaan zijn) imbedden zodat de gebruiker niet uit de application moet:

Bewerk
Terug


Intranet.sgvw.be

Instructiefilmpje

Omschrijving

Zelf een team aanmaken

Categorie

Teams

SubCategorie

Aanmaken Teams

Tag1

Teams





Tag2


aanmaken



Tag3

Office365

Aanmaken Teams en vakken in smartschool
33 0 0

 Smartschool
 Docento
 Unikid
 Teams

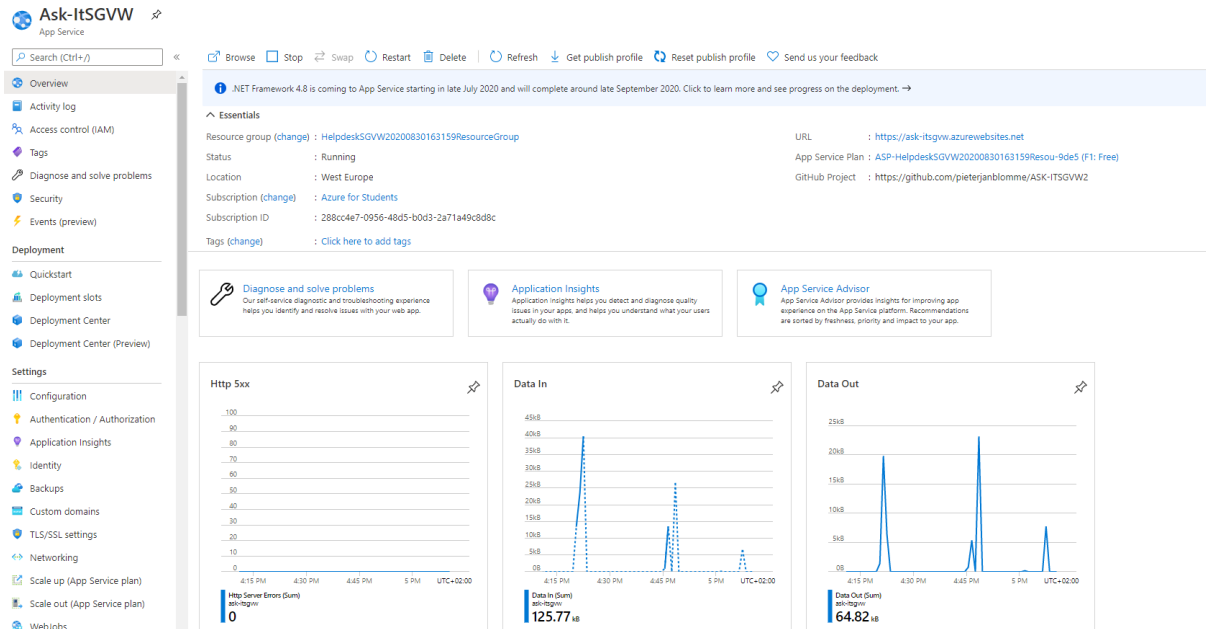


 Immaculata
 Talentis

Publiceren

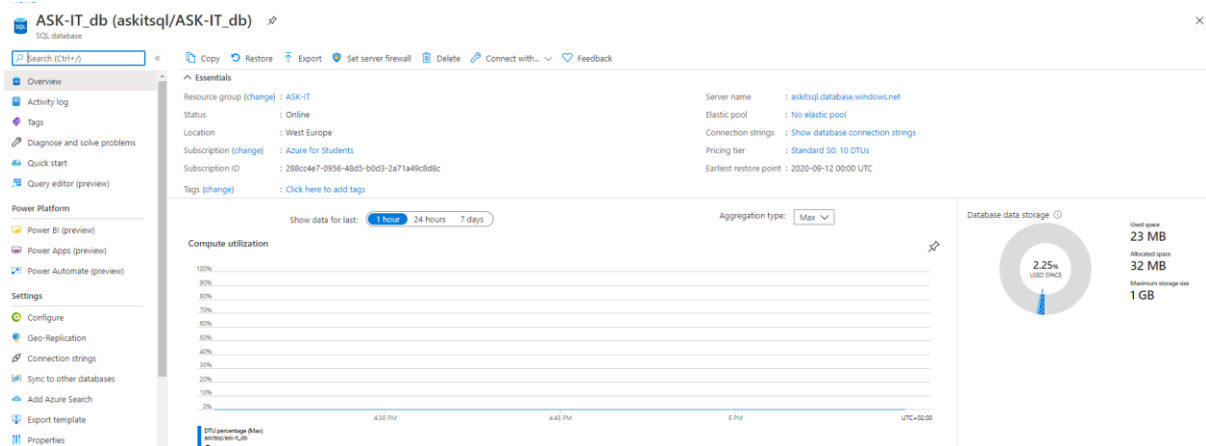
Het project is gepubliceerd in Azure en GitHub.

Om dit te kunnen doen heb ik mezelf een abonnement moeten toekennen.



Vanuit Azure kunnen we heel wat data opvolgen. Er hangen ook uitgebreide logs en access control aan vast. Het is een omgeving waarin ik mij nog grondig moet verdiepen, maar er is enorm veel mee te doen.

Ook de database zelf staat gepubliceerd in Azure:



Ook hier kan enorm veel opgevolgd en bijgestuurd worden.

Dit zijn de abonnementsprijzen in Azure:






- ☒ **Developer plan** **24.46 €/month**
For developers or teams looking to quickly and effectively get started on Azure, technical support is available weekdays from 9:00 AM to 5:00 PM with initial response times under 8 business hours.
- ☐ **Standard plan** **84.33 €/month**
For teams running production applications, get 24x7 technical support and fast initial response times under 2 hours.
- ☐ **Professional Direct plan** **843.30 €/month**
For business-critical applications, a cloud advisor provides guidance and advocacy to help improve reliability and optimize costs. You also get 24x7 technical support with the fastest initial response times under 1 hour.
- ☐ **No technical support** **Free**
No technical support or I'm already covered through Microsoft Premier Support.

Vanuit Visual Studio kan heel gemakkelijk gepubliceerd worden. Tenminste als de connectiestring correct staat. Ik experimenteer meestal met de locale database in SQLserver. Als ik echt veranderingen wil doorvoeren in de live-omgeving, dan verandering de settings in appsettings.json en voer ik een update-database door:

```
"ConnectionStrings": {
  "DefaultConnection": "Data Source=askitsql.database.windows.net;Initial Catalog=ASK-IT_db;User ID=Adminpj;Password=sgvw8660@immac;Connect Timeout=30;Encrypt=True;TrustServerCertificate=True;"
  //"Server=(localdb)\\mssqllocaldb;Database=Helpdesk_SGVW_DB_15062020;Trusted_Connection=True;MultipleActiveResultSets=true"
```

Het project is zo bereikbaar via volgende link:

Summary

Site URL	http://ask-itsgvw.azurewebsites.net 
Configuration	Release 
Target framework	netcoreapp3.1 
Deployment mode	Framework-dependent 
Target runtime	Portable 

Conclusie

Ik denk dat ik met dit project een groot deel van de vooropgestelde doelen kon bereiken. Er blijven nog enkele zaken die beter kunnen:

- De views kunnen nog iets compacter
- Het exporteren van gegevens in Excel staat nog niet op punt
- Code kan hier en daar 'properder'

Ik heb ook nog veel ongebruikte code staan. Dat is het resultaat van heel proberen en herproberen. Bepaalde code heb ik bewust laten staan.

Het is een heel leerrijk project geweest. Mocht ik nu herbeginnen met de kennis die ik nu heb zou het resultaat veel opgepoetster zijn.

Toch ben ik enigszinds trots op wat ik heb bereikt. Uiteindelijk kon in anderhalf jaar geleden nog niets. Nu kan ik toch al een vrij mooi project voorleggen.

Ik herhaal nog even mijn vooropgestelde doelen en ken mezelf een cijfer toe met de mate waarin ik er in ben geslaagd:

Wie	Wat	Zeer goed	Goed	Matig	Slecht
Gebruiker	efficiënt kunnen inloggen.	X			
	Heel makkelijk de weg vinden		X		
	Snel geholpen kunnen worden		X		
	Eigen tickets beheren		X		
	Makkelijk info terugvinden		X		
IT'er	Overzicht aangemaakte tickets		X		
	Sorteren en filteren			X	
	Afgewerkte tickets raadplegen		X		
	Snel op de hoogte zijn		X		
	Handleidingen en info publiceren		x		

Hopelijk is dit het begin van vele mooie projecten die nog mogen volgen. Ik heb er in elk geval zin in!