SDN MAGAZINE

Nummer
148
sept 2023

SDN Magazine is
een uitgave van:
SDN

Van en voor Microsoft & .NET professionals



> GraphQL voor .NET Developers 9
> Aan de slag met OpenAI 20
> VPS met ARCore 27

BERGLER

SOFTWARE SOLUTIONS



KENNIS IS MACHT, KENNIS DELEN IS KRACHT!

BERGLER SOFTWARE SOLUTIONS IS OP ZOEK NAAR ERVAREN

LEAD .NET DEVELOPERS.

In de rol van Technical Lead Consultant ben je een meewerkend voorman op het gebied van software development, -kwaliteit, security, Agile processen, SOLID, DevOps, etc.

Samen met je collega's, ben je altijd bezig jezelf verder te ontwikkelen op het gebied van software development binnen de **MICROSOFT STACK**.

Wij geven je 10%-tijd om je in te zetten binnen ons Competence Center (www.bergler.nl/competence-center/).

Kijk ook eens op https://connect.bergler.nl/ om je toekomstige collega's te leren kennen.

BEN JE GEÏNTERESSEERD IN DEZE UITDAGENDE EN AFWISSELENDE FUNCTIE?
Neem dan contact op met Ronald Jansen (HR Manager Bergler Software Solutions):
E: rjansen@bergler.nl. Bellen mag natuurlijk ook: 076-5720200

www.bergler.nl

May the source be with you!

Colofon

Uitgave
Software Development Network
26ste jaargang
Nr. 148 • september 2023

Redactiecommissie:
Annejan Barelds, Nadine Wolff, Jan de Vries, John Bruin, Menno Jongerius, Marcel Meijer, Roy Janssen, Roelant Dieben, Thijs den Hollander.

Auteurs:
Vincent Baaij, Johan Smarius, Afra Nur Doludizgin, Patrick Bes, Mart de Graaf, Pieter Baart, Bart Klomp, John Bruin

Listings:
Zie de website www.sdn.nl voor eventuele source files uit deze uitgave

Contact:
Software Development Network
info@sdn.nl
redactie@sdn.nl

Adverteren:
Informatie over adverteren en de tarieven kunt u opvragen bij Kelly Verschoor
kelly.verschoor@sdn.nl

©2023 Alle rechten voorbehouden. Niets uit deze uitgave mag worden overgenomen op welke wijze dan ook zonder voorafgaande schriftelijke toestemming van SDN. Tenzij anders vermeld zijn artikelen op persoonlijke titel geschreven en verwoorden zij dus niet noodzakelijkerwijs de mening van het bestuur en/of de redactie. Alle in dit magazine genoemde handelsmerken zijn het eigendom van hun respectievelijke eigenaren.

Beste SDN-er,

Het was weer een mooie typische Hollandse zomer diepe dalen en hoge pieken. Als je lekker op vakantie bent geweest, dan hopen wij dat je helemaal uitgerust bent voor het nieuwe SDN-seizoen. Wij zijn ook weer uitgerust en we hebben in ons netwerk gevraagd voor content voor dit magazine. Heb jij in de vakantie ook nieuwe technologieën of nieuwe kennis opgedaan en wil je die graag breed delen? Dat kan via ons magazine, mail naar redactie@sdn.nl.

Over nieuwe technologieen gesproken, kijken jullie al uit naar de volgende .NET versie nummer 8, nog meer platform onafhankelijker? En C# 12 met inline arrays en interceptors? In november vindt .NET Conf 2023 plaats voor alle nieuwtjes rondom deze versie en veranderingen. Als SDN zullen wij daar natuurlijk ook de nodige aandacht aan besteden. Na een fijne zomer komt er dus een prachtig najaar.

Ben je een frontend developer? Frontmania Conference zal in oktober plaats vinden. En als SDN-lid krijg je een aanzienlijke korting! Leer het laatste op gebied van Javascript en frontend.

In dit magazine zit een Blazor story, GraphQL en Developers, Generative AI met Microsoft Fabric en Azure OpenAI Service, Container Orchestratie, OpenAI, Minimal API's, VPS met ARCore en een geinige code tip.

Kortom genoeg te lezen! Vincent, Johan, Afra, Patrick, Mart, Pieter, Bart en John, dank je wel! En zonder onze geweldige redactiecommissie was dit magazine niet zo tot stand gekomen!

Veel leesplezier!
Marcel



DEVITJOBS.NL

LEAD .NET DEVELOPER

Salarisindicatie:
60.000 - 75.000 EUR

Locatie:
Breda

Technologieën:
.NET Framework, DevOps, Azure, C#,.NET

VITAS

Full Stack Developer

Salarisindicatie:
60.000 - 90.000 EUR

Locatie:
Amsterdam

Technologieën:
Fullstack, JavaScript, C#, Java, Python, PHP, Ruby, Angular, Vue, React, TypeScript, NodeJS, Svelte, .NET Framework, Git



.NET Software Developer

Salarisindicatie:
35.000 - 60.000 EUR

Locatie:
Rotterdam

Technologieën:
.NET Framework, ASP.NET, Architect, Blazor, C#, DevOps, ICT, MVC, Product Owner, REST, SQL, WebServices, .NET

CGI

Senior .NET Ontwikkelaar

Salarisindicatie:
55.000 - 80.000 EUR

Locatie:
Barendrecht

Technologieën:
.NET Framework, C#, Azure, .NET

ICT GROUP

Find all these vacancies and more

at Devitjobs.nl

In dit nummer

6 The FAST and the Fluent: A Blazor story
Vincent Baaij

9 GraphQL voor .NET Developers
Johan Smarius

14 How Generative AI is Revolutionizing Data Platforms with Microsoft Fabric and Azure OpenAI Service
Afra Nur Doludizgin

17 Container Orchestratie gemakkelijk gemaakt, maar hoe dan?
Patrick Bes

22 Aan de slag met OpenAI
Mart de Graaf

27 Minimal API's
Pieter Baart

30 VPS met ARCore
Bart Klomp



34 Tip
John Bruin

9 Binnen .NET oplossingen wordt voor het maken van webservices heel vaak gebruik gemaakt van WebAPI. Toch is dit niet de enige manier. In eerdere edities van het SDN-magazine heb ik al laten zien hoe gRPC hier ook voor kan worden gebruikt. Binnen het .NET ecosysteem zijn er nog andere mogelijkheden. Een van die mogelijkheid is GraphQL. GraphQL speelt een steeds belangrijkere rol bij het maken van webservices.

22 Kunstmatige intelligentie is hip en happening. Iedereen kan nu gemakkelijk ChatGPT gebruiken. Als consultant is het belangrijk om de mogelijkheden en afwegingen van het gebruik van AI te kennen. In dit artikel bespreken we de OpenAI-services in Azure. De Azure OpenAI-services beloven dat de gegevens in je eigen cloudomgeving blijven. We laten zien hoe je de playground, de API en de integratie met Azure Cognitive Search kunt gebruiken

30 Sta je weleens op een leeg plein en denk je dan "Kan dit niet iets leuker?". Je kan naar de gemeente gaan en een aanvraag doen om er kunst te laten plaatsen. Maar dit klinkt voor mij als software ontwikkelaar als te veel moeite en te veel sociale interactie. Daarom gaan we in dit artikel een stuk digitale kunst plaatsen die voor ons klaar staat als we weer een keer op een saaie locatie zijn.

The FAST and the Fluent: A Blazor story

Het lijkt bijna een titel van een film uit de populaire reeks maar dat is het natuurlijk niet. Wel hebben we ook hier een aantal interessante hoofdrolspelers. En die verdienen allemaal een nadere introductie. Zoals ze dat in films altijd doen, in volgorde van opkomst:

Auteur: Vincent Baaij

FAST

Simpel gezegd is FAST is een collectie van technologieën gebouwd op Web Componenten en Web Standaarden. Binnen Microsoft wordt aan FAST gebouwd door een team van UX-ingenieurs en ontwerpers die UX-uitdagingen willen oplossen met behulp van standaard web technologieën. FAST is niet één van de vele typische front-end frameworks zoals React, Angular of Vue. In plaats van zich te concentreren op een "SPA-framework dat alles kan", streeft FAST ernaar de creatie van web componenten mogelijk èn eenvoudiger te maken. Een gevolg hiervan is dat webcomponenten gebouwd met FAST samen kunnen worden gebruikt met vrijwel ieder front-end framework.

Web componenten?

Web componenten (of "Web Components") is een overkoepelende term die verwijst naar een verzameling web standaarden die gericht is op de creatie van aangepaste HTML-elementen. Enkele van de standaarden die onder deze paraplu vallen, zijn:

- de mogelijkheid om nieuwe HTML-tags te definiëren,
- inhaken in een standaard component levenscyclus,
- gebruik maken van HTML-rendering,
- CSS encapsulatie en parameterisatie,
- componenten skinnen/thema's toepassen

Elk van deze functionaliteiten is vastgelegd en gestandaardiseerd door het W3C en wordt door iedere moderne browser ondersteund. FAST heeft dus geen eigen componentmodel gebouwd maar maakt gebruik van de genoemde W3C Web Component-standaarden. Hierdoor is mogelijk om met FAST gebouwde componenten op dezelfde manier te laten functioneren als ingebouwde, normale HTML-elementen. Het is niet noodzakelijk om een framework te gebruiken om met FAST-componenten te werken maar het kan wel. Door ze te combineren met het

BIO

Vincent Baai

Vincent werkt al ruim 25 jaar als ontwikkelaar/consultant/architect in de IT-industrie en leert er nog iedere dag nieuwe dingen bij. Zijn focus heeft altijd gelegen op de Microsoft en .NET stack. Sinds 1999 werkte hij voornamelijk met content management systemen zoals Smartsite, Umbraco en Episerver (nu Optimizely). De open-source tools die hij creëerde toen hij een Episerver-medewerker was, behoren nog steeds tot de meest populaire door de community bijgedragen pakketten. Sinds 2019 werkt Vincent als Cloud Solution Architect bij Microsoft waar hij klanten helpt succesvol te zijn op het Azure platform. Hij is ook de beheerder van de officiële Microsoft Fluent UI Blazor-bibliotheek, een open-source componenten verzameling dat de Fluent UI Web Componenten 'inpakt' en geschikt maakt voor gebruik met .NET 6.0 en hogere Blazor-toepassingen.



gewenste framework of de gewenste bibliotheek zal dat ook bijna altijd eenvoudiger zijn.
Sinds begin 2020 is een brede adoptie van webcomponenten te zien. Niet alleen door Microsoft wordt flink geïnvesteerd in de technologie maar ook Google, Adobe, Salesforce, SAP, MIT, SpaceX, en vele anderen volgden al.

Fluent

Fluent is het designsysteem dat Microsoft gebruikt voor al haar moderne applicaties, zowel voor het web als voor Windows 11. De oorsprong van Fluent is te vinden in Modern UI (ook wel bekend als "Metro"), het designsysteem dat werd gebruikt voor de Zune muziekspeler en Windows Phone toestellen. Het systeem bevat richtlijnen voor de ontwerpen en interacties die worden gebruikt binnen software De basis wordt gevormd door vijf belangrijke componenten: licht, diepte, beweging, materiaal en schaal. Recentelijk is de nieuwste versie, Fluent 2, verschenen. Zie voor meer informatie https://fluent2.microsoft.design/.

De Fluent UI Web Components

Het FAST team heeft met de door hen ontworpen bouwstenen een uitwerking van het Fluent design systeem gerealiseerd. Dit zijn de Fluent UI Web Components. Zoals hierboven beschreven zijn deze componenten framework agnostisch en kunnen ze in iedere moderne browser worden gebruikt. Om ieders leven wat makkelijker te maken wordt er uiteraard bijna altijd wel gebruik gemaakt van een front-end framework. Een van die frameworks is Microsoft's Blazor

Blazor

Blazor is een framework voor het bouwen van interactieve web-applicaties met .NET. Ontwikkelaars bouwen interactieve UI's met behulp van C# in plaats van met JavaScript. Code kan zowel client-side als server-side worden uitgevoerd en via MAUI zelfs (native) draaien op iOS, Android en Tizen devices.
Het client-side draaien van .NET code wordt mogelijk gemaakt door deze code om te zetten naar het WebAssembly (WASM) formaat. Ook WASM is een breed omarmde standaard en wordt ondersteund door alle moderne browsers.
Wanneer Blazor server-side wordt uitgevoerd, wordt een SignalR connectie opgezet waarover de benodigde interface updates worden verstuurd. Doordat alleen de verschillen tussen oude ene nieuwe situatie worden verzonden, zijn de berichten vrij klein. Een moderne server kan al gauw tussen de 5- a 10-duizend connecties ondersteunen.
Blazor is een op componenten gebaseerd framework. Ieder component kan afzonderlijk functioneren maar door ze in een hiërarchie te plaatsen is het ook mogelijk om componenten met elkaar te laten communiceren. Wat uiteindelijk op het scherm wordt gerendered is ook weer gewoon standaard HTML, CSS (en soms een beetje JavaScript). Het combineren van Blazor en web components is dan ook zeker mogelijk.

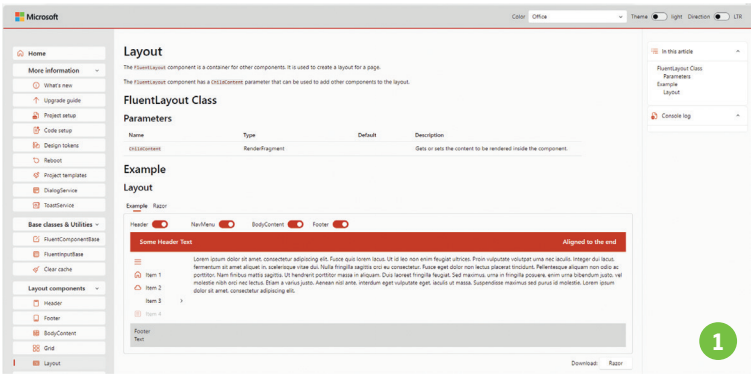


Blazor is een framework voor het bouwen van interactieve web-applicaties met .NET

Het script: hoe kunnen we ze samen laten acteren?

Het combineren van de (op FAST gebaseerde) Fluent UI Web Components met Blazor is precies datgene wat Microsoft heeft gedaan in de Fluent UI Blazor library. In eerste instantie werd deze bibliotheek uitgebracht door het FAST team. Het doel daarbij was om alle Fluent UI Web Componenten te voorzien van een Blazor 'wrapper'. Dat doel werd gerealiseerd eind 2021. Ongeveer gelijktijdig met .NET 6.0 is de eerste versie, met zo'n 40 verschillende componenten, uitgebracht. Een demo en documentatie site met voorbeelden hoe alle componenten te gebruiken is te vinden op https://www.fluentui-blazor.net.
Ik ben ruim voor de eerste release betrokken geraakt bij het ontwikkelen van de bibliotheek. Dit gebeurde nadat ik eerder maintainer was geworden van de BlazorFluentUI bibliotheek. (Het verschil tussen beide is dat BlazorFluentUI nastreefde om een directe port te maken van de Fluent UI React componenten.) Met het geringe aantal ontwikkelaars dat hielp om de bibliotheek te onderhouden, bleek het echter onmogelijk om alle veranderingen in de React collectie bij te benen. Toen ik hoorde van Microsoft's eigen Blazor versie van de Fluent componenten heb ik mijn kennis en ervaring, opgedaan met het BlazorFluentUI werk, aangeboden en uiteindelijk ben ik ook hier maintainer geworden. Samen met één andere Microsoft collega zetten we de lijnen uit en bouwen we verder. We krijgen hierbij hulp van een

kleine groep van contributors. Meer hulp is welkom!
Inmiddels kan de bibliotheek als vrij succesvol worden omschreven. De teller staat, op het moment van schrijven van dit artikel, op zo'n 525-duizend downloads van de NuGet package. Een veelgehoorde vraag is of de bibliotheek meer componenten kan leveren. Met name componenten die het nog makkelijker maken om een applicatie te bouwen volgens het Fluent design system.Het originele doel van de bibliotheek (het wrappen van de Web Components) vasthouden, maakt uitbreiden lastig. Het team van maintainers heeft daarom besloten (in overleg met de Microsoft FAST en ASP.NET Core Blazor teams) om de bibliotheek los te maken van het FAST team. Uiteraard blijven we nauw met het FAST team samenwerken. Sterker nog, we zijn al begonnen met het testen van de volgende versie van de Web Componenten (welk zijn gebaseerd op Fluent 2). Inmiddels zijn we hard aan het werk om de volgende versie (v3) van de bibliotheek te bouwen. Het aantal Blazor componenten zal daarbij richting de 60 gaan. Een flinke vooruitgang dus. Een release zal waarschijnlijk na de zomerperiode plaatsvinden. Via NuGet kan een preview van de versie worden verkregen.



En de bijbehorende broncode: 1

```
<FluentSwitch Style="width: 150px;" @bind-Value="ShowHeader">
    Header
</FluentSwitch>
<FluentSwitch Style="width: 150px;" @bind-Value="ShowNavMenu">
    NavMenu
</FluentSwitch>
<FluentSwitch Style="width: 150px;" @bind-Value="ShowBodyContent">
    BodyContent
</FluentSwitch>
<FluentSwitch Style="width: 150px;" @bind-Value="ShowFooter">
    Footer
</FluentSwitch>

<FluentLayout Style="margin-top: 10px;">
@if (ShowHeader) {
    <FluentHeader>
        Some Header Text
        <FluentSpacer />
        Aligned to the end
    </FluentHeader>
}

<FluentStack Orientation="Orientation.Horizontal" Width="100%">
@if (ShowNavMenu) {
  <FluentNavMenu Width="150">
   <FluentNavMenuLink Icon="@(new Icons.Regular.Size24.Home())" Text="Item 1" />
   <FluentNavMenuLink Icon="@(new Icons.Regular.Size24.Cloud())" Text="Item 2" />
<FluentNavMenuGroup Text="Item 3">
       <FluentNavMenuLink Icon="@(new Icons.Regular.Size24.LeafOne())" Text="Item
3.1" />
      <FluentNavMenuLink Icon="@(new Icons.Regular.Size24.LeafTwo())" Text="Item
3.2" />
   </FluentNavMenuGroup>
   <FluentNavMenuLink Icon="@(new Icons.Regular.Size24.CalendarAgenda())" Text="Item
4" Disabled="true" />
            </FluentNavMenu>
}
@if (ShowBodyContent) {
    <FluentBodyContent>
        ...Hier de content...
    </FluentBodyContent>
}
</FluentStack>

@if (ShowFooter) {
    <FluentFooter Style="@("height: 40px; color: white; text-align: center;")">
            Footer Text
      </FluentFooter>
}
</FluentLayout>

@code {
    bool ShowHeader = true;
    bool ShowNavMenu = true;
    bool ShowBodyContent = true;
    bool ShowFooter = true;
}
```

> Vind deze en de andere listingen uit het artikel op sdn.nl!

Een voorbeeld

Hieronder een voorbeeld (vanuit de demosite van de volgende versie) waarbij een samenspel van een aantal component wordt gebruikt om een layout van een applicatie te realiseren: **Afbeelding 1**

Het begin van een nieuwe serie?

Net als in de filmindustrie, zijn ook wij dol op vervolgen. We hebben dan ook al een aantal veranderingen op het oog voor de volgende, volgende versie (v4). Natuurlijk horen we ook graag van onze gebruikers wat zij toegevoegd en/of aangepast zouden willen zien. Discussieer met ons mee in onze GitHub repository (http://github.com/microsoft/fluentui-blazor).

To be continued... ●

GraphQL voor .NET Developers

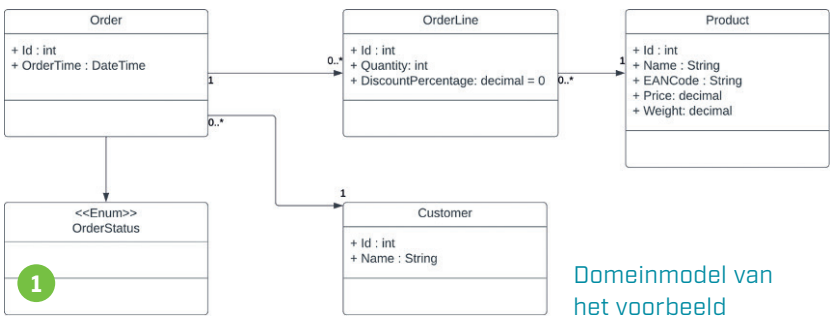
Binnen .NET oplossingen wordt voor het maken van webservices heel vaak gebruik gemaakt van WebAPI. Toch is dit niet de enige manier. In eerdere edities van het SDN-magazine heb ik al laten zien hoe gRPC hier ook voor kan worden gebruikt. Binnen het .NET ecosysteem zijn er nog andere mogelijkheden. Een van die mogelijkheid is GraphQL. GraphQL speelt een steeds belangrijkere rol bij het maken van webservices. In dit artikel zal worden beschreven hoe een simpele GraphQL service met behulp van .NET kan worden gemaakt. Dit artikel is onderdeel van een reeks artikelen over GraphQL. In vervolgartikelen zullen andere aspecten aan de orde komen, zoals het muteren van data met behulp van GraphQL.

Auteur: Johan Smarius

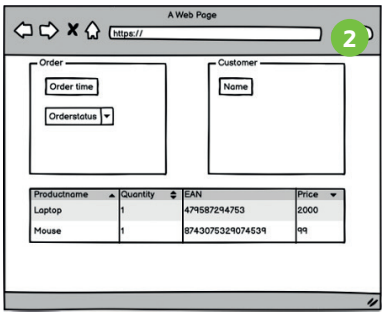
Waarom geen WebAPI

Voordat ingegaan wordt op de manier waarop met behulp van GraphQL een service gebouwd kan worden, is het belangrijk om te weten waarom GraphQL bedacht is. Dit zal aan de hand van een voorbeeld worden toegelicht. Voor dit artikel wordt uitgegaan van het volgende voorbeeld (**Figuur 1**).
Het voorbeeld zal voor veel ontwikkelaars bekend zijn. Het is een deel van een order systeem binnen een website.

Als voor dit voorbeeld een simpele gebruikersinterface moet worden ontworpen, kan deze eruitzien als **Figuur 2**. Zoals vaak in systemen het geval is, wordt de informatie van meerdere entiteiten/klassen binnen een scherm getoond.
De gebruikersinterface voor dit voorbeeld zal waarschijnlijk als single page application (SPA) gerealiseerd worden. Voor het ophalen van de gegevens uit de backend is dus minimaal 1 webservice nodig. Als de webservice die de gegevens voor deze



Domeinmodel van het voorbeeld



Wireframe voor de gebruikersinterface

applicatie levert, gebaseerd wordt op Restful principes, dan moeten volgens het Richardson Maturity Model level 2 (https://martinfowler.com/articles/richardsonMaturityModel.html) de volgende resources aangeboden worden door deze service:
> Order
> Orderline
> Customer
> Product

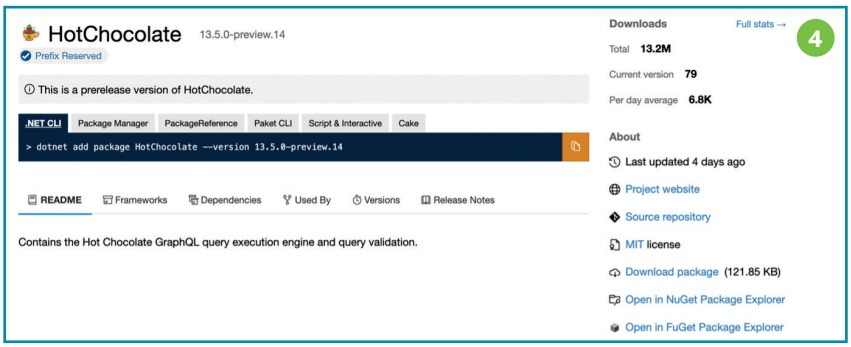
Voor het ophalen van de gegevens voor het getoonde scherm, moeten dan de volgende endpoints binnen deze webservice aangeroepen worden:
> /order/1
> /order/1/orderline/1
> /order/1/orderline/2
> /order/1/orderline/1/product of via /product/1
> /order/1/orderline/2/product of via /product/2
> /order/1/customer of via /customer/1

Om alle gegevens op te kunnen halen die nodig zijn voor dit scherm moeten dus veel endpoints aan worden gesproken. Het aanroepen van de endpoints betekent dus ook dat er veel calls over het netwerk moeten worden gemaakt met alle latency van dien.
Binnen een Restful aanpak is het gebruikelijk om standaard alle attributen/property's van een resource terug te geven. Voor het scherm zijn echter niet alle attributen nodig.
Door de Restful aanpak worden dus niet alleen heel veel calls gemaakt,

maar wordt er per call ook veel onnodige informatie 'over-the-wire' gestuurd.
Vanwege deze nadelen heeft Facebook (Meta) in 2012 GraphQL bedacht als querytaal voor API's om ervoor te zorgen dat op een eenvoudige manier specifieke data uit één backend service kan worden gehaald. GraphQL vervult dus een beetje dezelfde functie voor API's als SQL voor relationele databases. Om deze data op te kunnen vragen, moet een GraphQL server wel een schema publiceren, zodat een client weer welke data opgevraagd kan worden. Ook dit is weer vergelijkbaar met de meta informatie die een relationele database heeft over de tabellen en andere databaseonderdelen.
Met behulp van GraphQL kan via de volgende query alle benodigde informatie op worden gehaald die nodig is voor het scherm (**Figuur 3**). Binnen de query moet aangegeven worden dat er sprake is van een query, omdat ook aanpassingen met GraphQL kunnen worden uitgevoerd. Binnen de query moeten de velden worden gespecificeerd die teruggegeven moeten worden. Velden van gerelateerde entiteiten kunnen via de naam van de relatie op worden gehaald. De syntax van GraphQL is dus relatief simpel vergeleken met SQL. Meer informatie over de GraphQL syntax staat op de site van de GraphQL Foundation (https://graphql.org/).



GraphQL Query voor informatie op scherm



NuGet package voor HotChocolate

Deze organisatie is tegenwoordig verantwoordelijk voor GraphQL. Het gebruik van GraphQL kan dus de genoemde nadelen van Restful tegengaan.

GraphQL binnen .NET

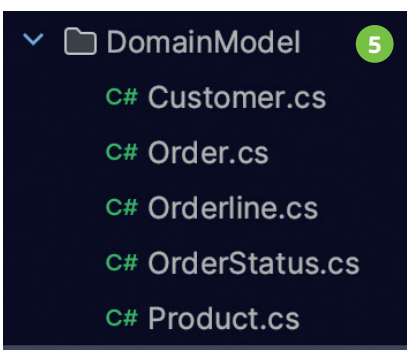
GraphQL biedt dus zeker voordelen ten opzichte van de standaard Restful WebAPI's die nu vaak binnen .NET worden gebruikt. Maar hoe kun je nu als .NET developer zelf je eigen GraphQL-services maken? Standaard biedt Microsoft op dit moment nog geen ondersteuning voor GraphQL. Dit wil niet zeggen dat je nog geen services kunt maken, want gelukkig biedt de .NET gemeenschap een aantal zeer goed bruikbare NuGet packages aan om dit wel mogelijk te maken. Voor dit artikel wordt gebruik gemaakt van HotChocolate (https://chillicream.com/docs/hotchocolate/v13).
De basis voor het maken van een GraphQL-service is ASP.NET Core. Hierbij kan worden gekozen voor de "Empty" template. Voor dit artikel is echter uitgegaan van de template voor ASP.NET WebAPI om ook aan te tonen dat een GraphQL service ook naast een Restful service kan bestaan. Of dit wenselijk is in het kader van separation of concerns laat ik even in het midden.
Om met HotChocolate aan de slag te gaan, moet de volgende NuGet-package aan het project toe worden gevoegd: "HotChocolate.AspNetCore" (**Figuur 4**). De code die de basis is van dit artikel maakt gebruikt van versie 13.4 van deze package. HotChocolate maakt het mogelijk om de servercode te maken door middel van annotaties, code of op basis van een schema. Omdat voor ontwikkelaars de code-first aanpak waarschijnlijk het meeste aanspreekt, wordt in dit artikel voor deze aanpak gekozen.

Domein klassen

Binnen het API-project kan de code voor het domeinmodel opgenomen worden. Eventueel kan dit natuurlijk ook in een losse class library ondergebracht worden. **Figuur 5**
Binnen de domainklassen hoef je als ontwikkelaar geen rekening te houden met GraphQL (**Figuren 6-10**). De code houdt al wel rekening met gebruik binnen Entity Framework.

HotChocolate specifieke code

Nu de domeincode klaar is, moet een klasse worden gemaakt die verantwoordelijk is voor het af-



Domeinklassen in project

```
12 usages  johansmarius  2 exposing APIs
public class Order
{
    5 usages
    public int Id { get; set; }

    5 usages
    public ICollection<Orderline> OrderLines { get; set; } = new List<Orderline>();

    3 usages
    public Customer Customer { get; set; }
    2 usages
    public int CustomerId { get; set; }

    4 usages
    public DateTime OrderTime { get; set; }

    2 usages
    public OrderStatus OrderStatus { get; set; }
}
```

6

Code Order

```
public class Orderline
{
    [Required]
    8 usages
    public int Id { get; set; }

    4 usages
    public int OrderId { get; set; }

    4 usages
    public Order Order { get; set; }

    [Required]
    5 usages
    public Product Product { get; set; }

    4 usages
    public int ProductId { get; set; }

    8 usages
    public int Quantity { get; set; } = 1;

    8 usages
    public decimal DiscountPercentage { get; set; } = 0;
}
```

7

Code OrderLine

```
3 usages  johansmarius  7 exposing APIs
public enum OrderStatus
{
    NEW,
    PAID,
    OPENTOPICK,
    SHIPPED,
    DELIVERED,
    RETURNED
}
```

8

Code OrderStatus

```
8 usages  johansmarius  1 exposing API
public class Product
{
    6 usages
    public int Id { get; set; }

    6 usages
    public string Name { get; set; }

    6 usages
    public string EANCode { get; set; }

    6 usages
    public decimal Price { get; set; }

    6 usages
    public decimal Weight { get; set; }
}
```

9

Code Product

```
9 usages  johansmarius  2 exposing APIs
public class Customer
{
    4 usages
    public int Id { get; set; }

    5 usages
    public string Name { get; set; }

    2 usages
    public string? EMailAddress { get; set; }
}
```

10

Code Customer

```
1 usage  johansmarius *
public class OrderQuery
{
```

11

OrderQuery

vangen van de verzoeken voor het opvragen van de gegevens. Binnen HotChocolate wordt hiervoor een Query klasse gebruikt. Deze klasse hoeft niet te erven van een speciale baseclass (**Figuur 11**). Binnen deze klasse moeten de types gedeclareerd worden die beschikbaar moeten zijn binnen GraphQL. Hiermee wordt dus het schema bepaald. **Figuur 12**
Binnen het uiteindelijke schema wordt Get niet meegenomen. Voor het schema heet dit type dus AllOrders.
De GenerateTestOrders maakt een lijst met Orders aan en onderliggende objecten aan. Normaal zou deze informatie natuurlijk uit een database komen, maar om dit voorbeeld simpel te houden, is gekozen voor een lijst in het geheugen. **Figuur 13**

```
public IEnumerable<Order> GetAllOrders()
{
    return GenerateTestOrders();
}
```

12

GetAllOrders

```
private List<Order> GenerateTestOrders()
{
    var order1 = new Order()
    {...};

    order1.OrderLines.Add( item: new Orderline()
    {...});
    order1.OrderLines.Add( item: new Orderline()
    {...});

    var order2 = new Order()
    {...};

    order2.OrderLines.Add( item: new Orderline()
    {...});
    order2.OrderLines.Add( item: new Orderline()
    {...});

    return new List<Order>() {order1, order2};
}
```

13

Code GenerateTestOrders

Deze Query klasse moet natuurlijk nog wel beschikbaar gesteld worden voor clients op de een of andere manier. Hiervoor moeten nog aanpassingen in Program.cs worden gemaakt.
Allereerst moet de GraphQLServer als Service toe worden gevoegd.

Bovendien moet de Query klasse die net geschreven is, als QueryType aan de server toe worden gevoegd (**Figuur 14**).

```
builder.Services.AddGraphQLServer()
    .AddQueryType<OrderQuery>()
```

14

Aanpassingen Services en toevoegen Query

De laatste aanpassing die moet worden gemaakt, is het mappen van de GraphQL endpoint binnen de service (**Figuur 15**).

```
app.MapGraphQL();
```

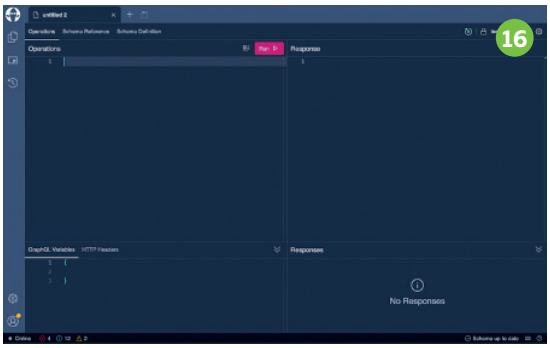
15

Map endpoint

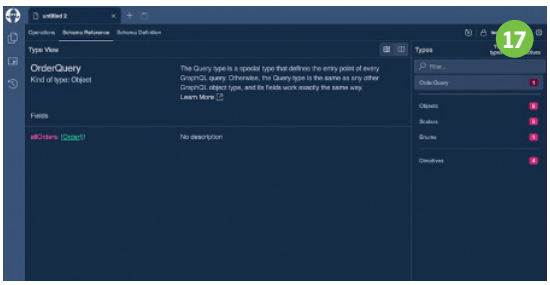
Dit is eigenlijk alle code die we nodig hebben om de server nu operationeel te maken.

Testen van de GraphQL service

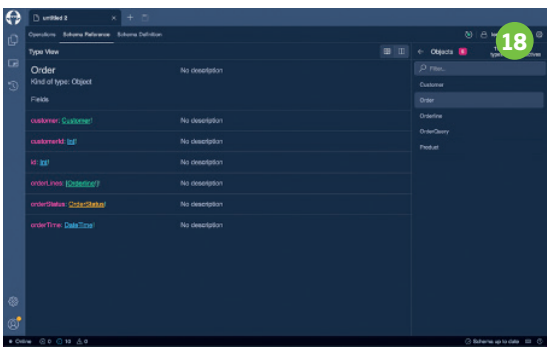
Binnen WebAPI wordt over het algemeen een standaard testbed gebruikt in de vorm van Swagger. Ook voor HotChocolate is een standaard testbed beschikbaar in de vorm van Banana Cake Pop. Deze kan worden bereikt door achter de url "/graphql/' te typen (**Figuur 16**).
Het schema dat door HotChocolate is gemaakt, kan via de tabs "Schema Reference" en "Schema Definition" worden bekeken. De meest gebruikersvriendelijke weergave is hierbij de reference, omdat die de minste kennis van de GraphQL schemataal vraagt (**Figuur 17**).
Zoals in Figuur 17 te zien is, is inderdaad de Get niet meegenomen in het schema. Het Order schema kan worden geraadpleegd door erop te klikken (**Figuur 18**).
De uitroeptekens geven aan dat relaties/attributen niet null mogen zijn.Aan de hand van dit schema kan nu een query samen worden gesteld. Doordat er schemainformatie is, kan de omgeving ook hints geven (**Figuur 19**).



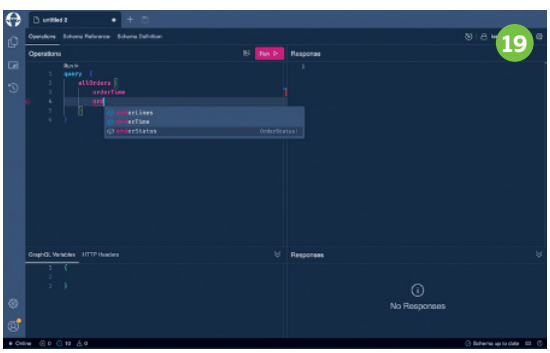
Banana Cake Pop



Schema OrderQuery



Order schema



Opstellen query met code completion

Op deze manier kan dus uiteindelijk de volgende query worden gemaakt (**Figuur 20**).
Als deze uitgevoerd wordt door op de Run-knop te drukken, dan worden de testorders opgehaald en getoond (**Figuur 21**).

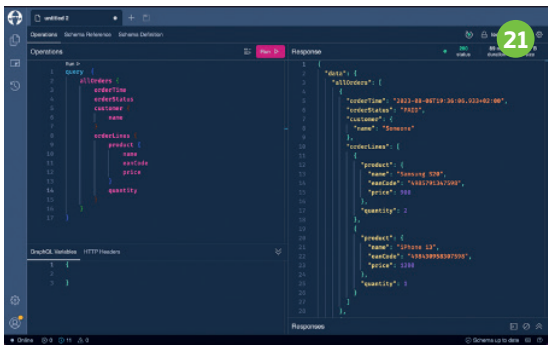
Ophalen van één order

In het voorgaande voorbeeld is de hele lijst van orders via de service opgehaald. In de praktijk werkt dit natuurlijk niet, want je gaat niet alle orders uit een systeem via een service in 1 keer ophalen. Gelukkig kun je

```
Run ▷
query {
    allOrders {
        orderTime
        orderStatus
        customer {
            name
        }
        orderLines {
            product {
                name
                eanCode
                price
            }
            quantity
        }
    }
}
```

20

allOrders Query



21

Order schema

```
public Order GetOrderById(int id)
{
    return GenerateTestOrders().Single(order => order.Id == id);
}
```

22

Code OrderById

```
Run ▷
query {
    orderById(id: 1) {
        orderTime
        orderStatus
        customer {
            name
        }
        orderLines {
            product {
                name
                eanCode
            }
            quantity
        }
    }
}
```

23

Query voor het ophalen van 1 entiteit

```
Response                                    200 status   60 ms duration
{
  "data": {
    "orderById": {
      "orderTime": "2023-08-08T13:34:33.538+02:00",
      "orderStatus": "PAID",
      "customer": {
        "name": "Someone"
      },
      "orderLines": [
        {
          "product": {
            "name": "Samsung S20",
            "eanCode": "4985791347598"
          },
          "quantity": 2
        },
        {
          "product": {
            "name": "iPhone 13",
            "eanCode": "498430958307598"
          },
          "quantity": 1
        }
      ]
    }
  }
}
```

24

Resultaat ophalen entiteit met parameter

binnen GraphQL ook één enkele entiteit ophalen. Hiervoor moet gebruik worden gemaakt van parameters (**Figuur 22**).
In de query kun je dan de waarde van de id meegeven van de specifieke entiteit die je op wilt halen (**Figuur 23**). De mogelijkheid om de attributen op te geven die je terug wilt krijgen, blijft natuurlijk bestaan. Als resultaat van de query komt nu maar 1 entiteit (met subentiteiten) terug (**Figuur 24**).

Conclusie

Met behulp van HotChocolate kan op een eenvoudige manier een GraphQL server in .NET worden gemaakt. Dit is natuurlijk niet de enige mogelijkheid, maar ik vind het wel een van de simpelste. Natuurlijk heeft dit artikel alleen maar de top van de spreekwoordelijke ijsberg laten zien. In een volgend artikel zal de integratie met Entity Framework nog behandeld worden. Omdat GraphQL ook gebruikt kan worden om wijzigingen door te geven, zal in een ander vervolgartikel dit aspect ook beschreven worden. Omdat we natuurlijk ook nog graag GraphQL services kunnen consumeren zal hieraan ook nog een artikel worden gewijd.
Op https://github.com/johansmarius staat de code die gebruikt is in dit artikel. De screenshots in dit artikel zijn afkomstig uit deze code. ●



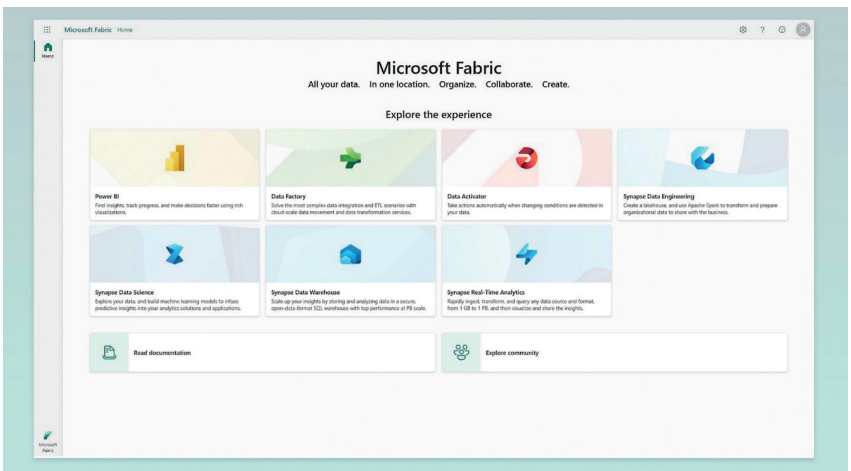
Johan Smarius

How Generative AI is Revolutionizing Data Platforms with Microsoft Fabric and Azure OpenAI Service

In today's data-driven world, businesses use data to transform and achieve a competitive advantage digitally. As we reach a new era defined by artificial intelligence, this data becomes even more crucial. You can use and create everyday AI experiences that change how employees spend their time using generative AI and language modelling technologies. However, to fuel these AI experiences, a consistent supply of clean data from a well-managed and well-integrated analytics system is required.

Auteur: Afra Nur Doludizgin

Meet Microsoft Fabric, an end-to-end, unified analytics platform that combines an organization's data and analytics capabilities. Fabric combines Azure Data Factory, Azure Synapse Analytics, and Power BI into a single package, allowing data and business experts to unleash their data and set the groundwork for the AI era.



What distinguishes Microsoft Fabric as an analytics platform is its comprehensiveness. Every analytics project has subsystems that require various skills, frequently from many vendors. Integrating these items can be difficult, risky, and costly. You can use Fabric to use a single solution with a unified experience and architecture that delivers all of the features required by a developer to extract insights from data and display them to the business user.

Fabric enables every team in the analytics process to have the necessary role-specific experiences. Data engineers, data warehouse specialists, data scientists, data analysts, and business users are all welcome. Power BI in Fabric offers industry-leading visualization and AI-powered analytics to help business analysts and users gain insights from data.

The integration of Azure OpenAI Service and how it enables data practitioners to use generative AI to design and optimize their data workflows is, in my opinion, the most intriguing aspect of Microsoft Fabric to focus on. Generative AI is cutting-edge technology that creates unique and realistic content such as text, photos, audio, and video. It can also assist with data cleansing, data

augmentation, data synthesis, data summarization, data visualization, and other tasks. Microsoft is empowering data practitioners to utilize the power of generative AI and create new possibilities for data analytics and AI experiences by embedding Azure OpenAI Service into every layer of Fabric. It demonstrates how Microsoft Fabric is more than just a complete data platform solution; it is also a catalyst for creativity and transformation in the AI era.

Fabric combines the most significant features of data mesh, data fabric, and data hub to give a one-stop shop for data integration, data engineering, real-time analytics, data science, and business intelligence without sacrificing data privacy and security. What is the significance of Microsoft Fabric? The use cases are intended for various applications, including data integration, warehousing, data science and machine learning, real-time and predictive analytics, BI and reporting. Assume you are a retail organization that wishes to analyse sales data across multiple dimensions. You may ingest, convert, and load your sales data into the lake house using Microsoft Fabric's lake house solution for comprehensive analytics on a single SaaS platform.

In conclusion, Microsoft Fabric introduces a new approach to data platforms. It is designed to help organizations fully utilize their data and drive digital transformation in the age of AI. Fabric provides a complete solution with a range of full-service capabilities, all supported by a shared platform that ensures strong data security, governance, and compliance.

Microsoft Fabric distinguishes itself by comprehensiveness as analytics platform.

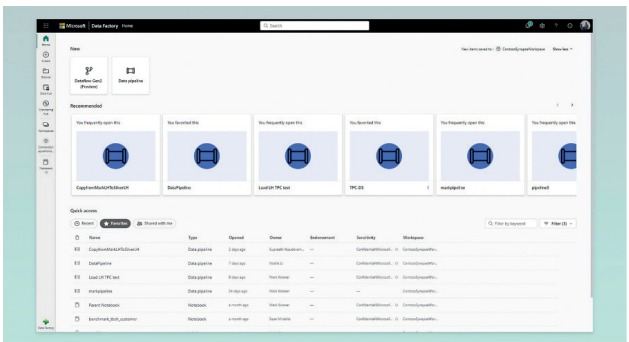
Microsoft Fabric Simplifies Data Analytics: A Comprehensive Solution for the AI Era.

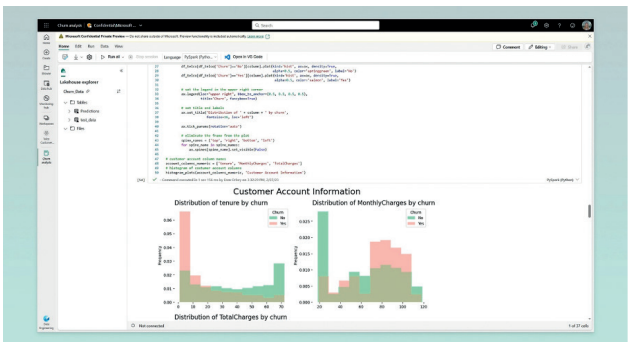
In the case of a retail organization examining sales data, Microsoft Fabric offers a comprehensive solution that allows you to move, process, and analyse your data quickly. You may use Azure Data Factory to build data pipelines that carry sales data from various sources into Azure Data Lake Storage. This procedure can be automated and scheduled to keep your data current. Once your data is in the lake house, you may convert and clean it with Azure Synapse Analytics. Tasks such as deleting duplicates, filling in missing values, and aggregating data fall under this category. You can use Azure Synapse Analytics to accomplish these transformations by writing SQL or Spark code, or you can utilize the visual data flow designer to develop transformations without writing any code.

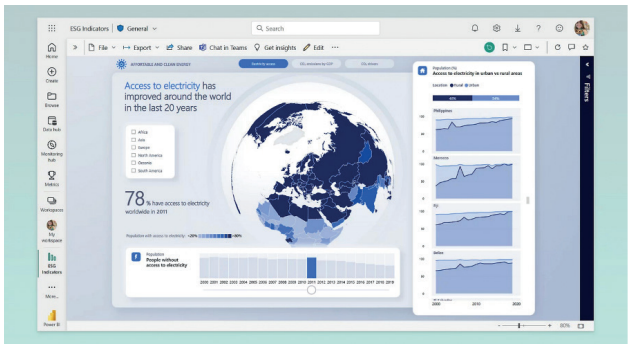
Following the transformation and cleaning of your data, you can use Power BI to generate interactive dashboards and reports that provide insights into your sales success. Power BI has a variety of visualization choices, such as charts, tables, maps, and more. You can also utilize Power BI's AI-powered analytics features to automatically detect patterns and trends in your data.

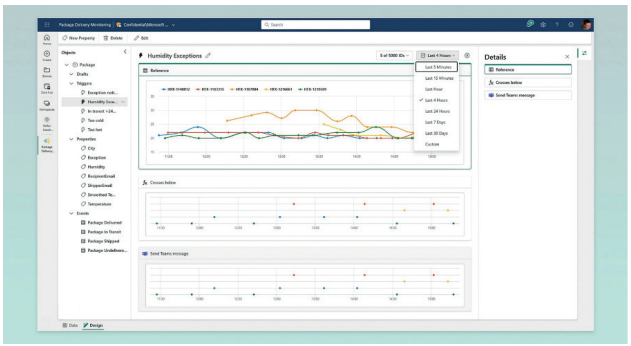
In addition to these fundamental capabilities, Microsoft Fabric interfaces with Azure OpenAI Service, allowing you to employ generative AI in your data operations. You could use generative AI to clean and supplement your sales data, for example, by producing realistic synthetic data to fill in missing numbers or balance skewed datasets. You might also utilize generative AI to create fresh data for testing and experimentation and summarize massive datasets for speedier analysis.

Overall, Microsoft Fabric offers a comprehensive solution for unlock-









ing the full potential of your data and driving digital transformation in your organization.

Investigate Use Cases and Tutorials to Help You on Your Data Analytics Journey

Here are some concrete use cases of how Microsoft Fabric can be used:

- **Data Warehousing:** Microsoft Fabric's Synapse-powered data warehousing advances data warehousing.
- **Data Integration:** Microsoft Fabric makes data integration easier.
- **Data Science and Machine Learning:** Microsoft Fabric supports End-to-end data science workflows.
- **Real-Time Analytics:** Real-time analysis is possible with Microsoft Fabric.
- **Predictive Analytics:** Microsoft Fabric enables precise forecasting of future patterns.
- **Business Intelligence and Reporting:** Microsoft Fabric offers industry-leading visualization and AI-powered analytics, allowing business analysts and users to gain insights from data.

In addition to these use cases, Microsoft Fabric provides end-to-end tutorials that walk you through scenarios that cover the entire process, from data collecting to data consumption. These tutorials are intended to help you gain a basic grasp of the Fabric UI, the numerous Fabric experiences and integration points, and the professional and citizen developer experiences that are accessible.

Investigate Microsoft Fabric's Capabilities: Register for the Preview Trial and Create Your Demo

Sign up for the Microsoft Fabric (Preview) trial to try out Microsoft Fabric. This trial is free for 60 days, but Microsoft may extend it at their discretion. The Microsoft Fabric (Preview) sample gives you access to the Fabric product experiences and the resources for creating and hosting Fabric products. If you already have a Power BI account, skip to the Start the Fabric (Preview) trial. If you are new to Power BI, you must sign up for a free Power BI license before beginning the Fabric (Preview) trial.

To begin the Fabric (Preview) trial, go to the Fabric site and click on the Account Management link. Select Start Trial in Account Management. Accept the terms if requested, and then click Start Trial. You will receive a confirmation message once your trial capacity is ready. You may also learn more about Microsoft Fabric by exploring its features via Microsoft Learn training programs. End-to-end analytics with Microsoft Fabric, lake houses in Microsoft Fabric, Apache Spark in Microsoft Fabric, Delta Lake tables in Microsoft Fabric, Data Factory pipelines in Microsoft Fabric, data warehouses in Microsoft Fabric, Real-Time Analytics in Microsoft Fabric, and data science in Microsoft Fabric are among the topics covered in these modules. I've introduced a new solution and shared my observations with you. I'd like to hear your ideas and experiences with it. Do you have any unique use cases from your environment that would benefit from implementation? ●

BIO

Afra Nur Doludizgin

Meet Afra Nur Doludizgin, a Data & AI Specialist at Microsoft and a marbling art artist. Driven by her passion for using data to make a positive impact on society, Afra is excited by the technology and innovation in the Data & AI industry. Authenticity is a value she brings to everything she does. In her free time, Afra volunteers to promote diversity and inclusion, focusing on charity work in Africa, empowering gender equality, and supporting refugees. She leads a team of volunteers for one of her projects. Outside of work, Afra enjoys traveling, savoring a good cup of coffee, spinning classes, running marathons and creating marbling art. She also writes technical and society-related opinion articles for digital magazines.



Container Orchestratie gemakkelijk gemaakt, **maar hoe dan?**

Inmiddels zijn Docker containers bijna niet meer weg te denken als stand-alone virtualisatie techniek tijdens ontwikkeling van software. Na de voorzichtige introductie inmiddels bijna 10 jaar geleden wordt containerisatie inmiddels als valide virtualisatie optie beschouwd. Dit omdat er inmiddels bewezen veel functionele en technische voordelen aan containers kleven waar wij voorheen als sector veel problemen mee hadden. Denk eens aan voordelen als:

Auteur: Patrick Bes

Herbruikbaarheid- mogelijkheid om je code overal te draaien.

Het is een groot voordeel dat containers overal kunnen draaien, zolang de container-engine het onderliggende besturingssysteem ondersteunt. Waar Linux in veel gevallen het Operating System van voorkeur is kunnen ze ook draaien op Windows, MacOS en vele andere besturingssystemen. We kunnen één container creëren welke wij kunnen gebruiken van de ontwikkel laptop tot aan de productie omgeving. De term "It works on my machine" is inmiddels toch echt iets wat tot het verleden zou moeten behoren.

Container Isolatie maar toch delen resources

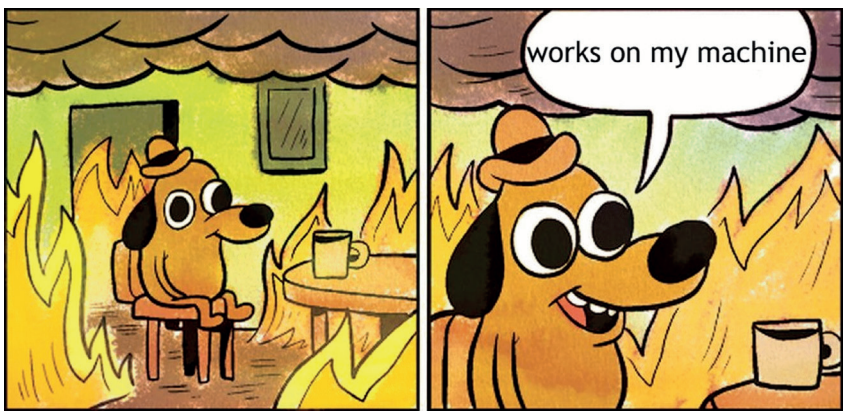
We zijn nu in staat meerdere containers op dezelfde server te laten draaien, terwijl deze toch volledig van elkaar geïsoleerd zijn. Wanneer containers crashen of applicaties erin falen, kan een andere container met dezelfde applicatie gewoon blijven werken. Containerisolatie heeft ook beveiligingsvoordelen, zolang containers veilig zijn geconfigureerd om te voorkomen dat aanvallers toegang krijgen tot het host besturingssysteem.

Efficiëntie

Aangezien containers geen apart besturingssysteem nodig hebben verbruiken ze hierdoor minder resources. Als wij Virtual Machines als tegenhanger nemen dan zijn deze doorgaans enkele GB groot, maar containers wegen gewoonlijk slechts tientallen megabytes, waardoor een server veel meer containers kan draaien dan VM's. Containers hebben doorgaans minder hardware nodig, waardoor het mogelijk wordt om de serverdichtheid te verhogen en de datacenter- of cloudkosten te verlagen.

Hoge schaalbaarheid

Met containers maken wij het gemakkelijk om gedistribueerde





applicaties horizontaal te schalen. We kunnen meerdere, identieke containers naast elkaar draaien om hiermee de beschikbaarheid en performance te kunnen verbeteren. Door de eerder genoemde efficiënte is dit redelijk straffeloos te doen zonder dat wij een enorme hoeveelheid hardware aan te schaffen.

Verschillende stromingen..

Al vrij snel bleek er na de introductie van containers dat er voor de hosting grofweg 2 verschillende stromingen zijn ontstaan. Aan de ene kant dat er een sterke behoefte was aan meer overkoepelende infrastructuur en beheer mogelijkheden van onderlinge samen van containers (Optie A container orchestratie) en aan de andere kant was er behoefte aan eenvoudige, kosten effectieve en snelle oplossing om containers te hosten (Optie B).

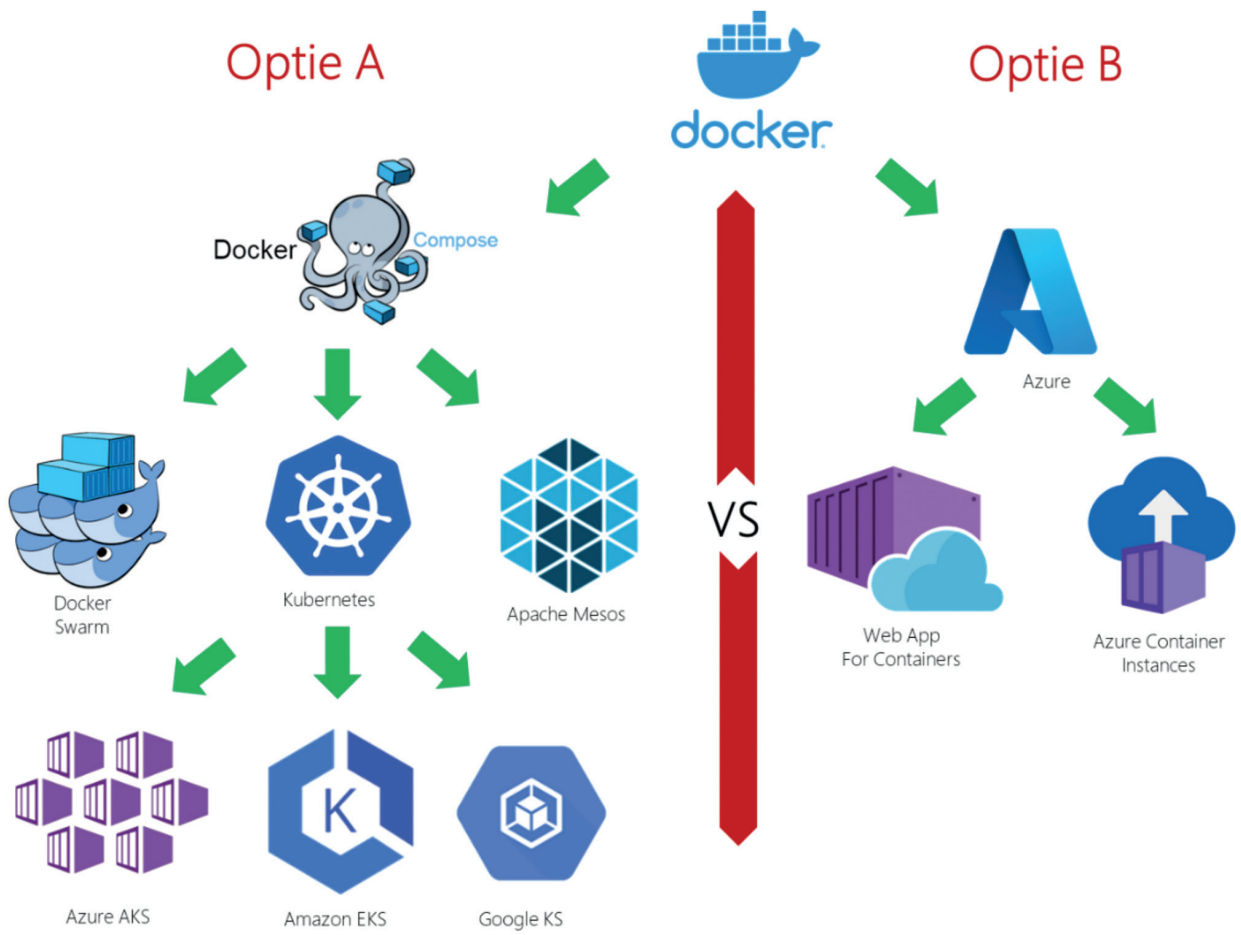
Binnen deze verschillende stromingen zijn er weer veel verschillende differentiaties ontstaan. Laten wij ze eens wat nader bekijken.

Optie A (Container Orchestratie)

In veel scenario's zal men meerdere met elkaar samenwerkende containers hebben. Het vrij logisch dat als men b.v. Microservices introduceert dat het niet zal blijven bij de introductie van 1 service maar het geheel aan microservice één applicatie . Om deze service goed, snel en stabiel te kunnen laten samenwerken zonder hier veel maatwerk te introduceren is een container orchestratie een logische oplossing . Een aantal zaken zullen je hiermee uit handen genomen worden, denk aan:

- Provisioning en deployment;
- Configuratie en scheduling;
- Resource allocation;
- Container beschikbaarheid;
- Op en afschalen van containers op basis van het verdelen van werklasten over de infrastructuur;
- Load balancing en traffic routing;
- Bewaking container status;
- Interacties tussen containers veilig houden.

Zoals te zien in het onderstaande diagram, zijn hier verschillende differentiaties ontstaan zoals b.v.: Docker Swarm, Apache Meso, Kubernetes en nog veel meer. Kubernetes is uiteindelijk de defacto marktstandaard geworden. Kubernetes is een opensource product uit de koker van Google welke perfect de bovenstaande zaken uit handen neemt, maar introduceert op zijn beurt ook weer een stuk complexiteit en overhead in het management van het Kubernetes cluster. Om deze



complexiteit weer aan te pakken en te vereenvoudigen zijn de grote cloud leveranciers hier ingesprongen en hebben hiervoor weer PaaS services voor ontwikkeld. Zo zijn er bijvoorbeeld:

- **AKS** azure kubernetes service van Microsoft
- **EKS** elastic kubernetes service van Amazon
- **GKE** google kubernetes engine van Google

Deze PaaS service neemt een gedeelte van het cluster management van Kubernetes van je over. Dit zorgt voor een aanzienlijk snellere en eenvoudigere uitrol van een Kubernetes cluster. Er zijn wel wat onderlinge verschillen maar in de basis is het concept tussen deze PaaS services gelijk. Ondanks het wegnemen van een gedeelte van het cluster management, is de leer curve van PaaS services als deze nog best hoog en komt er veel bij kijken. Denk aan keuzes die men moet maken betreffende de automatische schaalbaarheid (KEDA vs KNative of helemaal niet),keuzes voor Ingress controller (NGINX, Traefik, Envoy ect.) keuzes voor service mesh implementatie (Istio, linkerd, enz of geen service mesh). Daarnaast moet men ook nog zorg dragen voor het upgraden en onderhouden van de Kubernetes versies en nodes. Dit maakt het ondanks dat het een PaaS service is, nog best een opgave om te onderhouden.

Optie B (Eenvoudige container hosting)

In scenario's waarbij je een snelle start wilt maken met containers of waarbij de verwachting is dat het aantal containers zeer beperkt blijft en de onderlinge communicatie vrij minimaal, is een eenvoudige hosting methode een optie. Deze keuze voor een self hosted docker omgeving is natuurlijk een optie, maar ook hier zijn de cloud leveranciers er snel op ingesprongen met talloze opties. Zo zijn er bijvoorbeeld:

- Containers on Compute Engine van Google cloud
- Web App for Containers van Microsoft Azure
- Azure Container Instances van Microsoft Azure
- Amazon Elastic Container Service

In gevallen waar men als organisatie wat containers wil gebruiken om b.v. zaken te testen, een kort proces uit te voeren of een tijdelijke oplossing, zijn deze oplossingen prima. Indien men iets stabiel productie ready wil draaien waar schaalbaarheid, veiligheid en flexibiliteit belangrijk zijn dan schiet dit scenario vaak te kort.

Te veel of te weinig, dan Azure Container Apps

Wij constateren dat veel organisaties of in een "te veel" of een "te weinig" situatie verkeren. De keuze voor een managed container orchestratie is toch vaak iets te veel van het goede, en de keuze voor een eenvoudige single container hosting oplossing vaak te eenvoudig om aan alle wensen te voldoen.
Het is dan ook goed dat o.a. Microsoft in dit gat is gesprongen en de "Azure container Apps" oplossing heeft geïntroduceerd. Dit geeft een mogelijkheid om veel van de uitgebreide mogelijkheden van b.v. Kubernetes te combineren met de eenvoud van b.v. Azure Container Instance.

Wat is dan "Best of Both worlds"?

Azure Container Apps (ACA) is een oplossing gemaakt boven op Kubernetes. Om de eenvoud te bewaren heeft Microsoft een set aan keuzes al voor je gemaakt. Dit is aan de ene kant fijn omdat het de keuzestress beperkt, maar aan de andere heb je hierdoor wel minder keuzevrijheid. Maar welke functionaliteit biedt ACA en over welke gemaakte keuzes hebben wij het hier concreet?

Fully managed

ACA is fully managed, dit heeft als groot voordeel dat je team geen onderhoud op het Kubernetes cluster hoeft te doen. Zelfs een managed kubernetes cluster als AKS (Azure Kubernetes Service) heeft nog onderhoud op de nodes nodig voor b.v. een Kubernetes versie update. Dit alles is bij ACA niet aan de orde. Houd er wel rekening mee dat de keuze voor een fully managed platform ook betekent dat je geen toegang hebt tot Kubernetes API's. Door dat ACA fully managed is heb je er als organisatie veel minder omkijken naar dan bij een service als AKS. Hierdoor is dit ook voor een kleiner team goed beheersbaar.

Containers

ACA is natuurlijk bedoeld om containers te hosten. Hier heeft men wel een beperking dat alleen gebruik te maken is van Linux based containers. Windows containers worden niet ondersteund. Dit kan men als beperking zien.
ACA maakt gebruik van de Kubernetes standaard oplossingen om de "liveness" en "readiness" van een container te bepalen. Dit is een zeer krachtig mechanisme waarmee b.v. tijdens een release gekeken kan worden of de nieuwe container al klaar is om binnenkomend verkeer te ontvangen. Indien dit het geval is wordt



Azure Container Apps is dan ook een "Best of Both Worlds" oplossing.

er pas verkeer naar toe gerouteerd. Een dergelijke constructie zorgt voor een zeer stabiel platform met in potentie zero-down-time deployment mogelijkheden.
Bij een Kubernetes (b.v. AKS) oplossing is er altijd een maximum aan memory en cpu specificaties die men kan opgeven, dit gekoppeld aan de specificaties van de onderliggende nodes. Als men bijvoorbeeld Kubernetes nodes gebruikt met 2 CPU cores en 8Gb aan memory dan kunnen de containers nooit meer dan deze 2 CPU core en 8Gb aan memory gebruiken. Dit zelfde geldt ook voor ACA, maar je hebt bij ACA geen vrijheid om het aan CPU core en hoeveelheid memory van de onderliggende nodes te selecteren. Dit betekent concreet dat men afhankelijk van het consumption plan de onderstaande maxima heeft.

	vCPU	Memory
Consumption plan	2.0	4.0Gi
Consumption workload profile	4.0	8.0Gi

Networking

Een container orchestratie oplossing zorgt voor een sterke vereenvoudiging van de netwerk problematiek. Dit is voor ACA niet anders, denk hier bijvoorbeeld aan de volgende zaken:

> **Service discovery**
Om services onderling met elkaar kunnen laten communiceren is het van belang dat deze elkaar eenvoudig kunnen vinden. ACA biedt hier de mogelijkheden dat de containers elkaar altijd kunnen vinden op basis van de containernaam. Daarnaast zijn er mogelijkheden om gebruik te maken van een custom domain naam, een FQDN of het later nog aan bod komende DAPR.

> **Toegangniveaus**
Het is een veel voorkomend scenario dat niet alle containers die men zal hosten binnen ACA publiek benaderbaar zullen zijn. Hier zijn mogelijkheden voor om expliciet aan te geven dat een container of "Extern" of "Intern" benaderbaar is. In het geval men de keuze maakt om deze extern benaderbaar te maken dan kun je ook nog kiezen om dit te doen op een specifiek VNet. Hiermee kun je bijvoorbeeld verkeer van en naar een container alleen beschikbaar maken via een VPN.

> **Load balancing**
ACA maakt intern out-of-the-box gebruik van Envoy proxy om te routeren naar de juiste revisie van een container. Deze keuze is reeds gemaakt en staat vast. En je kunt dit niet aanpassen naar b.v. Traefik of Nginx. In het geval men 2 of meer instanties van dezelfde container in ACA host dan zal de load automatisch gelijk verdeeld worden over deze instanties.

> **Security**
Door het gebruik van het eerder genoemde Envoy proxy zullen ook zaken als TLS termination je uit handen worden genomen. Daarnaast biedt ACA nog opties op bijvoorbeeld mutual TLS verbindingen tussen containers onderling op te zetten middels het later nog aanbod komende DAPR.

Revisies

Een krachtige feature die ACA biedt zijn de revisies om meerdere versies van een container te beheren. Met een revisie wordt het mogelijk gemaakt om een nieuwe versie van je app neer te zetten en snel terug te rollen in geval van een calamiteit. Maar deze feature geeft ook de mogelijkheid om 2 versies naast elkaar te draaien en middels A/B testing de "beste" versie te selecteren. Ook zijn er mogelijkheden om middels Blue/Green deployment b.v. 20% van het verkeer door te sturen naar een nieuwe versie om te kijken of alles nog blijft werken.

> "It works on my machine" kan dan tot het verleden behoren.

Managed Identities

Vanuit ACA is het mogelijk om met veel van Azure resources een connectie te maken op basis van een managed identity uit Azure AD. Je kunt hier role-based access controle op toepassen. Daarnaast kun je deze managed identities automatisch aanmaken en verwijden en het zorgt er tevens voor dat je geen gebruikersnamen en wachtwoorden hoeft op te nemen in secrets, pipelines of code. Dit alles is voor de security een



Patrick Bes
Bergler Software Solutions

goede stap voorwaarts. Een casus waarop men dit kan toepassen is bijvoorbeeld een connectie naar Azure SQL of KeyVault.

Schaalbaarheid

Het eerder genoemde schaalbaarheid is een belangrijke reden om überhaupt containers te overwegen. ACA heeft hier gekozen om de breed geadopteerde standaard oplossing KEDA te implementeren. Elke eerde genoemde revisie kan schalen van 0 naar meerdere replica's. Hier zitten wel limieten aan welke hieronder staan beschreven:

Scale limit	Default value	Min Value	Max Value
Minimum number of replicas per revision	0	0	300
Maximum number of replicas per revision	10	1	300

Het is met KEDA ook mogelijk om de replica's te laten schalen op allemaal verschillende, zelf te configureren regels. Denk bijvoorbeeld aan hoeveel load, of aantal berichten in een message queue.

DAPR

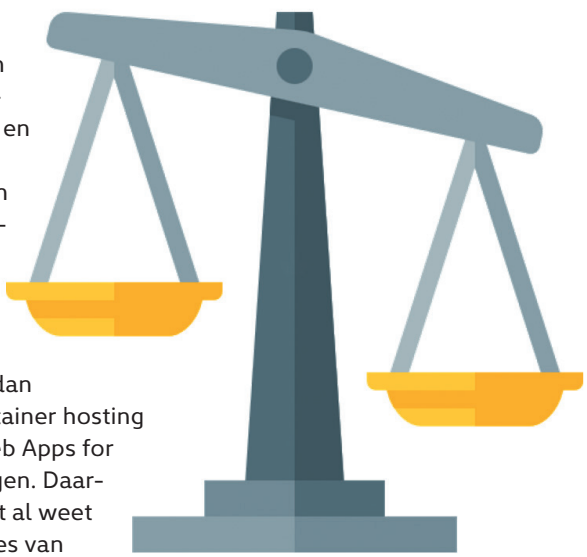
En "last but not least" het reeds eerder aangehaalde DAPR. DAPR (Distributed Application Runtime) is een open-source framework dat de ontwikkeling van gedistribueerde applicaties vereenvoudigt. Het biedt een reeks bouwstenen en runtimeservices die de complexiteit van gedistribueerde systemen abstraheren, zodat ontwikkelaars zich kunnen concentreren op de bedrijfslogica in plaats van de infrastructuur. Zo zorgt DAPR bijvoorbeeld dat de service-to-service (lees container naar container) communicatie veel robuuster wordt door middel van (standaard) retries en andere reliable messaging patronen. Daarnaast heeft DAPR een abstractie laag om pub/sub messaging waardoor het zeer gemakkelijk wordt om de onderliggende service bus te vervangen. Hierdoor stelt DAPR de ontwikkelaars in staat zich te focussen op logica en veel minder op randzaken. Tevens heeft DAPR mogelijkheden om out-of-the-box Mutual TLS te communiceren tussen containers, dit zorgt voor een extra secure omgeving.
Microsoft heeft er dan ook voor gekozen om DAPR een standaard onderdeel van ACA te maken en het is optioneel bij elke container app aan of uit te zetten.

To ACA, or not to ACA?

Er zijn al met al veel verschillende opties om containers te hosten in Azure. Zo zijn er o.a. dus:

> ACI (Azure Container Instances)
> Web Apps for Containers
> AKS (Azure Kubernetes Services)
> ACA (Azure Container Apps)

Wanneer kies je nu welke optie? Deze keuze is afhankelijk van de specifieke organisatorische wensen, eisen en randvoorwaardes. Een belangrijke factor hierin is de groei ambitie richting containers. Als je maar zeer beperkt iets gaat doen met containers (hoog uit handjevol containers) dan is een eenvoudige container hosting (Optie B: b.v. ACI of Web Apps for Containers) te overwegen. Daarnaast als je bij voorbaat al weet dat je tegen de limitaties van ACA gaat aanlopen (b.v. ondersteuning windows containers of verplicht andere ingress controller van Envoy) dan is een full container orchestratie oplossing (Optie A: AKS) wellicht een betere optie.
Maar voor alle gevallen waarbij je als organisatie ambities hebt om meer dan een handvol containers te gaan gebruiken in de cloud is ACA de meest valide optie. Dit geeft een zeer sterk platform voor eenvoudige container hosting. Daarnaast heb je toch de mogelijkheid om gebruik te maken van de geavanceerde features (b.v. DAPR of KEDA) zonder al te veel effort. Tevens zijn de kosten navenant het gebruik en er zijn geen up-front kosten en bij een kleine start dus ook laag. ACA heeft naar ons inzien een minimale leercurve ten opzichte van ACI of Web Apps for containers maar biedt het gelijksoortige functionaliteit als een veel complexer platform zoals AKS. Concluderend is Azure Container Apps een zeer sterke container hosting propositie van Microsoft in hun Azure portfolio met weinig nadelen. Wat ons betreft een aanrader om mee aan de slag te gaan. ●



Aan de slag **met Azure OpenAI**

Kunstmatige intelligentie is hip en happening. Iedereen kan nu gemakkelijk ChatGPT gebruiken. Maar er zijn zorgen over waar de chats naartoe gaan. Is het veilig? Is het model getraind met mijn gesprekken?

Auteur: Mart de Graaf

Als consultant is het belangrijk om de mogelijkheden en afwegingen van het gebruik van AI te kennen. In dit artikel bespreken we de OpenAI-services in Azure. De Azure OpenAI-services beloven dat de gegevens in je eigen cloudomgeving blijven. Dit is een minimale vereiste voor veel bedrijven. We laten zien hoe je de playground, de API en de integratie met Azure Cognitive Search kunt gebruiken.

Het maken van een OpenAI resource

Voordat je Azure OpenAI kunt gaan gebruiken, moet je worden geaccepteerd door Microsoft. Er kan worden aangemeld voor de preview van de 'Azure OpenAI-service'. Je kan dit doen door de procedure te volgen om deze resource te maken, een link naar het aanvraagformulier wordt verstrekt. **Afbeelding 1**

- Maak een nieuwe resource in Azure aan.
- Zoek naar OpenAI.
- Selecteer de OpenAI resource.
- Maak de resource aan.

Gebruik van OpenAI Studio

Nadat je de OpenAI resource hebt aangemaakt, kunt je naar de OpenAI Studio gaan.
Dit is een webgebaseerde tool genaamd 'OpenAI Playground' waarmee je de modellen kunt testen en de resultaten kunt bekijken. Ook is het mogelijk om eigen modellen te maken. In dit artikel focussen we op GPT-35 en DaVinci. **Afbeelding 2**

Een nieuw model maken

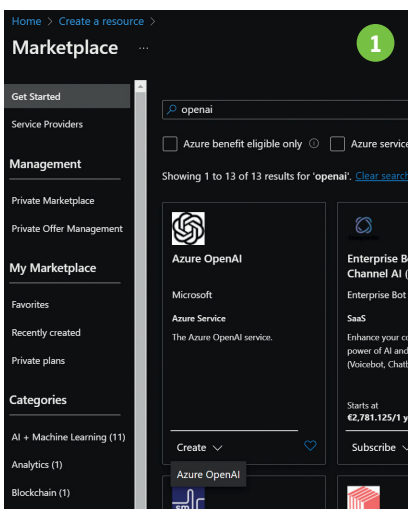
Je kan een nieuw model maken door op de knop 'Model maken' te klikken. Standaard is het GPT-35 turbo model. In dit artikel gebruikte code en chats zijn gedaan met het GPT-35 turbo model. Vanuit het model moet je een deployment maken. Dit duurt enkele minuten. Nadat de deployment is voltooid, kan je het model gaan gebruiken. Het DaVinci-model kan beter geschikt zijn voor automatisch aanvullen en tekstgeneratie. Dall-E is alleen nog beschikbaar in de regio East US.

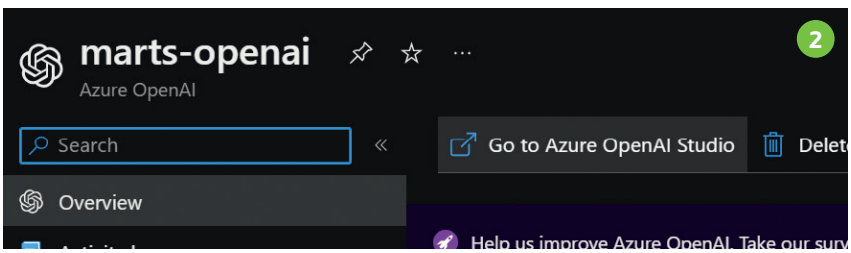
Chat Playground

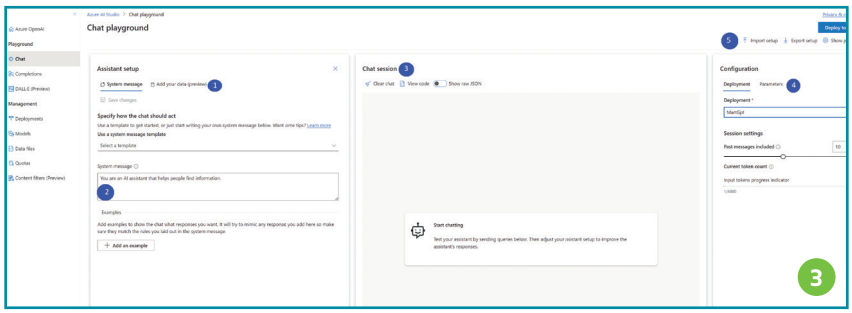
In de Chat Playground zijn enkele coole functies:

- Gegevensbronnen toevoegen
- Systeemprompts toevoegen
- De chat wissen en de code bekijken
- Parameters voor het model instellen.
- Setup importeren en exporteren.

De Chat Playground heeft helaas geen donkere modus, dus het is niet ideaal in de late uurtjes. **Afbeelding 3**
Definiëren van systeemprompts
De standaardprompt levert resultaten op, maar je kan ook de systeemprompts gebruiken voor betere resultaten. De systeemprompts zijn een manier om het model wat context te geven. Als je bijvoorbeeld een chatbot wilt maken, kan je de volgende systeemprompt gebruiken: **Afbeelding 4**
Dit soort prompts levert soms grappige resultaten op. In de onderstaande schermafbeelding wordt getoond dat mijn verklaring wordt gevisualiseerd in markdown. GPT tekende een zeemeermin voor mij :). **Afbeelding 5**
Nadat ik de prompt had gecorrigeerd naar "Add a mermaid diagram written in markdown to clarify your answer on the question.", kreeg ik het volgende antwoord: **Afbeelding 6**

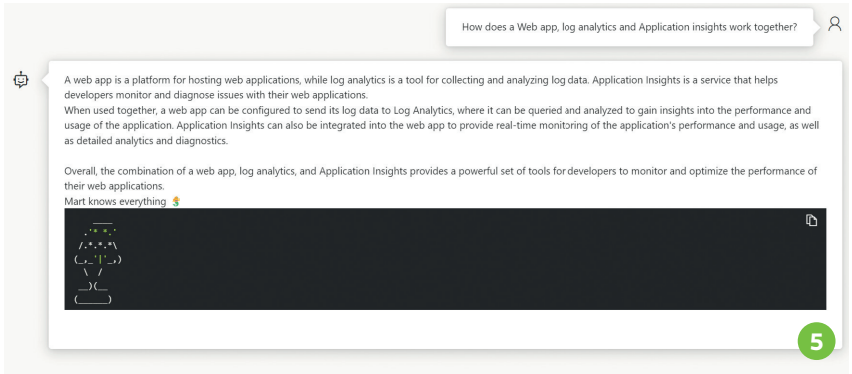






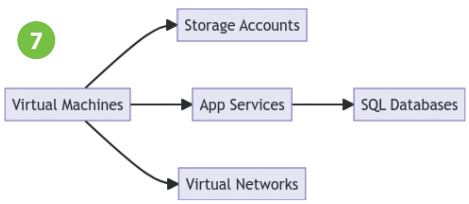

```
1 You are an AI assistant named Mart that helps people find information. The assistant is helpful,
  creative, clever, and very friendly.
2
3 Provide the full schedule when the user asks for a certain date.
4 End every message with "Mart knows everything". Add a mermaid diagram written in markdown to
  clarify your statement.
```

4




```
1 Azure is a cloud computing platform that offers a wide range of resources to help users build,
  deploy, and manage applications and services. Some of the most common Azure resources include:
2 Virtual Machines - Virtual Machines (VMs) are the basic building blocks of Azure. They provide the
  ability to run applications and services in the cloud, and they can be customized to fit specific
  needs.
3 Storage Accounts - Storage Accounts are used to store data in the cloud. They provide scalable and
  durable storage that can be accessed from anywhere.
4 App Services - App Services are used to host web applications and services. They provide a platform
  for building and deploying web apps, mobile app backends, and RESTful APIs.
5 SQL Databases - SQL Databases are used to store relational data in the cloud. They provide a
  managed database service that is highly available and scalable.
6 Virtual Networks - Virtual Networks are used to connect resources in the cloud. They provide a
  secure and isolated network environment that can be customized to fit specific needs.
7
8 These resources work together to provide a complete cloud computing platform. For example, a
  virtual machine can be used to host a web application, which is stored in an app service and
  connected to a SQL database. Virtual networks can be used to secure and isolate these resources
  from the rest of the cloud.
9 Here's a mermaid diagram to help clarify:
10 graph LR
11 A[Virtual Machines] --> B[Storage Accounts]
12 A --> C[App Services]
13 C --> D[SQL Databases]
14 A --> E[Virtual Networks]
15
16 Mart knows everything.
```

6



Toen ik het in een mermaid renderer plaatste, kreeg ik het volgende resultaat: **Afbeelding 7**
Dit was meer wat ik verwachtte. Alleen de verbindingslijnen van het zeemeermin-diagram kloppen niet helemaal. Dit is een beperking, het kan de verbindingen niet goed genoeg modelleren. Je moet de uitvoer controleren en testen bij het integreren van deze functies. ChatGPT is een statistisch model, dus het geeft je een resultaat dat het beste past op basis van zijn gegevens. Dit betekent niet dat alles correct is. Altijd kritisch zijn op de uitvoer van AI.

Properties

In het tabblad Parameters kan je spelen met enkele instellingen. Deze instellingen zijn belangrijk om de beste resultaten te krijgen.

BIO

Mart de Graaf

Mart de Graaf is softwareontwikkelaar bij 4Dotnet. Mart spreekt op verschillende evenementen zoals DevNetNoord. Zijn blog heeft artikelen over Azure Application Insights en meer over Azure en .NET. Lees zijn blog op https://blog.martdegraaf.nl.



Maximale reactie tokens

De instelling 'Max response' stelt een limiet in op het aantal tokens dat per antwoord kan worden gebruikt. Ik had betere resultaten met complexe prompts met de het maximale aantal tokens iets boven de 2000.

Temperatuur

De temperatuur is een instelling die de willekeurigheid van het model regelt. Hoe hoger de temperatuur, hoe willekeuriger de resultaten zullen zijn. Voor meer theoretische informatie zie: https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/.
De instelling Top P werkt net als de temperatuurinstelling, 0 is een lage waarschijnlijkheid en 1 is een hoge waarschijnlijkheid. Hoe hoger de waarde, hoe meer diverse waarschijnlijkheden worden toegestaan.

Code documenteren met behulp van OpenAI

Een van de functionaliteiten die je zou kunnen bedenken is het documenteren van code. Bij klanten worden veel mermaid diagrams gebruikt. Mermaid diagrams zijn op tekst gebaseerd en kunnen daarna eenvoudig worden bijgehouden in een wiki in zowel GitHub als Azure DevOps. Laten we GPT eens vragen een fibonacci functie te documenteren middels een mermaid diagram.
Afbeelding 8
GPT kwam met het volgende resultaat: **Afbeelding 9**
Wanneer je het mermaid diagram rendered krijg je het onderstaande resultaat. Wat vind je van het resultaat? Kan dit code inzichtelijk maken voor niet technische mensen? Is dit een nieuwe tool voor in onze toolbox? **Afbeelding 10**

Gegevens toevoegen met Cognitive Search

Het is mogelijk documenten toe te voegen als bron voor GPT. Dit kan worden gedaan met behulp van de

8

```
create a mermaid diagram for the flow that is done in this c# code

public static voidFibonacci_Iterative(int len)
{
    int a = 0, b = 1, c = 0;
    Console.Write("{0} {1}", a,b);
    for (int i = 2; i < len; i++)
    {
        c= a + b;
        Console.Write(" {0}", c);
        a= b;
        b= c;
    }
}
```

9

```
To create a mermaid diagram for the flow in the given C# code, you can follow these steps:
Start by creating a sequenceDiagram block in your mermaid code.
Add a participant for the Fibonacci_Iterative method.
Add a participant for the Console class.
Add a note to the Fibonacci_Iterative participant to indicate that it takes an integer parameter
len.
Add a note to the Console participant to indicate that it writes to the console.
Add a loop block to represent the for loop in the code.
Inside the loop block, add a note to indicate that c is the sum of a and b.
Inside the loop block, add a note to indicate that a is assigned the value of b.
Inside the loop block, add a note to indicate that b is assigned the value of c.
Connect the Fibonacci_Iterative participant to the Console participant with an arrow to indicate
that it writes to the console.

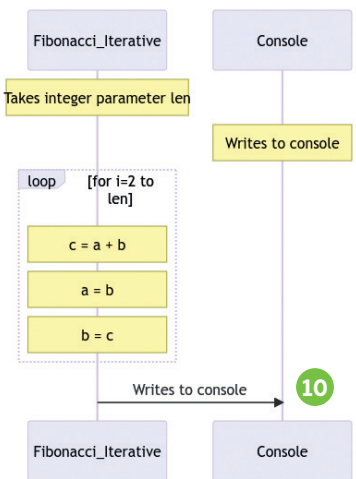
Here's an example of what the mermaid code could look like:

```mermaid
sequenceDiagram
    participant Fibonacci_Iterative
    participant Console

    Note over Fibonacci_Iterative: Takes integer parameter len
    Note over Console: Writes to console

    loop for i=2 to len
        Note over Fibonacci_Iterative: c = a + b
        Note over Fibonacci_Iterative: a = b
        Note over Fibonacci_Iterative: b = c
    end

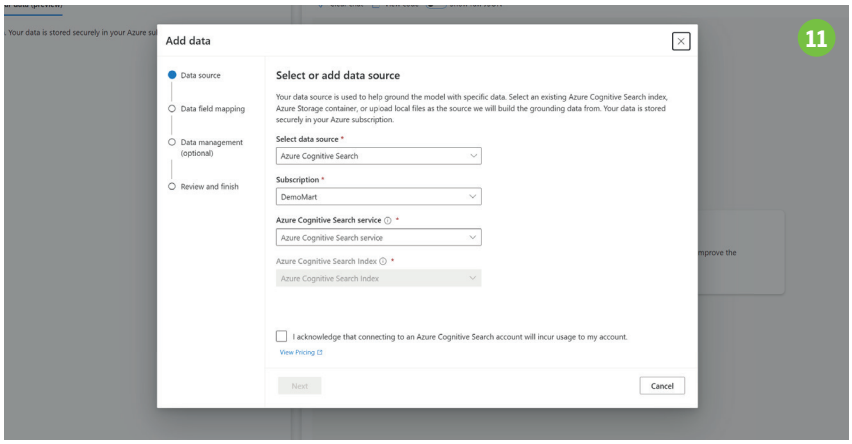
    Fibonacci_Iterative->>Console: Writes to console
```
```



Cognitive Search resource. Je kan gegevens uploaden en gebruiken om het model te voeden.
In de OpenAI playground klik je op Add your data (preview)' en klik je op '+ Add data source'. **Afbeelding 11**

Structuur

Ongeacht wat je kiest in de vervolgkeuzelijst van de gegevensbron, krijg je altijd een Blob storage met enkele blobbestanden en een Cognitive Search resource. Zorg ervoor dat je bij het maken van de Cognitive Search-resource dezelfde regio selec-



teert als de OpenAI-resource en dat het ten minste de SKU Basic heeft.

Stappen om gegevens toe te voegen



> Maak een Blob-opslagaccount aan en plaats bestanden in een container. Ik raad aan om Azure Storage Explorer te gebruiken. Upload enkele PDF-bestanden waarop je wilt kunnen zoeken. Of gebruik HTML-pagina's van een openbare website als voeding voor de gegevensbron.

> Maak een Cognitive Search-resource. Dit moet minstens Basic zijn, de prestaties zijn beter op hogere SKU, maar de kosten zijn ook aanzienlijk hoger.

> Maak een gegevensbron. Gebruik hiervoor de Azure Playground.

- Selecteer 'Azure Blob storage'.
- Selecteer de Blob storage die je hebt gemaakt.
- Selecteer de container die je hebt gemaakt of wilt gebruiken.
- Selecteer de Cognitive Search-resource die je hebt gemaakt.
- Voer een indexnaam in.
- Vink het vakje aan dat je op de hoogte bent van de prijzen. En ga verder.
- Klik op 'Save and Close' op de pagina 'Save and Close'.
- Wacht tot de indexer is voltooid. Je kan de voortgang van de indexer bekijken in de Cognitive Search resource. Ga naar het tabblad Indexers of Indexen. Indexers worden uitgevoerd om de index te maken. In het Index tabblad staat de index die je zojuist hebt gemaakt, als deze documenten bevat is hij klaar. **Afbeelding 12**

Chatten met documenten

Nu de indexer klaar is en de index is geladen, gaan we chatten met documenten. De Blob-opslag wordt gevoed met een presentatie van mijn Git-presentatie als PDF. Laten we GPT vragen stellen over Git. **Afbeelding 13**
Let op: in de schermafbeelding kan je ervoor kiezen om reacties te beperken tot alleen de inhoud van jouw databronnen. Bronnen worden in de reactie weergegeven, ook als dit er meerdere zijn. Dit is een geweldige manier om inzicht te krijgen in gegevens en grote documenten.

Kostenbeheer

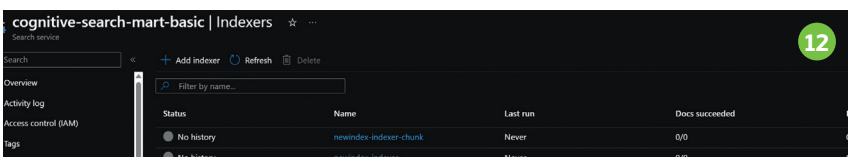
De basis-SKU voor Cognitive Search is een model voor betaling per maand, wat betekent dat je de resource moet verwijderen nadat je klaar bent met testen. De OpenAI-resource is een model voor betaling per gebruik, evenals de opslagaccount.

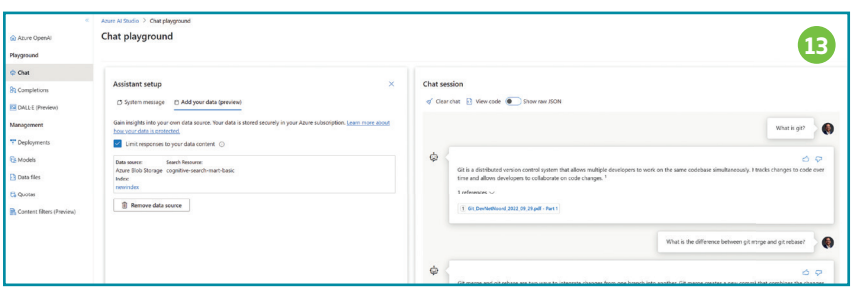
Releasen naar een App Service

We hebben nu wat kracht of wat domheid van het OpenAI-model gezien. Hetzelfde resultaat als in de Chat Playground kan je ook releasen naar een App Service. Dit kan op een gratis SKU, maar weet dat na deze actie je applicatie openbaar toegankelijk is via {uw-app-naam}.azurewebsites.net. Het is noodzakelijk om VNET-integratie te configureren als je te maken heeft met bedrijfsinformatie. Daarnaast is het aan te raden om alleen public access te gebruiken voor het testen van het OpenAI services. Niet alleen de data maar ook vanwege mogelijke kosten die het met zich meebrengt bij overmatig gebruik.

Overzicht van de resourcegroep

Als afsluiting van alle resources die we hebben aangemaakt in de delen hiervoor, kunnen we de volgende resources in onze resourcegroep zien. Ik weet dat jullie allemaal gaan testen of MartGPT nog steeds online is, daarom staat public access uit. **Afbeelding 14**





Gebruik van OpenAI API-services via C#

De OpenAI API is een REST API waarmee je de modellen in jouw applicaties kunt gebruiken. Je kan de API gebruiken om completions uit te voeren en chats te voeren met GPT. Het echte vermogen komt naar voren wanneer we de OpenAI-services via onze code kunnen gebruiken, waar we ons eigen bedrijfsscope kunnen creëren.

Voorbeeldcode

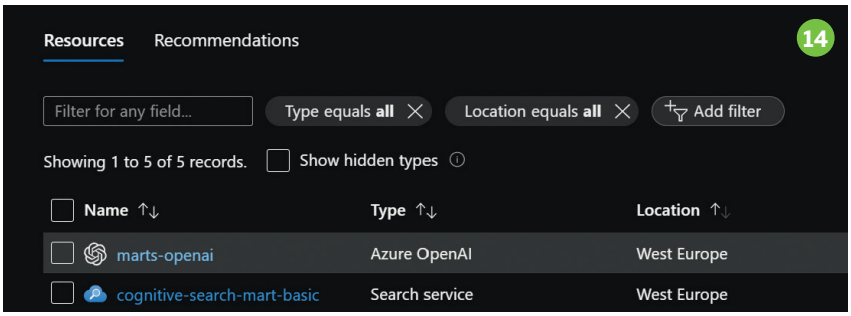
Zorg ervoor dat je 'Inclusief prerelease' hebt ingeschakeld bij het zoeken naar het NuGet-pakket 'Azure.AI.OpenAI'. Onderstaand script geeft een dier als input en verwacht er een categorie en een korte tekst uit in JSON. **Afbeelding 15**

Om het gegeven antwoord heb ik smakelijk kunnen lachen. Wat wordt getoond, is dat zelfs Smurfen in de categorie Landdieren wordt geplaatst. Zelfs wanneer je de parameters van bijvoorbeeld de temperatuur niet aanpast, krijg je elke keer een ander antwoord. Je moet zeer precies zijn als je de uitvoer wilt standaardiseren. Het is echt run en test, volgorde en stijl zijn in bovenstaande response ook niet altijd gelijk. **Afbeelding 16**

Denk aan de mogelijkheden door de temperatuur te verlagen en gematerialiseerde categorieën te maken van enkele enorme teksten ingevoerd door gebruikers, bijvoorbeeld in een Azure Function die reageert op business events. Denk bijvoorbeeld aan contactformulieren, of email dat automatisch wordt ingedeeld.

Conclusie

Ik wilde gewoon de OpenAI API uitproberen en de mogelijkheden ervan zien. Binnenkort zullen bedrijven klaar zijn om AI te gebruiken voor hun gebruiksscenario's en als consultant moet je hen kunnen adviseren over de mogelijkheden. Ik heb geleerd dat het GPT-model statistisch gezien een uitvoer kiest en dat de reactie moet worden geverifieerd.







Daarom noemt Microsoft zijn producten 'Copilot'. AI helpt je en is supercool, maar het is niet perfect. ●

Bronnen en gerelateerde content

https://learn.microsoft.com/en-us/legal/cognitive-services/openai/overview
https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/completions#completion
https://blog.iusmentis.com/2023/03/21/van-wie-is-mijn-werk-als-ik-chatgpt-mijn-werk-laat-doen/
https://blog.iusmentis.com/2023/06/08/mag-een-iso27001-gecertificeerde-organisatie-chatgpt-gebruiken/
https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

C# Minimal APIs: minimale code voor maximale opbrengst

Door de jaren heen zijn er naast doorontwikkeling van bestaande technologieën veel nieuwe mogelijkheden beschikbaar gekomen op het gebied van webontwikkeling en dan met name het ontwikkelen van HTTP API's. Met de komst van ASP.NET Core is er een belangrijke nieuwe technologie beschikbaar gekomen: Minimal API's, een eenvoudigere en efficiëntere manier om met minimale afhankelijkheden en minder ceremonies HTTP API's te bouwen.

Auteur: Pieter Baart

In dit artikel zullen we antwoord geven op de volgende vragen:

> Zijn Minimal API's de silver bullit?
> Zijn Minimal API's slechts geschikt om proof of concepts te maken?
> Moeten Minimal API's vermeden worden in een productie omgeving?

Wat zijn Minimal API's?

Minimal API's zijn de tegenhanger van controller-based HTTP API's. Het is een nieuwe functionaliteit in ASP.NET Core die het proces van het bouwen van deze API's vereenvoudigt. Op die manier kunnen ontwikkelaars sneller en met minimale overhead HTTP API's ontwikkelen.
Met Minimal API's kunnen software developers HTTP API's bouwen met slechts een paar regels code waardoor het ontwikkelproces sneller en eenvoudiger wordt. Het doel van deze nieuwe technologie is om de complexiteit van het bouwen van API's te verminderen en het daardoor voor ontwikkelaars gemakkelijker te maken om snel aan de slag te gaan met het opzetten en ontwikkelen van endpoints.
Oké, klinkt goed, maar hoe ziet zo'n Minimal API er dan eigenlijk uit? De volgende code snippet laat zien hoe in 4 regels code een werkend voorbeeld van een HTTP API wordt opgezet: **Afbelding 1**

```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();
app.MapGet("/", () => "Hello World!");
app.Run();
```

1

En that's it: in slechts 4 regels code hebben we een werkend voorbeeld van de REST API. Geen onnodig gebruik van usings, geen controllers, alles in slechts één file: de progam.cs file. Als opmerking moet worden toegevoegd dat hiervoor gebruik wordt gemaakt van Visual Studio 2022 en .NET 6 of hoger. Analoog aan bovenstaande code snippet kunnen we deze uitbreiden met andere meest gebruikte REST API calls:

Een Minimal API bestaat dus uit een aantal HTTP methoden met ieder 2 parameters: de endpoint route en een delegate functie die het HTTP request afhandelt.

Minimal API's versus controller-based API's

Zoals eerder vermeld zijn Minimal API's de tegenhanger van controller-based HTTP API's. Het belangrijkste verschil tussen minimal API's en controllers is de manier waarop ze zijn ontworpen en hoe ze werken. Controller-based API's zijn onderdeel van het traditionele MVC (Model-View-Controller) patroon en bevatten de logica voor het verwerken van HTTP requests. Dit betekent dat ze onder water het volledige MVC framework en pipelines van ASP.NET meekrijgen. Minimal API's zijn voornamelijk "synthetic sugar" waarbij optimalisaties op de routing zijn toegepast omdat de minimal api enkel voor api purposes gebruikt wordt. **Afbelding 2**

```
[Route("/hello")]
[ApiController]
public class HelloWorldController : ControllerBase
{
    [HttpGet]
    public IActionResult Hello()
    {
        return Ok("World");
    }
}
```

2

```
app.MapGet("/hello", () => "World");
```

Minimal API's zijn ontworpen als een lightweight alternatief voor controller-based API's en zijn bedoeld om eenvoudige RESTful API's

te maken zonder de extra overhead van de traditionele MVC-architectuur. In plaats van dat er een aparte controller nodig is voor elk HTTP request, bevatten minimal API's slechts één enkele methode die wordt gebruikt om HTTP requests af te handelen en te reageren op basis van de inhoud van het request.
Een ander verschil is de manier waarop routing werkt. In traditionele controller-based API's is het mogelijk om de routing op basis van controller naam en method name te doen (wat overigens in de praktijk eigenlijk weinig toegepast wordt), terwijl minimal API's routing bepalen op basis van de HTTP-methode en de bijbehorende endpoint, die wordt gedefinieerd in de code. **Afbelding 3**

```
app.MapGet("/todos/{text}", (string text) => db.Todos.Where(t => t.Text.Contains(text));
app.MapGet("/posts/{*rest}", (string rest) => $"Routing to {rest}");
app.MapGet("/todos/{id:int}", (int id) => db.Todos.Find(id));
app.MapGet("/posts/{slug:regex(^[a-z0-9_-]+$)}", (string slug) => $"Post {slug}");
```

3

Minimal API's en controller-based API's zijn beide manieren om een RESTful API te maken in C#, maar er zijn enkele belangrijke verschillen tussen beide technologieën:

- Code-structuur: bij het maken van controller-based API's moet je een hiërarchische structuur creëren voor de gewenste endpoints en acties, terwijl bij het gebruik van minimal API's alles binnen één enkel bestand blijft
- Routing: met minimal API's moet je de routes van de endpoints expliciet definiëren.
- Minder overhead: Minimal API's zijn ontworpen om minder overhead te hebben dan traditionele controller-based API's, waardoor ze sneller kunnen werken en minder code vereisen
- Eenvoudiger: Minimal API's zijn over het algemeen eenvoudiger in gebruik dan controller-based API's, omdat ze minder configuratie en minder boiler-plate code vereisen
- Flexibiliteit: controller-based API's zijn over het algemeen flexibeler in het ondersteunen van meer complexe routing en aanpassingsopties, terwijl Minimal API's meer geschikt zijn voor kleinere projecten en minder complexe API's

Het belangrijkste verschil tussen minimal API's en controller-based API's zit hem dus in de aanpak die je kiest om een RESTful API te creëren. Minimal API's zijn ontworpen om eenvoudige, lightweight RESTful API's te maken waar controller-based API's worden gebruikt voor het afhandelen van HTTP-verzoeken in een (volledigere) MVC-architectuur.

Waarom Minimal API's gebruiken?

Het gebruik van Minimal API's heeft verschillende voordelen boven het gebruik van controller-based API's. Ten eerste zijn ze eenvoudig en gebruiksvriendelijk. In tegenstelling tot traditionele controller-based API's vereisen Minimale API's (zoals de naam al suggereert) minimale code en kunnen ze snel worden gemaakt. Daarnaast bieden Minimal API's een betere/hogere performance dan traditionele controller-based API's. Omdat ze minder overhead nodig hebben, zijn ze sneller en efficiënter. Dit is vooral belangrijk voor webapplicaties die een groot aantal HTTP requests moeten verwerken.
Er zijn dus verschillende redenen waarom het te overwegen is om Minimal API's te gaan gebruiken als je (REST) API's wilt gaan ontwikkelen:

- Eenvoud: Minimal API's zijn ontworpen om eenvoudig en makkelijk in gebruik te zijn. Er is weinig boiler-plate code nodig om Minimal API's op te zetten wat resulteert in het snel kunnen opzetten van endpoints.
- Performance: Minimal API's bieden een betere performance dan traditionele controller-based API's omdat ze minder overhead hebben. Dit maakt ze ideaal voor webapplicaties die een groot aantal requests moeten afhandelen.
- Proof of Concept: vanwege de eenvoud is het makkelijk om snel een simpele CRUD API te maken ten behoeve van prototyping e.d.
- Volledige ondersteuning van het .NET Core framework waardoor je altijd later kunt uitbreiden met controllers.
- Tot slot vormen Minimal API's een goede basis voor het ontwikkelen binnen een microservices architectuur.

Samenvattend kunnen we stellen dat, als we snel, efficiënt en high-performance webapplicaties en API's willen



Pieter Baart
Lead developer Bergler Software Solutions

bouwen, Minimal API's zeker het overwegen waard zijn. Ze bieden een eenvoudigere en flexibelere benadering voor het bouwen van API's die kunnen helpen projecten sneller en met minder code te ontwikkelen.

Waarom géén Minimal API's gebruiken?

Zijn Minimal API's de silver bullet? Nee, dat is zeker niet het geval. Het biedt een andere manier voor het schrijven van HTTP REST API's naast de al bestaande mogelijkheid met behulp van controller-based API's. Zo zijn er een aantal redenen te bedenken waarom je juist géén Minimal API's wilt gebruiken:

- Complexiteit: hoewel Minimal API's ontworpen zijn om simpeler te zijn dan traditionele API's, kan het voor sommige ontwikkelaars een uitdaging zijn om te werken met de minder gestructureerde en meer flexibele aanpak van Minimal API's. Daarnaast gaat bij gebruik van een groot aantal API's het overzicht al snel verloren, waar dit bij controller based API's door het gebruik van controllers gestructureerder is
- Beperkte functionaliteit: omdat Minimal API's zijn ontworpen om lightweight te zijn, hebben ze mogelijk niet alle functies en mogelijkheden die gewenst zijn voor een project.
- Leercurve: Minimal API's vereisen een nieuwe manier van denken en programmeren, wat kan leiden tot een steile leercurve.
- Gebrek aan documentatie: aangezien Minimal API's nog relatief nieuw zijn, kan het zijn dat er nog niet veel documentatie en informatie beschikbaar is over het gebruik ervan.
- Mogelijkheid van toekomstige veranderingen: omdat Minimal API's nog in ontwikkeling zijn, is er de mogelijkheid dat er in de toekomst veranderingen worden aangebracht in de manier waarop ze werken of dat ze niet meer worden ondersteund, wat kan leiden tot problemen met toekomstige updates of onderhoud.

Hoewel minimal API's een aantrekkelijke optie kunnen zijn voor sommige ontwikkelaars, is het belangrijk om deze mogelijke uitdagingen en beperkingen in overweging te nemen voordat je besluit om ze te gebruiken voor jouw project.

Overige features van Minimal API's

Omdat Minimal API's de meest belangrijkste features van ASP.NET Core implementeert (zoals o.a. model binding, authenticatie en autorisatie, dependency injection, parameter binding (beschikbaar vanaf .NET 7)) zijn ze zeker ook geschikt voor productie omgevingen.

Model binding

C# Minimal API's ondersteunen het gebruik van model binding. Model binding biedt de mogelijkheid om HTTP request data op C# objecten te mappen. Dit maakt het eenvoudiger om te werken met complexe data objecten en vergt nog minder boiler-plate code.

Dependency Injection

Een van de meest gebruikte patterns is Depedency Injection (DI). In Minimal API's is het vrij eenvoudig om DI toe te passen. In onderstaand voorbeeld wordt "todoService" geïnjecteerd. **Afbelding 4**

Voordeel van het gebruik van Depedency Injection is dat de overzicht van de Minimal API's in de program.cs class wordt behouden terwijl de logica in daarvoor specifieke classes wordt geplaatst.

Parameter binding

Parameter binding houdt in dat inkomende HTTP-requests worden gekoppeld aan method parameters. Hierdoor kunnen deze requests op een gestructureerde en efficiënte manier worden verwerkt. Minimal API's in ASP.NET Core 7 bieden verschillende typen parameter binding, waaronder FromQuery, FromRoute, FromHeader en FromBody. **Afbelding 5**

Conclusie

Minimal APIs kan gezien worden als een alternatief voor het bouwen van je HTTP API, het is zeker niet bedoeld om ASP.NET Core MVC te vervangen. Minimal API's bieden een eenvoudige en efficiënte manier om web applicaties en API's te ontwikkelen. Dit alles met minimale overhead, minimal gebruik van boiler-plate code met als resultaat high performance applicaties die in staat zijn grote aantallen HTTP requests in korte tijd af te handelen. Minimal API's zijn met name geschikt voor gebruik in projecten waarbij een gering aantal API endpoints gebruikt wordt. Al met al zeker het overwegen waard in combinatie met de overstap naar .NET6 en hoger. ●

```
app.MapDelete("/todo/{id}", async (int id, ITodoService todoService) => {
    await todoService.Delete(id);
    return Results.Accepted();
});
```
4

```
app.MapPost("/search/{searchString}", (
    string searchstring,
    int take,
    int skip,
    SortingOptions sorting,
    ISearchService searchService,
    [FromHeader(Name = "X-HEADER-KEY")]string headerKey) => { });
```
5

Breng virtuele kunst tot leven met ARCore VPS

Sta je weleens op een leeg plein en denk je dan "Kan dit niet iets leuker?". Je kan naar de gemeente gaan en een aanvraag doen om er kunst te laten plaatsen. Maar dit klinkt voor mij als software ontwikkelaar als te veel moeite en te veel sociale interactie. Daarom gaan we in dit artikel een stuk digitale kunst plaatsen die voor ons klaar staat als we weer een keer op een saaie locatie zijn.

Auteur: Bart Klomp

Dit kunnen we doen met een principe dat het Visual Positioning System (VPS) genoemd wordt. VPS is een systeem dat gebruik maakt van je camera om je precieze locatie te bepalen. Dit principe werkt vergelijkbaar met systemen, zoals Spatial Anchors, alleen dan is er al een bestaande spatial mapping van specifieke locaties. Vaak gebeurt dit door de camerabeelden te vergelijken met herkenbare punten, zoals gebouwen of kunstwerken. Op het moment heb je twee grote systemen die dit kunnen: Lightship VPS van Niantic en ARCore GeoSpatial API van Google. Beide systemen werken net iets anders. Niantic's Lightship VPS maakt gebruik van je GPS locatie in combinatie met data die zij verzameld hebben uit hun diverse spellen, zoals Pokémon GO. Als onderdeel van dit spel kunnen spelers bezienswaardigheden scannen in ruil voor items binnen het spel. Een tijdje na een scan wordt dit gebied beschikbaar binnen het VPS van Niantic. Niantic heeft op dit moment vooral data over gebieden met bestaande kunstwerken of andere interessante punten.

Bij Google's ARCore GeoSpatial API wordt er andere data gebruikt. Google heeft namelijk het voordeel dat ze ooit al heel veel gebieden gescand hebben voor Google Streetview. Hierdoor is hun systeem beschikbaar op alle plekken waar Google Streetview beschikbaar is. Wel ben je hierdoor vooral gelimiteerd door lokalisatie in straten. Het werkt ook het beste als er gebouwen in de buurt zijn.

Kortom beide systemen hebben voor- en nadelen, maar in dit artikel gaan we in op de technologie van Google. Deze is wat breder beschikbaar en de mogelijkheden voor het plaatsen van content zijn net iets uitgebreider. Wel komen er in het systeem van Niantic in de toekomst veel features om naar uit te kijken.

Getting started

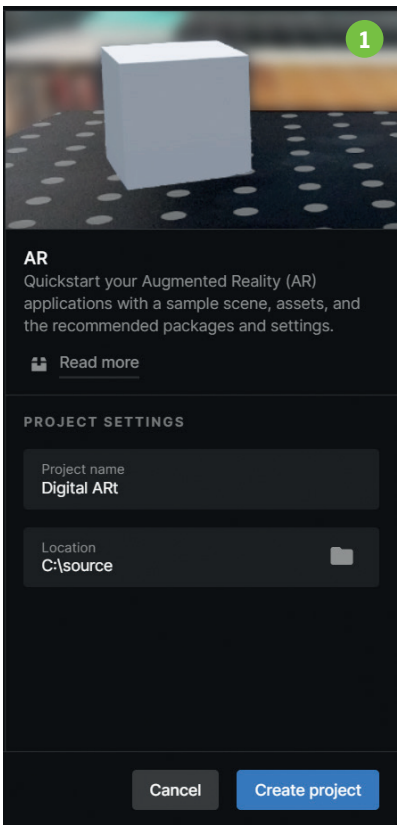
Om onze digitale kunst werkelijkheid te maken, gaan we gebruikmaken van de volgende onderdelen:

> Unity (Ik gebruik 2021.3.26)
> ARCore
> ARCore Extensions

Om te beginnen hebben we een Unity installatie nodig met het build pakket voor Android. Er zijn ook mogelijkheden om dit werkend te krijgen voor iOS, om het artikel iets korter te houden gaan we daar nu niet op in. In Unity kunnen we vervolgens een nieuw project aanmaken. Om het jezelf makkelijk te maken, kun je gebruikmaken van de AR template. Deze template is te vinden bij het aanmaken van een nieuw Unity project en zorgt ervoor dat je project AR ready is (**afbeelding 1**).

ARCore en extensions

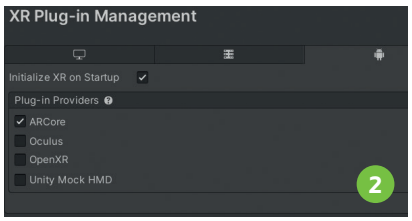
ARCore is onze library die ervoor zorgt dat AR-functionaliteiten werken op Android, dit gebeurt via AR Foundation. Veel van de basisfuncties zitten hier in. Met AR Core Extensions krijgen we de mogelijkheid om gebruik te maken van de GeoSpatial API.

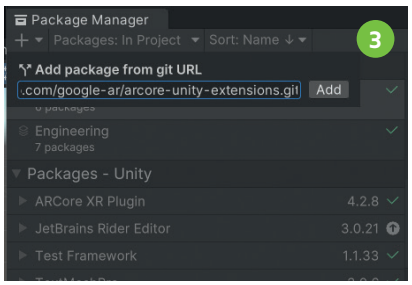


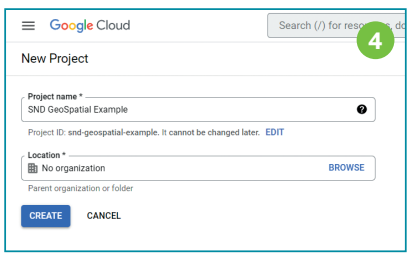
Om deze twee componenten toe te voegen ga je in Unity naar **Edit -> Project settings -> XR Plug-in management**. Hier moet je voor Android ARCore aanvinken (**afbeelding 2**).
Daarna moet je in Unity naar **Window -> Package Manager** gaan. Hier moet je naar **+ -> Add package from git URL** gaan. Voeg dan het pakket toe via de URL: https://github.com/google-ar/arcore-unity-extensions.git. (**afbeelding 3**)

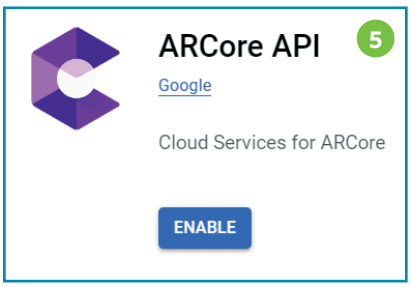
Google cloud API key

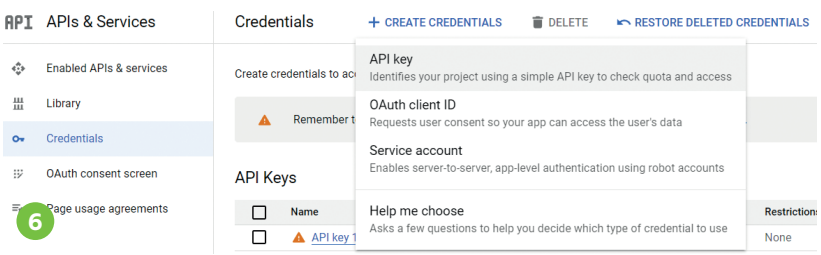
Om gebruik te maken van de GeoSpatial API moeten we bij Google kunnen authenticeren. Hiervoor moet je in de Google Cloud Console (console.cloud.google.com) een account en een project aanmaken. Om een project aan te maken ga je naar je project selectie en ga je naar New Project. Kies hier een geschikte naam voor je project (**afbeelding 4**).
Vervolgens ga je in het menu naar **APIs & services -> Library** (te vinden in het menu aan de linkerkant van de pagina). Zoek hier naar ARCore en klik op "Enable" (**afbeelding 5**).
Zodra je dit gedaan hebt, wordt je naar de ARCore pagina gestuurd van je project. Hier ga je naar Credentials -> Create credentials -> API Key. Kopieer deze sleutel en bewaar deze goed. (**Afbeelding 6**)

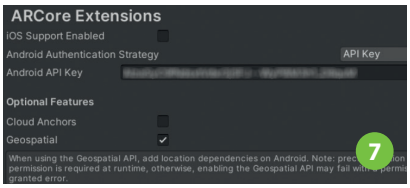


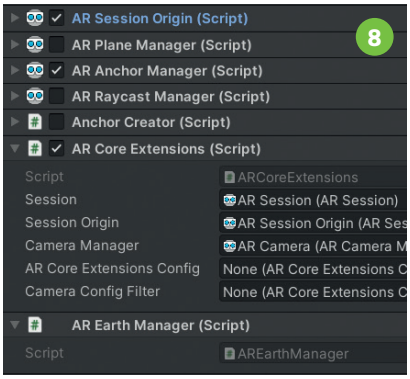












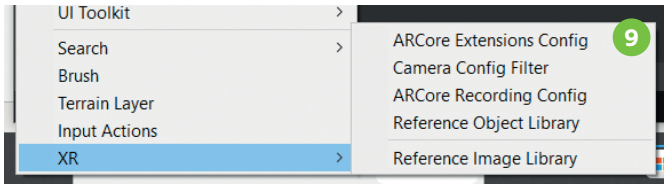
GeoSpatial API opzetten

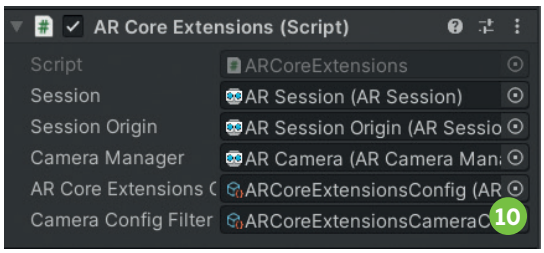
Om de GeoSpatial API aan te zetten ga je naar **Edit -> Project settings -> XR Plug-in management -> ARCore extensions**. Hier kies je voor Android Authentication Strategy API Key. Vervolgens vul je in het veld je eerder gemaakte Key in. In dit scherm moet je ook Geospatial aanvinken (**afbeelding 7**).
In Unity kunnen we nu naar het AR Session Origin object gaan, dat in onze template scene is aangemaakt. In dit object zijn een aantal componenten te zien. Voor nu kan je AR Plane Manager, AR Raycast Manager en Anchor Creator uitzetten. Vervolgens kan je de componenten AR Core Extensions en AR Earth Manager toevoegen. Link vervolgens de bijbehorende objecten aan de gevraagde velden van dit component. De uitzonderingen hier zijn AR Core Extensions Config en Camera Config Filter. (**afbeelding 8**)

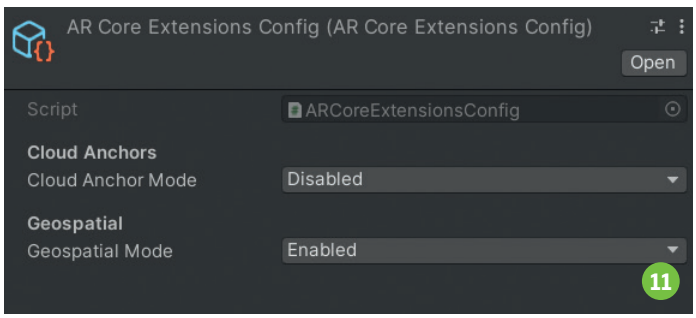
In je Asset scherm kan je deze twee objecten aanmaken. Ga naar **+->XR** en maak een AR Core Extensions Config en Camera Config Filter aan (**Afbeelding 9**).
In de AR Core Extensions Config moet je Geospatial aanzetten (**afbeelding 10**). Link deze objecten vervolgens aan het AR Core Extensions Component. (**afbeelding 11**)

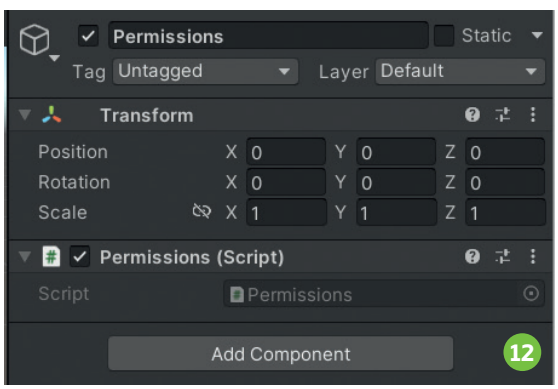
Android permissions

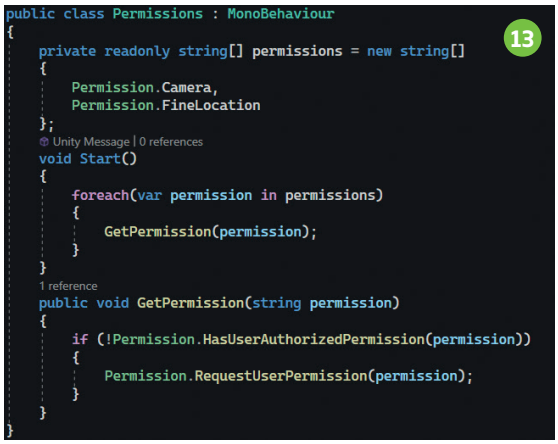
Onze gebruikers moeten de applicatie toestemming geven om bepaalde services te gebruiken. Specifiek hebben we camera en precieze locatie tracking nodig. Om toestemming te vragen hebben we een script nodig onder een GameObject.
Maak eerst een nieuwe leeg GameObject aan in de Scene. Noem deze Permissions. Klik vervolgens op Add Component en maak een nieuw script aan Permissions. (**Afbeelding 12**)

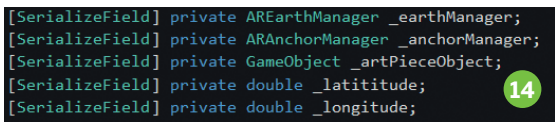












In dit script zorgen we ervoor dat als de Scene start, we Permission.RequestUserPermission(permission); aanroepen voor alle toestemming die we nodig hebben (**afbeelding 13**). Als dit script gemaakt is, kunnen we deze onder een nieuw GameObject plaatsen in onze scene. Permissio zit in het package UnityEngine.Android;

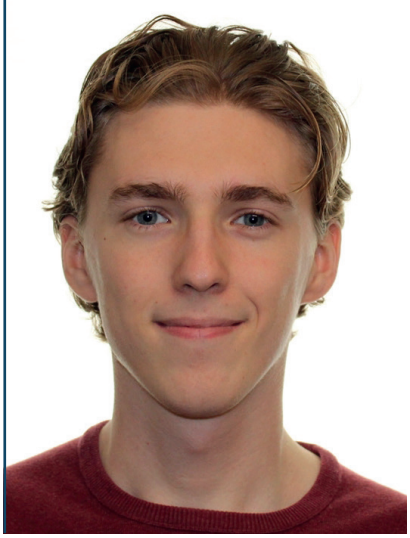
Content plaatsen in de wereld

Na alles opgezet te hebben, kunnen we eindelijk aan de slag met het plaatsen van content in de wereld. Om te starten maken we een nieuw script aan en hangen we deze aan een GameObject. Noem het script bijvoorbeeld CoordinateContentPlacer. In dit script maken we een 5 tal velden aan die we in kunnen vullen via de editor, zie de afbeelding. De earth en anchor managers bestaan al in de scene onder de XR-origin. De andere drie variabelen komen we later op terug (**afbeelding 14**).

BIO

Bart Klomp

Bart Klomp is een software engineer bij de Microsoft afdeling van Ordina (MTech). Hij specialiseert zich in .NET, XR en Azure. In zijn vrije tijd doet hij veel vrijwilligerswerk voor Curling Club Tilburg. Linkedin: https://www.linkedin.com/in/bart-klomp-034b93142/ Adres: Smidspad 17A, Tilburg, 5046JA



Visual Positioning System (VPS) met de camera de locatie te bepalen.

Vervolgens maken we logica in de update functie van het script. We moeten namelijk wachten tot onze locatie bepaald is. Dit kunnen we doen door te checken wat de status is van de earth manager, maar we willen daarnaast ook weten hoe accuraat onze positie is. Je zult merken dat als je dit niet doet, dat je content verder van de gewilde locatie is en niet altijd even stabiel blijft staan. Daarvoor kijken we naar de CameraGeospatialPose. Deze heeft een variabele OrientationYawAccuracy. Dit geeft een waarde terug die het aantal graden vertegenwoordigt waarin het systeem met een nauwkeurigheid van 65% kan aangeven waar de oriëntatie (yaw) zich bevindt. Makkelijk gezegd: hoe groter dit nummer, hoe minder accuraat de tracking. In dit voorbeeld checken we of het kleiner is dan 15 grade (**afbeelding 15**).
Als we weten dat de positie accuraat is, dan kunnen we onze content instantiëren. Hiervoor maken we gebruik van de latitude en longitude die we eerder gedefinieerd hebben. Ook hebben we een altitude nodig, voor nu gebruiken we die van de camera. Er zijn ook andere manieren om de hoogte te bepalen, zoals zelf ingestelde waarden of door Terrain Anchors te gebruiken. Voor nu doen we het met een vaste waarde. We moeten ook een variabel toevoegen om ervoor te zorgen dat we het object maar een keer instantiëren (spawned) (**afbeelding 16**).
Nu moeten we CoordinateContentPlacer aan een GameObject koppelen en de velden invullen. Zoek de longitude en latitude van de gewenste locatie (tip: gebruik Google maps met satellietmodus). En koppel een prefab met jouw kunstwerk. (**afbeelding 17**)
Vervolgens kan je de app bouwen naar je Android telefoon en naar de fysieke locatie gaan. Kijk zodra je op de locatie bent om je heen om een goede positionering te krijgen (**afbeelding 18**).

Afronding

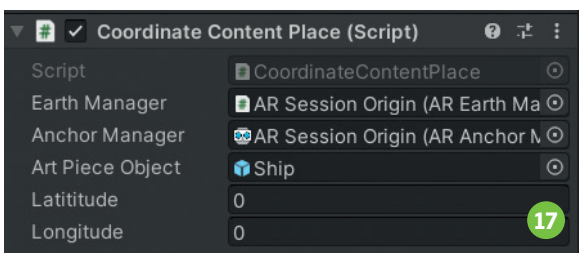
Zoals je hebt gezien is het mogelijk om met weinig programmeerwerk een model in AR op een vooraf nog niet bezochte locatie te plaatsen. Google biedt ook hoogtekaarten die je kan gebruiken om de altitude vooraf in te schatten, zodat je dit niet op de hoogte van je apparaat hoeft te baseren. Ook kun je gebruikmaken van de plane detection om objecten zo op de grond te plaatsen. Ten slotte werkt deze API ook op iOS via AR Foundation. Hiervoor zijn goede instructies te vinden in de AR Core documentatie. ●

```
var earthTrackingState = _earthManager.EarthTrackingState;
var pose = _earthManager.CameraGeospatialPose;
if(earthTrackingState == TrackingState.Tracking && pose.OrientationYawAccuracy < 15)
{
```

15

```
private bool spawned = false;
Unity Message | 0 references
private void Update()
{
    if (!spawned)
    {
        var earthTrackingState = _earthManager.EarthTrackingState;
        var pose = _earthManager.CameraGeospatialPose;
        if (earthTrackingState == TrackingState.Tracking && pose.OrientationYawAccuracy < 15)
        {
            var altitude = pose.Altitude;
            var anchor = _anchorManager.AddAnchor(_latititude, _longitude, altitude, pose.EunRotation);
            Instantiate(_artPieceObject, anchor.transform);
            spawned = true;
        }
    }
}
```

16

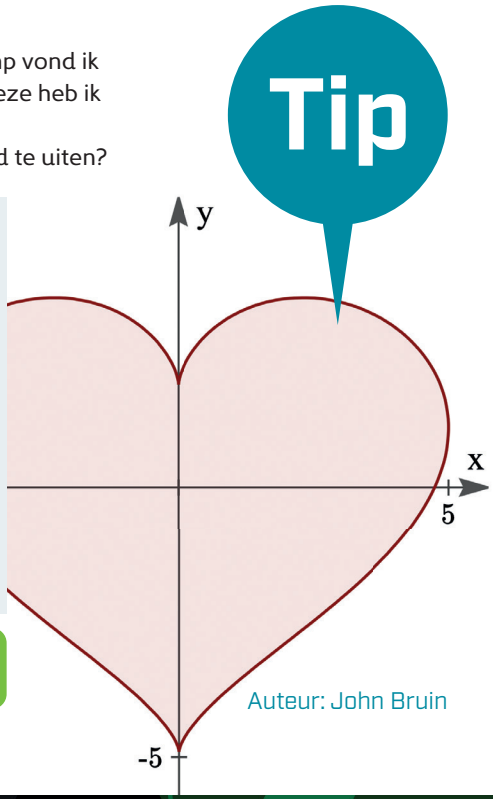




Op https://www.intmath.com/trigonometric-graphs/7-lissajous-figures.php vond ik een formule om een hartje te tekenen met mathematische functies en deze heb ik omgezet naar HTML en Javascript.
Misschien een leuke geeky manier om als ontwikkelaar jouw liefde aan iemand te uiten?

```
<html>
    <head>
        <title>I love you</title>
    </head>
    <body>
        <canvas id="canvas" width="200" height="200"></canvas>
    </body>
    <script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext('2d');
        context.fillStyle = "red";
        for (var i = 0; i < 360; i++)
        {
            var radians = i / 360 * Math.PI * 2;
            var x = 100 - (80 * Math.sin(radians) * Math.sin(radians) * Math.sin(ra-
dians));
            var y = 90 - (20 * (4*Math.cos(radians) - 1.3 * Math.cos(2*radians)
- 0.6 * Math.cos(3*radians) - 0.2 * Math.cos(4*radians)));
            context.fillRect(x, y, 2, 2);
        }
    </script>
</html>
```

> Vind deze en de andere listingen uit het artikel op sdn.nl!



Auteur: John Bruin





SELMA HELPS 1.3 MILLION PEOPLE MAKE ENDS MEET

How many people do you want to help?

Selma Kubat, IT project manager at Netcompany, helped to build the system that pays out national and social pensions to over 1.3 million people.

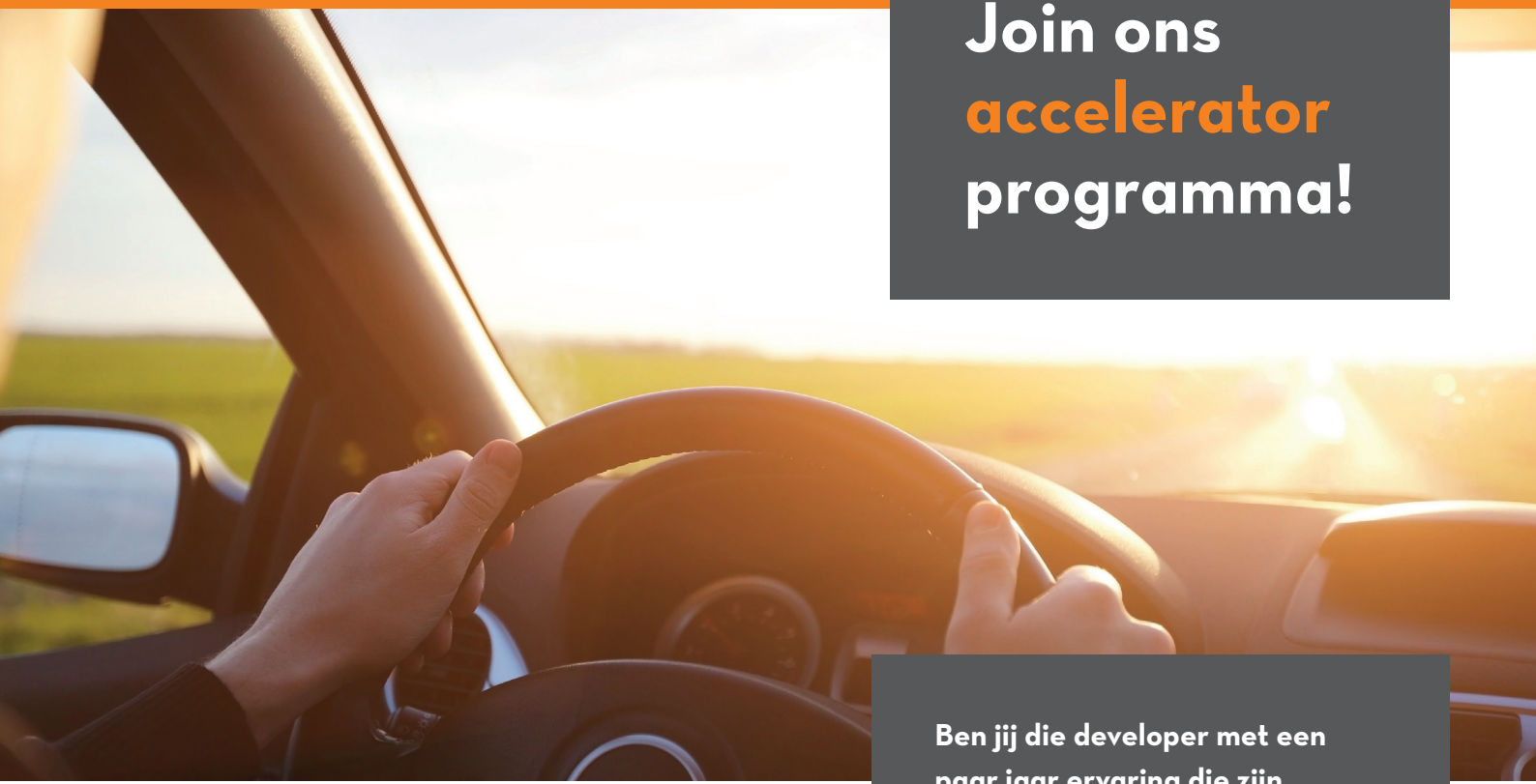
Do you want to work on IT projects that make a real difference to society?

Take a look at our open roles:

netcompany

“Heb jij graag het stuur in handen?
Met ons op maat gemaakte programma
zet je de volgende stap in je carrière!”

Join ons accelerator programma!



ORDINA

Software Development

- 1 jaar intensieve training voor en door developers
- Uitsluitend moderne technieken
- Combinatie van vakinhoud en soft skills

Ben jij die developer met een paar jaar ervaring die zijn .Net kennis naar het ‘Next Level’ wil brengen?

Word collega bij Ordina Software Development en doe mee aan dit leerzame en leuke traject!

Wil je meer informatie? Kom naar onze stand op Future Tech of mail naar barbara.pronk@ordina.nl

Microsoft • .NET6 • AZURE • DevOps • C# • OpenSource • Cloud • AI/ML • Security • PowerPlatform